

DX11-B

ADDRESSING TEST
MD-11-DZDXJ-A

EP-DZDXJ-A-DL-A
COPYRIGHT © 1977
FICHE 1 OF 1

JUN 1977
digital
MADE IN USA

This microfiche card contains a grid of frames on the left side, with a large blank area on the right. The frames are arranged in approximately 15 rows and 6 columns. Each frame contains a small, high-contrast image, likely a test pattern or a specific data point. The right side of the card is mostly blank, with a small, faint grid visible in the bottom right corner.

B01

EOF1DZKACSEQ

00010000

770608

PDP10 411

HDR1DZDXJASEQ

00010000

770608

.REM %

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDXJ-A-D
 PRODUCT NAME: DX11-B OFF-LINE ADDRESSING TEST
 DATE: APRIL 29, 1977
 MAINTAINER: DIAGNOSTIC PROGRAMMING
 AUTHOR: R. MISNER

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUET 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 BY DIGITAL EQUIPMENT CORPORATION

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

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

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-QZAU8-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.

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.

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
1582.0 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 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.

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

"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

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.

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

3.0 ERROR INFORMATION

THE GENERAL ERROR REPORT FORMAT IS AS FOLLOWS:

LINE 1

ERROR PC: XXXXXX ERROR NO. YYYY

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

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

H01

MD-11-DZDXJ-A DX11-B ADDRESSING TEST MACY11 27(1006) 02-MAY-77 10:36 PAGE 6
DZDXJA.P11 28-APR-77 09:59

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

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.

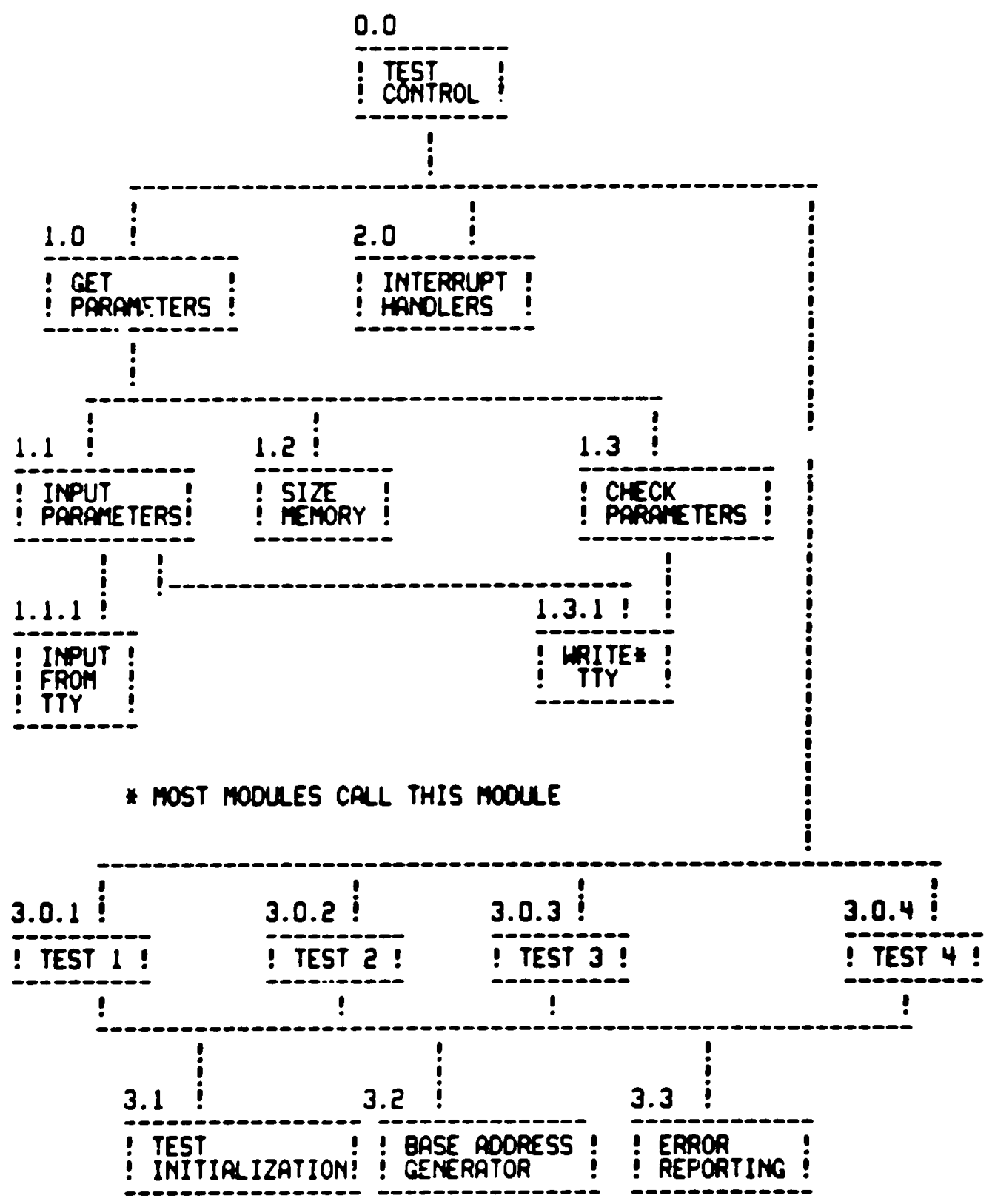
NOTE: AT THE END OF THE FIRST PASS EXECUTION THE FOLLOWING MESSAGE IS PRINTED:

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 INFORMATION 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

4.1 BLOCK DIAGRAM



298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352

J01

MD-11-DZPXJ-A DX11-B ADDRESSING TEST MACY11 27(1006) 02-MAY-77 10:36 PAGE 8
DZOXJA.P.1 28-APR-77 09:59

353
354
355
356
357
358
359

4.2 TEST DESCRIPTIONS

4.2.1 TST 1

416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471

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
.MCALL .HEADER, .SCATCH, .SETUP, .SCMTAG, .STYPE, .GETSWR, .CKSWR
.MCALL .STRAP, .EQUAT, .SPOWER, .SETPRI, .STYPOCT, .SREAD, .SRODOCT
.MCALL .SACT11, .KT11, .SSIZE, .PUSH, .POP
```

```
.TITLE MD-11-DZDXJ-A 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-C3), JAN 19, 1977.
*
$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
```

000001
 160000

 001100

 000011
 000012
 000015
 000200
 177776

472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527

177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

000000
000040
000100
000140
000200
000240
000300
000340

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

.EQUIV PS,PSW
STKLM= 177774
PIRQ= 177772
DSWR= 177570
DOISP= 177570
:: STACK LIMIT REGISTER
:: PROGRAM INTERRUPT REQUEST REGISTER
:: HARDWARE SWITCH REGISTER
:: HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0
R1= %1
R2= %2
R3= %3
R4= %4
R5= %5
R6= %6
R7= %7
SP= %6
PC= %7
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: GENERAL REGISTER
:: STACK POINTER
:: PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS
PR0= 0
PR1= 40
PR2= 100
PR3= 140
PR4= 200
PR5= 240
PR6= 300
PR7= 340
:: PRIORITY LEVEL 0
:: PRIORITY LEVEL 1
:: PRIORITY LEVEL 2
:: PRIORITY LEVEL 3
:: PRIORITY LEVEL 4
:: PRIORITY LEVEL 5
:: PRIORITY LEVEL 6
:: PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09,SW9
.EQUIV SW08,SW8
.EQUIV SW07,SW7
.EQUIV SW06,SW6
.EQUIV SW05,SW5
.EQUIV SW04,SW4
.EQUIV SW03,SW3
.EQUIV SW02,SW2
.EQUIV SW01,SW1
.EQUIV SW00,SW0

```

528          100000          .:DATA BIT DEFINITIONS (BIT00 TO BIT15)
529          040000          BIT15= 100000
530          020000          BIT14= 40000
531          010000          BIT13= 20000
532          004000          BIT12= 10000
533          002000          BIT11= 4000
534          001000          BIT10= 2000
535          000400          BIT09= 1000
536          000200          BIT08= 400
537          000100          BIT07= 200
538          000040          BIT06= 100
539          000020          BIT05= 40
540          000010          BIT04= 20
541          000004          BIT03= 10
542          000002          BIT02= 4
543          000001          BIT01= 2
544                                BIT00= 1
545          .EQUIV BIT09,BIT9
546          .EQUIV BIT08,BIT8
547          .EQUIV BIT07,BIT7
548          .EQUIV BIT06,BIT6
549          .EQUIV BIT05,BIT5
550          .EQUIV BIT04,BIT4
551          .EQUIV BIT03,BIT3
552          .EQUIV BIT02,BIT2
553          .EQUIV BIT01,BIT1
554          .EQUIV BIT00,BIT0
555
556          000004          .:BASIC "CPU" TRAP VECTOR ADDRESSES
557          000010          ERRVEC= 4          .: TIME OUT AND OTHER ERRORS
558          000014          RESVEC= 10         .: RESERVED AND ILLEGAL INSTRUCTIONS
559          000014          TBITVEC=14        .: "T" BIT
560          000014          TRTVEC= 14        .: TRACE TRAP
561          000014          BPTVEC= 14        .: BREAKPOINT TRAP (BPT)
562          000020          IOTVEC= 20        .: INPUT/OUTPUT TRAP (IOT) **SCOPE**
563          000024          PWRVEC= 24        .: POWER FAIL
564          000030          EMTVEC= 30        .: EMULATOR TRAP (EMT) **ERROR**
565          000034          TRAPVEC=34        .: "TRAP" TRAP
566          000060          TKVEC= 60         .: TTY KEYBOARD VECTOR
567          000064          TPVEC= 64         .: TTY PRINTER VECTOR
568          000240          PIRQVEC=240      .: PROGRAM INTERRUPT REQUEST VECTOR
569          .SBTTL MEMORY MANAGEMENT DEFINITIONS
570
571          ;*KT11 VECTOR ADDRESS
572
573          000250          MMVEC= 250
574
575          ;*KT11 STATUS REGISTER ADDRESSES
576
577          177572          SR0= 177572
578          177574          SR1= 177574
579          177576          SR2= 177576
580          172516          SR3= 172516
581
582          ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
583

```

584 172300
585 172302
586 172304
587 172306
588 172310
589 172312
590 172314
591 172316

KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

;
;#KERNEL "!" PAGE ADDRESS REGISTERS

592
593
594
595 172340
596 172342
597 172344
598 172346
599 172350
600 172352
601 172354
602 172356

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

603
604 000200
605 000004
606 020000
607 040000
608 002000
609 001000
610 004000

DONE=200
SOSTEN=4
SELO=20000
HLDO=40000
CMD0=2000
SRV0=1000
ADR0=4000
.SBTTL TRAP CATCHER

611
612
613 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

614
615
616
617 000174

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

618 000174 000000
619 000176 000000
620
621 000200 000137 001164

;; *****
;#HOOKS REQUIRED BY ACT11

622
623
624
625
626 000204
627 000046
628 000046 001566
629 000052
630 000052 040000
631 000204
632 001000

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

633
634
635
636
637
638
639

.MACRO \$\$SCHREG A,B
\$REG'A: .WORD 0 ;; CONTAINS ((\$REGAD)+'B)
.NLIST
\$CM1=\$CM1+1
\$CM2=\$CM2+2
.LIST
.ENDM \$\$SCHREG

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 14
ACT11 HOOKS

640
641
642
643
644
645

```
.MACRO $$CMTMP A  
$TMP'A: .WORD 0                ;;USER DEFINED  
.NLIST  
$CM4=$CM4+1  
.LIST  
.ENDM $$CMTMP
```

.SBTTL COMMON TAGS

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

646
647
648
649
650
651
652 001100
653 001100
654 001100 000000
655 001102 000
656 001103 000
657 001104 000000
658 001106 000000
659 001110 000000
660 001112 000000
661 001114 000
662 001115 001
663 001116 000000
664 001120 000000
665 001122 000000
666 001124 000000
667 001126 000000
668 001130 000000
669 001132 000000
670 001134 000
671 001135 000
672 001136 000000
673 001140 177570
674 001142 177570
675 001144 177560
676 001146 177562
677 001150 177564
678 001152 177566
679 001154 000
680 001155 002
681 001156 012
682 001157 000
683 001160 077
684 001161 015
685 001162 000012
686

. =1100
\$CMTAG: .WORD 0
\$PASS: .WORD 0
\$STINM: .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
\$BODADR: .WORD 0
\$GDOAT: .WORD 0
\$BDOAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DOISP
\$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

687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF SITEMB 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:


```

758 001416 000002
759 001420
760
761 001420 032777 040000 177512
762 001426 001356
763 001430 005237 011662
764 001434 023737 011662 011612
765 001442 003750
766 001444 005337 011670
767 001450 001336
768 001452 005737 011750 EOP:
769 001456 001026
770 001460 005737 011746
771 001464 001423
772 001466 004537 005134
773 001472 013162
774 001474 000000
775 001476 004537 005134
776 001502 011744
777 001504 100001
778 001506 004537 005134
779 001512 013220
780 001514 000000
781 001516 004537 005134
782 001522 011746
783 001524 100001
784 001526 012737 000001 011750
785 001534 005737 011646 10$:
786 001540 001402
787 001542 005037 177572
788 001546 004537 005134 15$:
789 001552 012712
790 001554 000000
791 001556 013700 000042
792 001562 001405
793 001564 000005
794 001566 004710 SENDAD:
795 001570 000240
796 001572 000240
797 001574 000240
798 001576 000656 END1:
799

```

```

RTI ; ; POP NEW PC AND PS

BIT #40000, @SWR ; IS LOOP SWITCH ON
BNE TST ; BR IF YES
INC CTESTN ; INCREMENT TO NEXT TEST
CMP CTESTN, HTESTN ; HAVE ALL TEST BEEN RUN
BLE TST ; BR IF NO, DISPATCH TO NEXT ONE
DEC CNT ; DECREMENT ITERATION COUNT
BNE GO ; START OVER IF NOT EXPIRED
TST DEVMS ; HAVE WE PRINTED DEVICE CONFIG YET?
BNE 10$ ; BR IF YES
TST DEVCT ; WERE DEVICES RECOGNIZED?
BEQ 10$ ; BR IF NO, NOTHING TO SAY
JSR R5, PRINT ; PRINT DEV NO. ASCII
.WORD DEVMS
.WORD 0
JSR R5, PRINT ; PRINT DEV NO. OCTAL
.WORD FRSTDV
.WORD 100001
JSR R5, PRINT ; PRINT DEVICE COUNT ASCII
.WORD DEVCMS
.WORD 0
JSR R5, PRINT ; PRINT DEVICE COUNT OCT
.WORD 100001
MOV #1, DEVMS ; SET DEV CONFIG PRINTED SWITCH
TST KT ; IS KT IN USE?
BEQ 15$ ; BR IF NO
CLR @#SR0 ; TURN OFF MEM MGT
JSR R5, PRINT
.WORD BELL
.WORD 0 ; PRINT CONTROL PARAMETER
MOV @#42, R0 ; ARE WE UNDER ACT
BEQ END1 ; BR IF NO, START OVER
RESET
JSR PC, @R0 ; GO TO MONITOR
NOP ; ROOM FOR
NOP ; ACT11
NOP ; STUFF
BR RESTART ; START OVER

```

800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855

001600 005737 011650
001604 001402
001606 000137 002270
001612 005777 176224
001616 001415
001620 013737 011600 011630
001626 013737 011602 011632
001634 013737 011604 011634
001642 013737 011606 011636
001650 000564
001652 022737 001566 000042
001660 001560
001662 005737 011664
001666 001004
001670 004537 005134
001674 012262
001676 000000
001700 004537 002272
001704 000001
001706 122700 000104
001712 001015
001714 013737 011600 011630
001722 013737 011602 011632
001730 013737 011604 011634
001736 013737 011606 011636
001744 000526
001746 122700 000120
001752 001011
001754 005737 011664
001760 001401
001762 000517
001764 004537 005134
001770 012520

```
.SBTTL MODULE 1.0 GET PARAMETERS
MODULE 1.0 GET PARAMETERS

CALLED AS FOLLOWS

JSR PC,GETPRM

THIS MODULE ASSURES THAT ALL PARAMETERS ARE ACQUIRED FOR
OPERATION OF THE DIAGNOSTIC. TO DO THIS IT ESTABLISHES
A DIALOG WITH THE OPERATOR THROUGH THE INPUT PARAM MODULE.
MEMORY IS SIZED BY THE 'SIZE' MODULE, AND ALL PARAMS ARE
CHECKED FOR VALIDITY BY THE CHECKP MODULE.

PC AT ANY TIME CLEARS THE 1ST PASS SWITCH AND EVERYTHING
STARTS OVER. CR IN RESPONSE TO ANY QUESTION CAUSES THE DEFAULT
VALUE TO BE SUBSTITUTED.

THIS MODULE NEVER RETURNS TO CALLER UNLESS ALL PARAMETERS
HAVE BEEN ESTABLISHED WITH GOOD VALUES.

GETPRM: TST FIRSTP ; IS THIS THE FIRST PASS
        BEQ 5$ ; BR IF YES
        JMP ENDPRM ; DON'T GET PARAM IF WE ALREADY HAVE THEM
5$: TST 242 ; ARE WE RUNNING IN AUTO MODE
    BEQ 10$ ; BRANCH IF MACHINE IS ATTENDED
    MOV DTESTN,ATESTN ; SETUP DEFAULTS
    MOV DUBA,AUBA ; SETUP DEFAULTS
    MOV DVECT,AVECT ; SETUP DEFAULTS
    MOV DPRIOR,APRIOR ; SETUP DEFAULTS
    BR 90$ ; GO SEE ABOUT MEMORY SIZING
10$: CMP #SENDAD,242 ; UNDER ACT11?
    BEQ 90$ ; NO OPERATOR INTERACTION
    TST PARMV ; ARE PARAMETERS GOOD
    BNE 15$ ; BRANCH IF YES BECAUSE THIS
    ; ALSO MEANS HEADER ISN'T NEEDED
    JSR RS,PRINT ; CALL THE PRINTER
    ; ADDRESS OF HEADER ASCIZ
    .WORD 0 ; CONTROL WORD, CRLF + DON'T CONVERT
15$: JSR RS,INPRM ; GET D,P,S, OR N FROM OPER.
    ; THIS TELLS INPRM TO GET D,P,S,N
    ; THE CHAR TYPED
    .WORD 1 ; WAS DEFAULT PARAMS REQUESTED?
    ; INPRM WILL RETURN IN RO
    CMPB #'D,RO ; BR IF NO
    BNE 20$ ; DEFAULT SETUP
    MOV DTESTN,ATESTN ; DEFAULT SETUP
    MOV DUBA,AUBA ; DEFAULT SETUP
    MOV DVECT,AVECT ; DEFAULT SETUP
    MOV DPRIOR,APRIOR ; DEFAULT SETUP
    BR 90$ ; GO SEE ABOUT MEMORY SIZING
20$: CMPB #'P,RO ; DOES OPERATOR WANT PREVIOUS
    BNE 30$ ; BR IF NO
    TST PARMV ; ARE PARAMS SET UP
    BEQ 25$ ; BR IF PARAMETERS NOT VALID
    BR 90$ ; USE PREVIOUS PARAMS, NO ACTION NEEDED
25$: JSR RS,PRINT ; TROUBLE, P TYPED AND PARAMS
    .WORD INVM ; NEVER SET UP PROPERLY
```

856	001772	000000			WORD	0		: CONTROL WORD, CRLF + DONT CONVERT
857	001774	000726			BR	10\$: GO TO TRY AGAIN
858	001776	122700	000123	30\$:	CMPB	#'S,RO		: ARE PARMS TO BE SELECTED?
859	002002	001062			BNE	40\$: BR IF NO
860	002004	005037	011664	35\$:	CLR	PARMV		: CLEAR PARM VALID SW
861	002010	004537	002272		JSR	RS,INPRM		: ASK INPUT MODULE TO GET TEST NO:
862	002014	000002			.WORD	2		: TEST NUMBER WILL BE RETURNED IN RO
863								: IN OCTAL
864	002016	010037	011630		MOV	RO,ATESTN		: SAVE TEST NUMBER IN ACTIVE PARM TABLE
865	002022	004537	004766		JSR	RS,CHECKP		: CHECK IF TEST NO OK
866	002026	000002			WORD	6		: PARAMETER INDICATES VERIFY TEST #
867	002030	005700			TST	RO		: RO=0 IF CHECK OK
868	002032	001364			BNE	35\$: STAY HERE UNTIL OPERATOR GETS IT RIGHT
869	002034	004537	002272	36\$:	JSR	RS,INPRM		: ASK INPUT MODULE TO GET BASE ADDR
870	002040	000003			.WORD	3		: PARAMETER INDICATING GET UBA BASE
871	002042	010037	011632		MOV	RO,AUBA		: SAVE UB BASE ADDR IN ACTIVE TABLE
872	002046	004537	004766		JSR	RS,CHECKP		: CHECK FOR VALID ENTRY
873	002052	000003			.WORD	3		: PARM TO INDICATE CHECK UBA
874	002054	005700			TST	RO		: RO=0 IF CHECK OK
875	002056	001366			BNE	36\$: STAY HERE UNTIL HE GETS IT RIGHT
876	002060	004537	002272	37\$:	JSR	RS,INPRM		: ASK INPUT MODULE TO GET VECTOR ADDR
877	002064	000004			.WORD	4		: PARM INDICATING VECTOR NEEDED
878	002066	010037	011634		MOV	RO,AVECT		: SAVE VECTOR IN ACTIVE PARM TABLE
879	002072	004537	004766		JSR	RS,CHECKP		: CHECK FOR VALID VECTOR
880	002076	000004			.WORD	4		: PARM INDICATING CHECK VECTOR
881	002100	005700			TST	RO		: RO=0 IF VECTOR OK
882	002102	001366			BNE	37\$: STAY HERE UNTIL HE GETS IT RIGHT
883	002104	004537	002272	38\$:	JSR	RS,INPRM		: GO GET DX PRIORITY
884	002110	000005			.WORD	5		
885	002112	010037	011636		MOV	RO,APRIOR		: SAVE PRIORITY
886	002116	004537	004766		JSR	RS,CHECKP		: CHECK PRIORITY ENTRY
887	002122	000005			.WORD	5		: PARM INDICATING PRIORITY CHECK
888	002124	005700			TST	RO		
889	002126	001366			BNE	38\$: STAY HERE UNTIL HE GETS IT RIGHT
890	002130	004537	005134		JSR	RS,PRINT		: PRINT SET SWITCHES MESSAGE
891	002134	012674			.WORD	SETSW		: ADDR OF MESSAGE
892	002136	000000			.WORD	0		
893	002140	004537	002532		JSR	RS,INASC		: DUMMY INPUT REQUEST TO WAIT FOR
894	002144	000001			.WORD	1		: OPERATOR TO SET SWITCHES
895	002146	000425			BR	90\$: GO SEE ABOUT MEMORY SIZING
896	002150	122700	000116	40\$:	CMPB	#'N,RO		: WAS AN "N" INPUT
897	002154	001236			BNE	10\$: BR IF NONE OF DPSN WERE INPUT
898	002156	004537	002272	45\$:	JSR	RS,INPRM		: CALL FOR TEST NUMBER TO BE INPUT
899	002162	000002			.WORD	2		: PARM SAYS GET TEST #.
900	002164	010037	011630		MOV	RO,ATESTN		: SAVE STARTING TEST #
901	002170	004537	004766		JSR	RS,CHECKP		: HAVE TEST # CHECKED FOR PROPER
902	002174	000002			.WORD	2		: RANGE
903	002176	005700			TST	RO		: IS TEST # OK?
904	002200	001366			BNE	45\$: STAY HERE UNTIL HE GETS IT RIGHT
905	002202	005737	011664		TST	PARMV		: WE HAVE STARTING TEST #, ARE OTHER PARMS OK?
906	002206	001005			BNE	90\$: BR IF OK
907	002210	004537	005134		JSR	RS,PRINT		: PRINT
908	002214	012536			INVPRM			: POINTER TO "NEED MORE PARMS" MSG
909	002216	000000			.WORD	0		: CONTROL WORD, CRLF + DON'T CONVERT
910	002220	000614			BR	10\$: GO BACK AND DO IT ALL AGAIN
911								

912				
913	002222	004737	004212	
914	002226	012700	000015	
915	002232	012701	011530	
916	002236	013702	011632	
917	002242	010221		
918	002244	062702	000002	
919	002250	005300		
920	002252	001373		
921	002254	012737	000001	011664
922	002262	012737	000001	011650
923	002270	000207		

```

90$: JSR PC SIZE ;CALL MEMORY SIZING MODULE
     MOV #15,R0 ;START OF LOOP TO SET UP DX ACCESS ADDRESSES
     MOV #DX05A,R1 ;GET ADDR OF START OF TABLF.
     MOV AUBA,R2 ;GET DX REG BASE ADDRESS
95$: 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 ;RETURN TO CALLER, END OF MODULE
PC

```

924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979

.SBTTL MODULE 1.1 INPUT PARAMETERS

MODULE 1.1 INPUT PARAMETERS

CALLED AS FOLLOWS

JSR RS,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 CX 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

002272 012501
002274 120127 000001
002300 001016
002302 004537 005134
002306 012556
002310 000000
002312 004537 002532
002316 000001
002320 111100
002322 120027 000015
002326 001100
002330 113700 000120
002334 000475
002336 120127 000002
002342 001014
002344 004537 005134
002350 012570
002354 004537 002532
002360 000002
002362 011100
002364 001061
002366 013700 011600
002372 000456
002374 120127 000003
002400 001014
002402 004537 005134
002406 012613
002410 000000

```
INPRM: MOV (RS)+,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 RS,PRINT ;PRINT
          .WORD PROMPT ;D,P,S,N?
          .WORD 0       ;CONTROL WORD, CRLF + DON'T CONVERT
          JSR RS,INASC  ;READ THE INPUT -
          .WORD 1       ;INASC RETURNS POINTER TO
          MOVB (R1),R0  ;INPUT IN R1
          ;1ST CHAR TO R0
          CMPB R0,#CR  ;WAS IT DEFAULTED?
          BNE 90$      ;NO, ALL DONE
          MOVB '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 RS,PRINT   ;PRINT
     .WORD TESTN    ;ADDR OF STARTING TESTN MSG.
     .WORD 0        ;CONTROL WORD, CRLF + DON'T CONVERT
     JSR RS,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 RS,PRINT   ;PRINT
     .WORD BASEA    ;ASK FOR BASE ADDR
     .WORD 0        ;CONTROL WORD, CRLF + DON'T CONVERT
```

980	002412	004537	002532		JSR	R5, INASC	: GET THE OCTAL
981	002416	000002			.WORD	2	: INPUT
982	002420	011100			MOV	(R1), R0	: GET THE NUMBER INPUT
983	002422	001042			BNE	90\$: BR IF NOT DEFAULTED
984	002424	013700	011602		MOV	DUBA, R0	: SUBSTITUTE THE DEFAULT
985	002430	000437			BR	90\$: DONE
986	002432	120127	000004	30\$:	CMPB	R1, #4	: DOES CALLER WANT VECTOR?
987	002436	001014			BNE	40\$: BR IF NO
988	002440	004537	005134		JSR	R5, PRINT	: PRINT
989	002444	012631			.WORD	VECTA	: ASK FOR VECTOR ADDR
990	002446	000000			.WORD	0	: CONTROL WORD, CRLF + DON'T CONVERT
991	002450	004537	002532		JSR	R5, INASC	: GET THE INPUT
992	002454	000002			.WORD	2	: IN OCTAL
993	002456	011100			MOV	(R1), R0	: GET THE NUMBER INPUT
994	002460	001023			BNE	90\$: BR IF NOT DEFAULTED
995	002462	013700	011604		MOV	DVECT, R0	: SUBSTITUTE THE DEFAULT
996	002466	000420			BR	90\$: DONE
997	002470	120127	000005	40\$:	CMPB	R1, #5	: DOES CALLER WANT PRIORITY?
998	002474	001014			BNE	80\$: BR IF INVALID CALL
999	002476	004537	005134		JSR	R5, PRINT	: PRINT
1000	002502	012651			.WORD	PRI LV	: ASK FOR PRIORITY LEVEL
1001	002504	000000			.WORD	0	: CONTROL WORD, CRLF + DON'T CONVERT
1002	002506	004537	002532		JSR	R5, INASC	: REQUEST INPUT
1003	002512	000002			.WORD	2	: IN OCTAL
1004	002514	011100			MOV	(R1), R0	: GET THE INPUT
1005	002516	001004			BNE	90\$: BR IF OK, NOT DEFAULTED
1006	002520	013700	011606		MOV	DPRIOR, R0	: SUBSTITUTE THE DEFAULT
1007	002524	000401			BR	90\$	
1008							
1009	002526	000000		80\$:	HALT		: HALT ON INVALID CALL TO THIS MOD
1010	002530	000205		90\$:	RTS	R5	: RETURN TO CALLER
1011					.SBTTL	MODULE 1.1.1	: INPUT FROM TTY
1012							

M02

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 24
MODULE 1.1.1 INPUT FROM TTY

1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027 002532 004737 002620
1028 002536 012500
1029 002540 120027 000001
1030 002544 001004
1031 002546 104410
1032 002550 013637 013306
1033 002554 000406
1034
1035 002556 120027 000002
1036 002562 001006
1037 002564 104411
1038 002566 012637 013306
1039 002572 012701 013306
1040 002576 000205
1041 002600 000000
1042
1043
1044
1045
1046
1047
1048 002602 000000
1049 002604 000000
1050 002606 000000
1051 002610 000010
1052 002620 002620
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062 002620 005037 002602
1063 002624 012737 002610 002604
1064 002632 013737 002604 002606
1065 002640 012737 002670 000060
1066 002646 012737 002020 000062
1067 002654 005777 176266
1068 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 \$SREAD MACRO
MOV @(\$P)+,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 (\$P)+,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,\$TKKVEC ;INITIALIZE THE KEYBOARD VECTOR
MOV #200,\$TKKVEC+2 ;"BR" LEVEL 4
TST \$TKB ;CLEAR DONE FLAG
MOV #100,\$TKS ;ENABLE TTY KEYBOARD INTERRUPT

```

1069 002666 000207          RTS      PC          ;;RETURN TO CALLER
1070
1071
1072
1073
1074
1075
1076
1077
1078 002670 117746 176252 $TKSRV: MOVB  $STKB, -(SP)      ;; PICKUP THE CHARACTER
1079 002674 042716 177600      BIC      #177, (SP)      ;; STRIP THE JUNK
1080 002670 021627 000003      CMP      (SP), #3       ;; IS IT A CONTROL C?
1081 002704 001007          BNE      1$             ;; BRANCH IF NO
1082 002706 104401 004010      TYPE    $CNTLC        ;; TYPE A CONTROL-C (1C)
1083 002712 004737 002620      JSR     PC, $TRINT    ;; INIT THE KEYBOARD
1084 002716 005726          TST     (SP)+         ;; CLEAN UP STACK
1085 002720 000137 001164      JMP     START         ;; CONTROL C RESTART
1086 002724 021627 000007      1$:    CMP      (SP), #7       ;; IS IT A CONTROL G?
1087 002730 001004          BNE      2$             ;; BRANCH IF NO
1088 002732 022737 000176 001140      CMP     #SWREG, SWR   ;; IS SOFT-SWR SELECTED?
1089 002740 001500          BEQ     6$             ;; GO TO SWR CHANGE
1090
1091 002742          2$:
1092 002742 022737 000010 002602      CMP     #10, $STKCNT  ;; IS THE QUEUE FULL?
1093 002750 001004          BNE      3$             ;; BRANCH IF NO
1094 002752 104401 004004      TYPE    $BELL        ;; RING THE TTY BELL
1095 002756 005726          TST     (SP)+         ;; CLEAN CHARACTER OFF OF STACK
1096 002760 000451          BR      5$             ;; EXIT
1097 002762 021627 000023      3$:    CMP      (SP), #23     ;; IS IT A CONTROL-S?
1098 002766 001021          BNE      32$           ;; BRANCH IF NO
1099 002770 005077 176150      CLR     $STKS        ;; DIS 3LE TTY KEYBOARD INTERRUPTS
1100 002774 005726          TST     (SP)+         ;; CLEAN CHAR OFF STACK
1101 002776 105777 176142      31$:   TSTB    $STKS        ;; WAIT FOR A CHAR
1102 003002 100375          BPL     31$           ;; LOOP UNTIL ITS THERE
1103 003004 117746 176136      MOVB   $STKB, -(SP)   ;; GET THE CHARACTER
1104 003010 042716 177600      BIC     #177, (SP)    ;; MAKE IT 7-BIT ASCII
1105 003014 022627 000021      CMP     (SP)+, #21    ;; IS IT A CONTROL-Q?
1106 003020 001366          BNE      31$           ;; BRANCH IF NO
1107 003022 012777 000100 176114      MOV     #100, $STKS   ;; REENABLE TTY KEYBOARD INTERRUPTS
1108 003030 000002          RTI                    ;; RETURN
1109 003032 005237 002602      32$:   INC     $TKCNT       ;; COUNT THIS CHARACTER
1110 003036 021627 000140      CMP     (SP), #140    ;; IS IT UPPER CASE?
1111 003042 002405          BLT     4$             ;; BRANCH IF YES
1112 003044 021627 000175      CMP     (SP), #175    ;; IS IT A SPECIAL CHAR?
1113 003050 003002          BGT     4$             ;; BRANCH IF YES
1114 003052 042716 000040      BIC     #40, (SP)     ;; MAKE IT UPPER CASE
1115 003056 112677 177522      4$:    MOVB   (SP)+, $STKQIN ;; AND PUT IT IN QUEUE
1116 003062 005237 002604      INC     $TKQIN       ;; INCR THE POINTER
1117 003066 023727 002604 002620      CMP     $TKQIN, $STKQEND ;; GO OFF THE END?
1118 003074 001003          BNE      5$             ;; BRANCH IF NO
1119 003076 012737 002610 002604      MOV     #STKQSR, $TKQIN ;; RESET THE POINTER
1120 003104 000002      5$:    RTI                    ;; RETURN
1121
1122
1123
1124

```

```

*****
;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL

```

```

1125 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
1126 ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
1127 003106 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;: IS THE SOFT-SWR SELECTED
1128 003114 001124          BNE      15$          ;: EXIT IF NOT
1129 003116 105777 176022      TSTB   2$TKS          ;: IS A CHAR WAITING?
1130 003122 100121          BPL     15$          ;: IF NOT, EXIT
1131 003124 117746 176016      MOVB   2$TKB,-(SP)    ;: YES
1132 003130 042716 177600      BIC   #1C177,(SP)   ;: MAKE IT 7-BIT ASCII
1133 003134 021627 000007      CMP   (SP),#7       ;: IS IT A CONTROL-G?
1134 003140 001300          BNE    2$           ;: IF NOT, PUT IT IN THE TTY QUEUE
1135 ;: AND EXIT
1136
1137 ;:*****
1138 ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
1139 ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
1140 ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
1141 003142 123727 001134 000001 6$:  CMPB   $AUTOB,#1    ;: ARE WE RUNNING IN AUTO-MODE?
1142 003150 001674          BEQ    2$           ;: BRANCH IF YES
1143 003152 005726          TST   (SP)+         ;: CLEAR CONTROL-G OFF STACK
1144 003154 004737 002620      JSR   PC,STKINT     ;: FLUSH THE TTY INPUT QUEUE
1145 003160 005077 175760      CLR   2$TKS        ;: DISABLE TTY KEYBOARD INTERRUPTS
1146 003164 112737 000001 001135      MOVB   #1,$INTAG    ;: SET INTERRUPT MODE INDICATOR
1147
1148 003172 104401 004022      TYPE   , $CNTLG     ;: ECHO THE CONTROL-G (↑G)
1149 003176 104401 004027      $GTSWR: TYPE   $MSWR    ;: TYPE CURRENT CONTENTS
1150 003202 013746 000176      MOV   SWREG,-(SP)  ;: SAVE SWREG FOR TYPEOUT
1151 003206 104402          TYPOC          ;: GO TYPE--OCTAL ASCII(ALL DIGITS)
1152 003210 104401 004040      TYPE   , $MNEW     ;: PROMPT FOR NEW SWR
1153 003214 005046          CLR   -(SP)        ;: CLEAR COUNTER
1154 003216 005046          CLR   -(SP)        ;: THE NEW SWR
1155 003220 105777 175720      7$:  TSTB   2$TKS        ;: CHAR THERE?
1156 003224 100375          BPL    7$          ;: IF NOT TRY AGAIN
1157
1158 003236 117746 175714      MOVB   2$TKB,-(SP) ;: PICK UP CHAR
1159 003232 042716 177600      BIC   #1C177,(SP) ;: MAKE IT 7-BIT ASCII
1160
1161 003236 021627 000003          CMP   (SP),#3      ;: IS IT A CONTROL-C?
1162 003242 001015          BNE   9$           ;: BRANCH IF NOT
1163 003244 104401 004010      TYPE   , $CNTLC    ;: YES, ECHO CONTROL-C (↑C)
1164 003250 062706 000006      ADD   #6,SP        ;: CLEAN UP STACK
1165 003254 123727 001135 000001      CMPB   $INTAG,#1   ;: REENABLE TTY KEYBOARD INTERRUPTS?
1166 003262 001003          BNE   8$           ;: BRANCH IF NO
1167 003264 012777 000100 175652      MOV   #100,2$TKS  ;: ALLOW TTY KEYBOARD INTERRUPTS
1168 003272 000137 001164      8$:  JMP    START      ;: CONTROL-C RESTART
1169
1170
1171 003276 021627 000025      9$:  CMP   (SP),#25    ;: IS IT A CONTROL-U?
1172 003302 001005          BNE   10$          ;: BRANCH IF NOT
1173 003304 104401 004015      TYPE   , $CNTLU    ;: YES, ECHO CONTROL-U (↑U)
1174 003310 062706 000006      20$: ADD   #6,SP      ;: IGNORE PREVIOUS INPUT
1175 003314 000737          BR    19$          ;: LET'S TRY IT AGAIN
1176
1177
1178 003316 021627 000015      10$: CMP   (SP),#15    ;: IS IT A (CR)?
1179 003322 001022          BNE   16$          ;: BRANCH IF NO
1180 003324 005766 000004          TST   4(SP)        ;: YES, IS IT THE FIRST CHAR?

```

```

1181 003330 001403          BEQ      11$          ; BRANCH IF YES
1182 003332 016677 000002 175600  MOV     2(SP),@SWR   ; SAVE NEW SWR
1183 003340 062706 000006          11$:  ADD     @6,SP        ; CLEAR UP STACK
1184 003344 104401 001161          14$:  TYPE   $CRLF      ; ECHO <CR> AND <LF>
1185 003350 123727 001135 000001  CMPB   $INTAG,#1    ; RE-ENABLE TTY KBD INTERRUPTS?
1186 003356 001003          BNE     15$          ; BRANCH IF NOT
1187 003360 012777 000100 175556  MOV     @100,@STKS  ; RE-ENABLE TTY KBD INTERRUPTS
1188 003366 000002          15$:  RTI                    ; RETURN
1189 003370 001737 005356          16$:  JSR     PC,$TYPEC   ; ECHO CHAR
1190 003374 021627 000060          CMP     (SP),@60    ; CHAR < 0?
1191 003400 002420          BLT     18$          ; BRANCH IF YES
1192 003402 021627 000067          CMP     (SP),@67    ; CHAR > 7?
1193 003406 003015          BGT     18$          ; BRANCH IF YES
1194 003410 042726 000060          BIC     @60,(SP)+   ; STRIP-OFF ASCII
1195 003414 005766 000002          TST    2(SP)        ; IS THIS THE FIRST CHAR
1196 003420 001403          BEQ     17$          ; BRANCH IF YES
1197 003422 006316          ASL    (SP)         ; NO, SHIFT PRESENT
1198 003424 006316          ASL    (SP)         ; CHAR OVER TO MAKE
1199 003426 006316          ASL    (SP)         ; ROOM FOR NEW ONE.
1200 003430 005266 000002          17$:  INC     2(SP)        ; KEEP COUNT OF CHAR
1201 003434 056616 177776          BIS    -2(SP),(SP) ; SET IN NEW CHAR
1202 003440 000667          BR      7$          ; GET THE NEXT ONE
1203 003442 104401 001160          18$:  TYPE   $QUES      ; TYPE ?<CR><LF>
1204 003446 000720          BR      20$        ; SIMULATE CONTROL-U
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216 003450 011646          $RDCR: MOV    (SP),-(SP) ; PUSH DOWN THE PC AND
1217 003452 016666 000004 000002  MOV    4(SP),2(SP)  ; THE PS
1218 003460 005066 000004          CLR    4(SP)        ; GET READY FOR A CHARACTER
1219 003464 005046          CLR    -(SP)        ; PUT NEW PS ON STACK
1220 003466 012746 003474          MOV    @64$,-(SP)  ; PUT NEW PC ON STACK
1221 003472 000002          RTI                    ; POP NEW PC AND PS
1222 003474
1223 003474 005737 002602          64$:  TST    $STKCNT     ; ; WAIT ON A CHARACTER
1224 003500 001775          1$:   BEQ     1$          ;
1225 003502 005337 002602          DEC    $STKCNT     ; DECREMENT THE COUNTER
1226 003506 117766 177074 000004          MOVB   @STKQOUT,4(SP) ; GET ONE CHARACTER
1227 003514 005237 002606          INC    $STKQOUT    ; UPDATE THE POINTER
1228 003520 023727 002606 002620  CMP    $STKQOUT,@STKQEND ; DID IT GO OFF OF THE END?
1229 003526 001003          BNE    2$          ; BRANCH IF NO
1230 003530 012737 002610 002606  MOV    @STKQST,$STKQOUT ; RESET THE POINTER
1231 003536 000002          2$:   RTI                    ; RETURN
1232
1233
1234
1235
1236
; *****
; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
; CALL:
; * RDCR
; * RETURN HERE
; * GET A CHARACTER FROM THE QUEUE
; * CHARACTER IS ON THE STACK
; * WITH PARITY BIT STRIPPED OFF
; *****
; *****
; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
; CALL:
; * RDLIN
; * RETURN HERE
; * INPUT A STRING FROM THE TTY
; * ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK

```

```

1237 ;* ; ; TERMINATOR WILL BE A BYTE OF ALL 0'S
1238
1239 003540 010346 $RDLIN: MOV R3, -(SP) ; ; SAVE R3
1240 003542 005046 CLR -(SP) ; ; CLEAR THE RUBOUT KEY
1241 003544 012703 003774 1$: MOV $TTYIN, R3 ; ; GET ADDRESS
1242 003550 022703 004004 2$: CMP $TTYIN+10, R3 ; ; BUFFER FULL?
1243 003554 101456 BLOS 4$ ; ; BR IF YES
1244 003556 104407 RDCMR ; ; GO READ ONE CHARACTER FROM THE TTY
1245 003560 112613 MOVB (SP)+, (R3) ; ; GET CHARACTER
1246 003562 122713 000177 10$: CMPB #177, (R3) ; ; IS IT A RUBOUT
1247 003566 001022 BNE 5$ ; ; BR IF NO
1248 003570 005716 TST (SP) ; ; IS THIS THE FIRST RUBOUT?
1249 003572 001007 BNE 6$ ; ; BR IF NO
1250 003574 112737 000134 003772 MOVB #' \, 9$ ; ; TYPE A BACK SLASH
1251 003602 104401 003772 TYPE 9$
1252 003606 012716 177777 MOV #-1, (SP) ; ; SET THE RUBOUT KEY
1253 003612 005303 6$: DEC R3 ; ; BACKUP BY ONE
1254 003614 020327 003774 CMP R3, $TTYIN ; ; STACK EMPTY?
1255 003620 103434 BLO 4$ ; ; BR IF YES
1256 003622 111337 003772 MOVB (R3), 9$ ; ; SETUP TO TYPEOUT THE DELETED CHAR.
1257 003626 104401 003772 TYPE 9$ ; ; GO TYPE
1258 003632 000746 BR 2$ ; ; GO READ ANOTHER CHAR.
1259 003634 005716 5$: TST (SP) ; ; RUBOUT KEY SET?
1260 003636 001406 BEQ 7$ ; ; BR IF NO
1261 003640 112737 000134 003772 MOVB #' \, 9$ ; ; TYPE A BACK SLASH
1262 003646 104401 003772 TYPE 9$
1263 003652 005016 CLR (SP) ; ; CLEAR THE RUBOUT KEY
1264 003654 122713 000025 7$: CMPB #25, (R3) ; ; IS CHARACTER A CTRL U?
1265 003660 001003 BNE 8$ ; ; BR IF NO
1266 003662 104401 004015 TYPE $CNTLU ; ; TYPE A CONTROL "U"
1267 003666 000726 BR 1$ ; ; GO START OVER
1268 003670 122713 000022 8$: CMPB #22, (R3) ; ; IS CHARACTER A "r"?
1269 003674 001011 BNE 3$ ; ; BRANCH IF NO
1270 003676 105013 CLRB (R3) ; ; CLEAR THE CHARACTER
1271 003700 104401 001161 TYPE $CRLF ; ; TYPE A "CR" & "LF"
1272 003704 104401 003774 TYPE $TTYIN ; ; TYPE THE INPUT STRING
1273 003710 000717 BR 2$ ; ; GO PICKUP ANOTHER CHARACTER
1274 003712 104401 001160 4$: TYPE $QUES ; ; TYPE A '?'
1275 003716 000712 BR 1$ ; ; CLEAR THE BUFFER AND LOOP
1276 003720 111337 003772 3$: MOVB (R3), 9$ ; ; ECHO THE CHARACTER
1277 003724 104401 003772 TYPE 9$
1278 003730 122723 000015 CMPB #15, (R3)+ ; ; CHECK FOR RETURN
1279 003734 001305 BNE 2$ ; ; LOOP IF NOT RETURN
1280 003736 105063 177777 CLRB -1(R3) ; ; CLEAR RETURN (THE 15)
1281 003742 104401 001162 TYPE $LF ; ; TYPE A LINE FEED
1282 003746 005726 TST (SP)+ ; ; CLEAN RUBOUT KEY FROM THE STACK
1283 003750 012603 MOV (SP)+, R3 ; ; RESTORE R3
1284 003752 011646 MOV (SP), -(SP) ; ; ADJUST THE STACK AND PUT ADDRESS OF THE
1285 003754 016666 000004 000002 MOV 4(SP), 2(SP) ; ; FIRST ASCII CHARACTER ON IT
1286 003762 012766 003774 000004 MOV $TTYIN, 4(SP)
1287 003770 000002 RTI ; ; RETURN
1288 003772 000 9$: .BYTE 0 ; ; STORAGE FOR ASCII CHAR. TO TYPE
1289 003773 000 .BYTE 0 ; ; TERMINATOR
1290 003774 000010 $TTYIN: .BLKB 10 ; ; RESERVE 10 BYTES FOR TTY INPUT
1291 004004 177607 000377 $BELL: .ASCIZ <207><377><377> ; ; CODE FOR BELL
1292 004010 041536 005015 000 $CNTLC: .ASCIZ /r/ <15> <12> ; ; CONTROL "C"

```

E03

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 29
TTY INPUT ROUTINE

1293	004015	136	006525	000012	\$CNTLU: .ASCIZ /!U/<15><12>	::CONTROL "U"
1294	004022	043536	005015	000	\$CNTLG: .ASCIZ /!G/<15><12>	::CONTROL "G"
1295	004027	015	051412	051127	\$MSWR: .ASCIZ <15><12>/SWR = /	
1296	004034	036440	000040			
1297	004040	020040	042516	020127	\$MNEW: .ASCIZ / NEW = /	
1298	004046	020075	000			
1299		004052			.EVEN	
1300					.SBTTL READ AN OCTAL NUMBER FROM THE TTY	
1301						
1302					*****	
1303					*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND	
1304					*CHANGE IT TO BINARY.	
1305					*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL	
1306					*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED	
1307					*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST	
1308					*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.	
1309					*CALL:	
1310					* RDOCT	:::READ AN OCTAL NUMBER
1311					* RETURN HERE	:::LOW ORDER BITS ARE ON TOP OF THE STACK
1312					* *	:::HIGH ORDER BITS ARE IN \$HI OCT
1313					* *	
1314	004052	011646			\$RDOCT: MOV (SP), -(SP)	:::PROVIDE SPACE FOR THE
1315	004054	016666	000004	000002	MOV 4(SP), 2(SP)	:::INPUT NUMBER
1316	004062	010046			MOV R0, -(SP)	:::PUSH R0 ON STACK
1317	004064	012146			MOV R1, -(SP)	:::PUSH R1 ON STACK
1318	004066	010246			MOV R2, -(SP)	:::PUSH R2 ON STACK
1319	004070	104410			1\$: ROLIN	:::READ AN ASCII LINE
1320	004072	012600			MOV (SP)+, R0	:::GET ADDRESS OF 1ST CHARACTER
1321	004074	010037	004200		MOV R0, 5\$:::AND SAVE IT
1322	004100	005001			CLR R1	:::CLEAR DATA WORD
1323	004102	005002			CLR R2	
1324	004104	112046			2\$: MOVB (R0)+, -(SP)	:::PICKUP THIS CHARACTER
1325	004106	001420			BEQ 3\$:::IF ZERO GET OUT
1326	004110	122716	000060		CMPB #'0, (SP)	:::MAKE SURE THIS CHARACTER
1327	004114	003026			BGT 4\$:::IS AN OCTAL DIGIT
1328	004116	122716	000067		CMPB #'7, (SP)	
1329	004122	002423			BLT 4\$	
1330	004124	006301			ASL R1	:::*2
1331	004126	006102			ROL R2	
1332	004130	006301			ASL R1	:::*4
1333	004132	006102			ROL R2	
1334	004134	006301			ASL R1	:::*8
1335	004136	006102			ROL R2	
1336	004140	042716	177770		BIC #'C7, (SP)	:::STRIP THE ASCII JUNK
1337	004144	062601			ADD (SP)+, R1	:::ADD IN THIS DIGIT
1338	004146	000756			BR 2\$:::LOOP
1339	004150	005726			3\$: TST (SP)+	:::CLEAN TERMINATOR FROM STACK
1340	004152	010166	000012		MOV R1, 12(SP)	:::SAVE THE RESULT
1341	004156	010237	004210		MOV R2, \$HI OCT	
1342	004162	012602			MOV (SP)+, R2	:::POP STACK INTO R2
1343	004164	012601			MOV (SP)+, R1	:::POP STACK INTO R1
1344	004166	012600			MOV (SP)+, R0	:::POP STACK INTO R0
1345	004170	000002			RTI	:::RETURN
1346	004172	005726			4\$: TST (SP)+	:::CLEAN PARTIAL FROM STACK
1347	004174	105C10			CLRB (R0)	:::SET A TERMINATOR
1348	004176	104401			TYPE	:::TYPE UP THRU THE BAD CHAR.

F03

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 30
READ AN OCTAL NUMBER FROM THE TTY

1349	004200	000000	
1350	004202	104401	001160
1351	004206	000730	
1352	004210	000000	
1353			
1354			

```

SS:      WORD      0
        TYPE      $QUES
        BR        1$
$HI OCT: .WORD      0
        .SBTTL   MODULE 1.2  SIZE MEMORY
        : " ? " " CR " & " LF "
        : TRY AGAIN
        : HIGH ORDER BITS GO HERE

```

1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366 004212 005037 004516
1367 004216 004737 004460
1368 004222 005037 011646
1369 004226 013737 004762 011642
1370 004234 013737 004762 011640
1371 004242 023727 004762 017776
1372 004250 103415
1373 004252 162737 000276 011640
1374 004260 005737 000042
1375 004264 001407
1376 004266 023737 000042 000046
1377 004274 001403
1378 004276 162737 005500 011640
1379 004304 005037 011644
1380 004310 005037 011700
1381 004314 037727 174620 002000
1382 004322 01055
1383 004324 012737 000200 004516
1384 004332 004737 004460
1385 004336 005737 004516
1386 004342 100045
1387 004344 012737 000001 011646
1388 004352 013737 004764 011644
1389 004360 062737 000040 011644
1390 004366 013737 004764 011702
1391 004374 006337 011702
1392 004400 006337 011702
1393 004404 006337 011702
1394 004410 006337 011702
1395 004414 006337 011702
1396 004420 006337 011702
1397 004424 052737 003776 011702
1398 004432 013737 004764 011700
1399 004440 042737 171777 011700
1400 004446 000337 011700
1401 004452 006337 011700
1402 004456 000207
1403
1404
1405
1406
1407
1408
1409
1410

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,MMAX ;SAVE END OF CORE WITH KT OFF
MOV \$LSTAD,MSIZE ;GET END OF CORE TO WORK WITH
CMP \$LSTAD,#17776 ;LESS THAN BK?
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
CLR EASIZE ;CLEAN UP EXTENDED ADDR
CLR EABITS
BIT #SWR,#SW10 ;CHECK IF MM TO BE USED
BNE 30\$;BR IF NO
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
MOV \$LSTBK,EASIZE
MOV #40,EASIZE ;INCR TO FIRST INVALID SETTING
MOV \$LSTBK,PHYMAX ;START GEN OF MAX PHYSICAL
ASL PHYMAX
ASL PHYMAX
ASL PHYMAX
ASL PHYMAX
ASL PHYMAX
ASL PHYMAX
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:
; JSR PC,\$SIZE
; RETURN
; \$LSTAD WILL CONTAIN:

H03

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 32
ROUTINE TO SIZE MEMORY

1411
1412
1413
1414
1415
1416
1417
1418
1419
1420 004460 010046
1421 004462 010146
1422 004464 010246
1423 004466 010346
1424 004470 013746 000004
1425 004474 013746 000006
1426 004500 010600
1427
1428 004502 104400
1429 004504 012637 000006
1430 004510 012701 003776
1431 004514 105727
1432 004516 000200
1433 004520 100062
1434 004522 012737 004660 000004
1435 004530 005737 177572
1436 004534 052737 100000 004516
1437 004542 005046
1438 004544 012702 172340
1439 004550 012703 000010
1440 004554 012762 077406 177740 1S:
1441 004562 011622
1442 004564 062716 000200
1443 004570 077307
1444 004572 012742 177600
1445 004576 005042
1446 004600 012737 004616 000004
1447 004606 012737 000020 172516
1448 004614 000401
1449 004616 022626
1450 004620 005237 177572
1451 004624 012737 004650 000004
1452 004632 005737 143776
1453 004636 062712 000040
1454 004642 023712 172356
1455 004646 101371
1456 004650 011202
1457 004652 005037 177572
1458 004656 000421
1459 004660 042737 100000 004516
1460 004666 012737 004716 000004
1461 004674 005002
1462 004676 062701 004000
1463 004702 062702 000040
1464 004706 005711
1465 004710 022701 177776
1466 004714 001370

```

;* WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
;* WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
;*SLSTBK WILL CONTAIN THE LAST BANK AS A SAF
;*SKT11 IS THE MEMORY MANAGEMENT KEY
;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
;* MUST BE SETUP BEFORE THE CALL
;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
;* DETERMINED BY ROUTINE

SSIZE: MOV R0,-(SP) ;;SAVE R0 ON THE STACK
MOV R1,-(SP) ;;SAVE R1 ON THE STACK
MOV R2,-(SP) ;;SAVE R2 ON THE STACK
MOV R3,-(SP) ;;SAVE R3 ON THE STACK
MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
MOV @#ERRVEC+2,-(SP)
MOV SP,R0 ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP ;;PUSH OLD PSW AND PC ON STACK
MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
MOV #3776,R1 ;;SETUP ADDRESS
TSTB (PC)+ ;;USE MEMORY MANAGEMENT?
SKT11: .WORD 200 ;;SET TO USE MEMORY MANAGEMENT
BPL SCORE ;;BR IF NO
MOV #SKTNEX,@#ERRVEC ;;SET FOR TIMEOUT
TST @#SR0 ;;KT11 ARE YOU THERE?
BIS #100000,SKT11 ;;YES--SET KT11 KEY
CLR -(SP) ;;INITIALIZE FOR "PAR" LOADING
MOV #KIPAR0,R2 ;;ADDRESS OF FIRST "PAR"
MOV #108,R3 ;;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
1S: MOV #77406,-40(R2) ;;POR = 4K UP, READ/WRITE
MOV (SP),(R2)+ ;;LOAD "PAR"
ADD #200,(SP) ;;UPDATE FOR NEXT "PAR"
SOB R3,1S ;;LOOP UNTIL ALL EIGHT ARE LOADED
MOV #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
CLR -(R2) ;;SETUP KIPAR6 FOR TESTING
MOV #25,@#ERRVEC ;;CATCH TIMEOUT IF NO SR3
MOV #20,@#SR3 ;;ENABLE 22 BIT MODE
BR 3S ;;THIS POP-11 HAS A SR3 REGISTER
2S: CMP (SP)+,(SP)+ ;;CLEAN OFF THE STACK--NO SR3
3S: INC @#SR0 ;;TURN ON MEMORY MANAGEMENT
MOV #SKTOUT,@#ERRVEC ;;SET FOR TIME OUT
4S: TST @#143776 ;;TRAP ON NON-EX-MEM
ADD #40,(R2) ;;MAKE A 1K STEP
CMP @#KIPAR7,(R2) ;;LAST ONE?
BHI 4S ;;NO--TRY IT
SKTOUT: MOV (R2),R2 ;;GET LAST BANK+1
CLR @#SR0 ;;TURN OFF MEMORY MANAGEMENT
BR SSIZEX
SKTNEX: BIC #100000,SKT11 ;;KT11 NON-EXISTENT
SCORE: MOV #SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
CLR R2 ;;SET UP BANK
1S: ADD #4000,R1 ;;INCREMENT BY 1K
ADD #40,R2 ;;1K STEP
TST (R1) ;;TRAP ON TIME OUT
CMP #177776,R1 ;;LAST ONE
BNE 1S ;;NO--TRY AGAIN
```

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 33
ROUTINE TO SIZE MEMORY

1467	004716	162701	004000
1468	004722	162702	000040
1469	004728	010006	
1470	004730	012637	000006
1471	004734	012637	000004
1472	004740	010137	004762
1473	004744	010237	004764
1474	004750	012603	
1475	004752	012602	
1476	004754	012601	
1477	004756	012600	
1478	004760	000207	
1479	004762	000000	
1480	004764	000000	
1481			
1482			

```

$CROUT: SUB #4000,R1
$SIZEX: SUB #40,R2          ;; DROP BACK
        MOV RO,SP          ;; RESTORE THE STACK
        MOV (SP)+,2#ERRVEC+2 ;; RESTORE ERROR VECTOR
        MOV (SP)+,2#ERRVEC
        MOV R1,$LSTAD      ;; LAST ADDRESS
        MOV R2,$LSTBK      ;; LAST BANK
        MOV (SP)+,R3       ;; RESTORE R3
        MOV (SP)+,R2       ;; RESTORE R2
        MOV (SP)+,R1       ;; RESTORE R1
        MOV (SP)+,R0       ;; RESTORE R0
        RTS PC
$LSTAD: .WORD 0          ;; CONTAINS THE LAST ADDRESS
$LSTBK: .WORD 0          ;; CONTAINS THE LAST BANK
.SBTTL  MODULE 1.3 CHECK PARAMETERS

```

```

1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504 004766 012501
1505 004770 120127 000002
1506 004774 001007
1507 004776 020037 011610
1508 005002 002443
1509 005004 020037 011612
1510 005010 003040
1511 005012 000446
1512 005014 120127 000003
1513 005020 001007
1514 005022 020037 011614
1515 005026 002431
1516 005030 020037 011616
1517 005034 003026
1518 005036 000434
1519 005040 120127 000004
1520 005044 001007
1521 005046 020037 011620
1522 005052 002417
1523 005054 020037 011622
1524 005060 003014
1525 005062 000422
1526 005064 120127 000005
1527 005070 001007
1528 005072 020037 011624
1529 005076 002405
1530 005100 020037 011626
1531 005104 003002
1532 005106 000410
1533 005110 000000
1534 005112 004537 005134
1535 005116 012724
1536 005120 000000
1537 005122 012700 177777
1538 005126 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 R0,LTESTN ;CHECK LOW LIMIT
BLT 90$ ;BR IF ERROR
CMP R0,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 R0,LUBA ;CHECK LOW LIMIT
BLT 90$ ;BR IF BAD
CMP R0,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 R0,LVEC ;CHECK THE LOW LIMIT
BLT 90$ ;BR IF BAD
CMP R0,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 R0,LPRIOR ;CHECK LOW LIMIT
BLT 90$ ;BR IF BAD
CMP R0,HPFIOR ;CHECK HIGH LIMIT
BGT 90$ ;BR IF BAD
BR 80$ ;OK, RETURN
40$: HALT ;ERROR, BAD MODULE CALL PARM
90$: JSR R5,PRINT ;PRINT PARM LIMIT ERROR
.WORD BADP ;ADDR OF MESSAGE
.WORD 0 ;CONTROL WORD, CRLF + DON'T CONVERT
MOV #-1,R0 ;SET ERROR RETURN CODE
BR 95$ ;RETURN
    
```

K03

MD-11-DZOXJ-A DX11-B ADDRESSING TEST
DZOXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 35
MODULE 1.3 CHECK PARAMETERS

1539 005130 005000
1540 005132 000205
1541
1542

805: CLR R0 ;SET GOOD RETURN CODE
955: RTS R5 ;RETURN TO CALLER.

.SBTTL MODULE 1.3.1 WRITE TTY

```

1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558 005134
1559 005134 010046
1560 005136 010146
1561 005140 012500
1562 005142 012501
1563 005144 032701 000001
1564 005150 001002
1565 005152 104401
1566 005154 013301
1567 005156 005701
1568 005160 100003
1569 005162 011046
1570 005164 104402
1571 005166 000404
1572 005170 010037 005176
1573 005174 104401
1574 005176 000000
1575 005200
1576 005200 012601
1577 005202 012600
1578 005204 000205
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597 005206 105737 001157
1598 005212 100002

```

```

MODULE 1.3.1 WRITE TTY

THIS MODULE HANDLES ALL OUTPUT TO THE OPERATOR

CALLED AS FOLLOWS

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

```

```

1599 005214 000000          HALT           ;: HALT HERE IF NO TERMINAL
1600 005216 000407          BR           3$          ;: LEAVE
1601 005220 010046          1$: MOV        RO, -(SP) ;: SAVE RO
1602 005222 017600 000002    MOV        22(SP), RO ;: GET ADDRESS OF ASCIZ STRING
1603 005226 112046          2$: MOVB      (RO)+, -(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
1604 005230 001005          BNE        4$          ;: BR IF IT ISN'T THE TERMINATOR
1605 005232 005726          TST       (SP)+       ;: IF TERMINATOR POP IT OFF THE STACK
1606 005234 012600          60$: MOV      (SP)+, RO ;: RESTORE RO
1607 005236 062716 000002    3$: ADD      #2, (SP)   ;: ADJUST RETURN PC
1608 005242 000002          RTI                    ;: RETURN
1609 005244 122716 000011    4$: CMPB     #HT, (SP)  ;: BRANCH IF <HT>
1610 005250 001430          BEQ       8$          ;: BRANCH IF NOT <CR LF>
1611 005252 122716 000200    CMPB     #CR LF, (SP) ;: BRANCH IF NOT <CR LF>
1612 005256 001006          BNE        5$          ;:
1613 005260 005726          TST       (SP)+       ;: POP <CR><LF> EQUIV
1614 005262 104401          TYPE                    ;: TYPE A CR AND LF
1615 005264 001161          SCRLF                    ;:
1616 005266 105037 005422    CLRB     $CHARCNT     ;: CLEAR CHARACTER COUNT
1617 005272 000755          BR        2$          ;: GET NEXT CHARACTER
1618 005274 004737 005356    5$: JSR      PC, $TYPEC ;: GO TYPE THIS CHARACTER
1619 005300 123726 001156    6$: CMPB     $FILLC, (SP)+ ;: IS IT TIME FOR FILLER CHARS.?
1620 005304 001350          BNE        2$          ;: IF NO GO GET NEXT CHAR.
1621 005306 013746 001154    MOV      $NULL, -(SP) ;: GET # OF FILLER CHARS. NEEDED
1622                                     AND THE NULL CHAR.
1623 005312 105366 000001    7$: DECB     1(SP)     ;: DOES A NULL NEED TO BE TYPED?
1624 005316 003770          BLT       6$          ;: BR IF NO--GO POP THE NULL OFF OF STACK
1625 005320 004737 005356    JSR      PC, $TYPEC ;: GO TYPE A NULL
1626 005324 105337 005422    DECB     $CHARCNT     ;: DO NOT COUNT AS A COUNT
1627 005330 000770          BR        7$          ;: LOOP
1628
1629                                     ;: HORIZONTAL TAB PROCESSOR
1630
1631 005332 112716 000040    8$: MOVB     #' , (SP) ;: REPLACE TAB WITH SPACE
1632 005336 004737 005356    9$: JSR      PC, $TYPEC ;: TYPE A SPACE
1633 005342 132737 000007 005422    BITB     #' , $CHARCNT ;: BRANCH IF NOT AT
1634 005350 001372          BNE        9$          ;: TAB STOP
1635 005352 005726          TST       (SP)+       ;: POP SPACE OFF STACK
1636 005354 000724          BR        2$          ;: GET NEXT CHARACTER
1637 005356 105777 173566    $TYPEC: TSTB     $STPS ;: WAIT UNTIL PRINTER IS READY
1638 005362 100375          BPL       $TYPEC
1639 005364 116677 000002 173560    MOVB     2(SP), $STPB ;: LOAD CHAR TO BE TYPED INTO DATA REG.
1640 005372 122766 000015 000002    CMPB     #CR, 2(SP)   ;: IS CHARACTER A CARRIAGE RETURN?
1641 005400 001003          BNE        1$          ;: BRANCH IF NO
1642 005402 105037 005422    CLRB     $CHARCNT     ;: YES--CLEAR CHARACTER COUNT
1643 005406 000406          BR        $TYPEX
1644 005410 122766 000012 000002    1$: CMPB     #LF, 2(SP) ;: IS CHARACTER A LINE FEED?
1645 005416 001402          BEQ       $TYPEX     ;: BRANCH IF YES
1646 005420 105227          INCB     (PC)+       ;: COUNT THE CHARACTER
1647 005422 000000          $CHARCNT: .WORD     0 ;: CHARACTER COUNT STORAGE
1648 005424 000207          $TYPEX: RTS        PC
1649
1650                                     ;: TYPEC TRAP CALL COMES TO $TYPOC TAG BELOW
1651                                     ;: $BITL BINARY TO OCTAL (ASCII) AND TYPE
1652
1653                                     ;: *****
1654                                     ;: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT

```

N03

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 38
BINARY TO OCTAL (ASCII) AND TYPE

```

1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676 005426 017646 000000
1677 005432 116637 000001 005651
1678 005440 112637 005653
1679 005444 062716 000002
1680 005450 000406
1681 005452 112737 000001 005651
1682 0 460 112737 000006 005653
1683 0 466 112737 000005 005650
1684 0 474 010346
1685 0 476 010446
1686 0 480 010546
1687 0 482 113704 005653
1688 0 484 005404
1689 0 486 062704 000006
1690 0 488 110437 005652
1691 0 490 113704 005651
1692 0 492 016475 000012
1693 0 494 016473
1694 0 496 006105 18:
1695 0 498 010404
1696 0 500 006105 28:
1697 005540 006105
1698 005542 006105
1699 005544 005503
1700 005546 006103 38:
1701 005550 105337 005652
1702 005554 100016
1703 005556 042703 177770
1704 005558 001002
1705 005560 005704
1706 005562 001403
1707 005564 005204 48:
1708 005566 062703 000060
1709 005568 062703 000040
1710 005570 110337 005648
    
```

```

*OCTAL (ASCII) NUMBER AND TYPE IT
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOS   ;; CALL FOR TYPEOUT
*   .BYTE  N              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;; M=1 OR 0
*                           ;; 1=TYPE LEADING ZEROS
*                           ;; 0=SUPPRESS LEADING ZEROS
*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPON   ;; CALL FOR TYPEOUT
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOC   ;; CALL FOR TYPEOUT
*STYPOS: MOV     2(SP),-(SP) ;; PICKUP THE MODE
*         MOVVB 1(SP),SOFILL ;; LOAD ZERO FILL SWITCH
*         MOVVB (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
*         ADD   #2,(SP)      ;; ADJUST RETURN ADDRESS
*         BR    STYPON
*STYPOC: MOVVB #1,SOFILL    ;; SET THE ZERO FILL SWITCH
*         MOVVB #6,SOMODE+1 ;; SET FOR SIX(6) DIGITS
*STYPON: MOVVB #5,SOCNT     ;; SET THE ITERATION COUNT
*         MOV   R3,-(SP)    ;; SAVE R3
*         MOV   R4,-(SP)    ;; SAVE R4
*         MOV   R5,-(SP)    ;; SAVE R5
*         MOVVB SOMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
*         NEG   R4
*         ADD   #6,R4       ;; SUBTRACT IT FOR MAX. ALLOWED
*         MOVVB R4,SOMODE   ;; SAVE IT FOR USE
*         MOVVB SOFILL,R4   ;; GET THE ZERO FILL SWITCH
*         MOV   12(SP),R5   ;; PICKUP THE INPUT NUMBER
*         CLR   R3         ;; CLEAR THE OUTPUT WORD
*         ROL  R5         ;; ROTATE MSB INTO "C"
*         BR   28:        ;; GO DO MSB
*         ROL  R5         ;; FORM THIS DIGIT
*         ROL  R5
*         ROL  R5
*         MOV  R5,R3
*         ROL  R3         ;; GET LSB OF THIS DIGIT
*         DECB SOMODE     ;; TYPE THIS DIGIT?
*         BPL  78:        ;; BR IF NO
*         BIC  #177770,R3 ;; GET RID OF JUNK
*         BNE  48:        ;; TEST FOR 0
*         TST  R4         ;; SUPPRESS THIS 0?
*         BEQ  58:        ;; BR IF YES
*         INC  R4         ;; DON'T SUPPRESS ANYMORE 0'S
*         BIS  #'0,R3     ;; MAKE THIS DIGIT ASCII
*         BIS  #' ,R3     ;; MAKE ASCII IF NOT ALREADY
*         MOVB R3,R5     ;; SAVE FOR TYPING
    
```

1711	005606	104401	005646
1712	005612	105337	005650
1713	005616	003347	
1714	005620	002402	
1715	005622	005204	
1716	005624	000744	
1717	005626	012605	
1718	005630	012604	
1719	005632	012603	
1720	005634	016666	000002 000004
1721	005642	012616	
1722	005644	000002	
1723	005646	000	
1724	005647	000	
1725	005650	000	
1726	005651	000	
1727	005652	000000	
1728			

```

7S:   TYPE      BS
      DECB     $OCNT
      BGT      2S
      BLT      6S
      INC      R4
      BR       2S
6S:   MOV      (SP)+,R5
      MOV      (SP)+,R4
      MOV      (SP)+,R3
      MOV      2(SP),4(SP)
      MOV      (SP)+,(SP)
      RTI
BS:   .BYTE    0
      .BYTE    0
$OCNT: .BYTE    0
$OFILL: .BYTE    0
$OMODE: .WORD   0
      .SBTTL  MODULE 2.0 INTERRUPT HANDLERS
      :: GO TYPE THIS DIGIT
      :: COUNT BY 1
      :: BR IF MORE TO DO
      :: BR IF DONE
      :: INSURE LAST DIGIT ISN'T A BLANK
      :: GO DO THE LAST DIGIT
      :: RESTORE R5
      :: RESTORE R4
      :: RESTORE R3
      :: SET THE STACK FOR RETURNING
      :: RETURN
      :: STORAGE FOR ASCII DIGIT
      :: TERMINATOR FOR TYPE ROUTINE
      :: OCTAL DIGIT COUNTER
      :: ZERO FILL SWITCH
      :: NUMBER OF DIGITS TO TYPE

```

1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784

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.
    
```

```

005654 016646 000002
005660 042716 000020
005664 012746 005672
005670 000002
005672 010046
005674 016600 000002
005700 005740
005702 111000
005704 006300
005706 016000 005726
005712 000200
    
```

```

$TRAP:  MOV    2(SP),-(SP)    ; ASSUME THE STATUS OF
        BIC    #20,(SP)      ; THE CALLER--DO NOT ALLOW
        MOV    #1$,-(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

```

005714 011646
005716 016666 000004 000002
005724 000002
    
```

```

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

```

.MACRO SETTRAP A,B,MSG
$$SET A,B,\(TRAP+$TRP),\$TRP,(MSG)
    
```

```

.NLIST
$TRP=$TRP+1
.LIST
.ENDM SETTRAP
.MACRO $$SET A,B,C,D,COMNT
.IF EQ $TRP-1
.SBTTL TRAP TABLE
    
```

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

```

; ROUTINE
; -----
$TRPAD: .WORD  $TRAP2
.ENDC
.IIF NDF GNS,.NLIST
A= C
    
```

1785
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

```
.IIF NOF GMS,.LIST
      B          ;;CALL=A          TRAP+D(C)      COMNT
.ENDM  $$SET
.MACRO TRMTRP
$TERM=-$TRPAD
.ENDM  TRMTRP
.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
```

.SBTTL POWER DOWN AND UP ROUTINES

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

```
POWER UP ROUTINE
$PWRUP: MOV  $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
        MOV  $$SAVR6,SP      ;;GET SP
        CLR  $$SAVR6        ;;WAIT LOOP FOR THE TTY
        IS: INC  $$SAVR6     ;;WAIT FOR THE INC
        BNE  IS            ;;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
```

E04

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 42
POWER DOWN AND UP ROUTINES

1841	006064	012601			MOV	(SP)+,R1	:: POP STACK INTO R1
1842	006066	012600			MOV	(SP)+,R0	:: POP STACK INTO R0
1843	006070	012737	005752	000024	MOV	#\$PWRDN,2#\$PWRVEC	:: SET UP THE POWER DOWN VECTOR
1844	006076	012737	000340	000026	MOV	#\$340,2#\$PWRVEC+2	:: Prio:7
1845	006104	104401			TYPE		:: REPORT THE POWER FAILURE
1846	006106	006124			\$PWRMG: .WORD	\$POWER	:: POWER FAIL MESSAGE POINTER
1847	006110	012716			MOV	(PC)+,(SP)	:: RESTART AT RESTART
1848	006112	001334			\$PWRAD: .WORD	RESTART	:: RESTART ADDRESS
1849	006114	000002			RTI		
1850	006116	000000			\$ILLUP: HALT		:: THE POWER UP SEQUENCE WAS STARTED
1851	006120	000776			BR	.-2	:: BEFORE THE POWER DOWN WAS COMPLETE
1852	006122	000000			\$SAVR6: 0		:: PUT THE SP HERE
1853	006124	005015	047520	042527	\$POWER: .ASCIZ	<15><12>"POWER"	
1854	006132	000122					
1855					.EVEN		

.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.

1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911

```

006134 004537 006530
006140 004737 006560
006144 004537 010274
006150 100402
006152 013700 011634
006156 012720 006230
006162 012710 000340
006166 012737 000402 011706
006174 004737 006510
006200 012737 100525 011574
006206 053777 011712 003320
006214 013777 011716 003316
006222 004537 006366
006226 177376
006230 005737 006506
006234 001006
006236 012716 006244
006242 000002
006244 012705 006252
006250 000205
006252 013700 011704
006256 013701 011706
006262 123720 011574
006266 001404
006270 004537 010746
006274 003001
006276 011762
006300 005301
006302 001367
006304 042777 000200 003222
006312 042777 000004 003234
006320 042777 000030 003206
006326 105737 011574
006332 100404
006334 012737 100652 011574
006342 000721
    
```

```

TST1: JSR RS,INIT ;INITIALIZE THE SOFTWARE, PS, ETC
      JSR PC,DXRES ;PUT THE DX IN A GOOD KNOWN STATE
      JSR RS,BUFAD ;GET FIRST BUFFER BASE AND SET UP KT
      .WORD 100402 ;CTL BIT 15=INITIALIZE, BUFFER BASE DIVISABLE
      ;BY 400, SET UP BUFMAX
      MOV AVECT,RO ;SET UP
      MOV #INTR1,(RO)+ ;DX INTERRUPT
      MOV #340,(RO) ;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,DXM00 ;SET DATA PATTERN AND OPOUT
      BIS EACUR,DXCSA ;SET UP DX EXT ADDR BITS
      MOV BFBAPH,DXBAA ;SET UP DX BUS ADDR REG.
      JSR RS,DXG0 ;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 B$ ;BR IF NO, STACK IS OK
      MOV #5$,JSP ;SET UP RETURN ADDR
      RTI ;RELIEVE INTERRUPT STATUS
5$: MOV #8$,RS ;SET UP JSR RETURN ADDR
      RTS RS ;RELIEVE SUBROUTINE STATUS
8$: MOV BUFBAS,RO ;GET STARTING ADDR
      MOV BUFSIZE,R1 ;GET BUFFER SIZE
10$: CMPB DXM00,(RO)+ ;CHECK A BYTE
      BEQ 15$ ;BRANCH IF NO ERROR
12$: JSR RS,ERRR ;REPORT AN ERR
      .WORD 1 ;ERROR ID
      .WORD ERR1 ;ADDR OF ERROR MESSAGE
15$: 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
      TSTB DXM00 ;CHECK IF FIRST OF TWO PASSES
      BMI 30$ ;BR IF NOT FIRST PASS
      MOV #100652,DXM00 ;SET UP SECOND DATA PATTERN
      BR G01 ;DO TRANSFER WITH THIS DATA
    
```

```

1912 006344 004737 006510 30$: JSR PC, BUFCLR ; PUT BUFFER BACK TO ZEROS
1913 006350 004537 010274 ENDT1: JSR R5, BUFAD ; GET NEXT BUFFER ADDRESS
1914 006354 000402 .WORD 402 ; BASE TO BE DIVISABLE BY 400
1915 006356 005701 TST R1 ; CHECK FOR OUT OF MEMORY
1916 006360 001702 BEQ LOOP1 ; BRANCH IF THERE IS MORE
1917 006362 000137 001410 JMP ENDTST ; OTHERWISE WE'RE DONE WITH TEST
1918
1919 ;
1920 ; SUBROUTINE TO START THE DX IN SOSIEN MODE
1921 006366 013777 011574 003150 DXGO: MOV DXMOO, 2DXMOA ; SEND DATA AND OP OUT TO MAINT REG
1922 006374 012577 003142 MOV (R5)+, 2DXBCA ; SET UP BYTE COUNT FROM PASSED PARM.
1923 006400 005046 CLR -(SP) ; PUT NEW PS ON STACK
1924 006402 012746 006410 MOV #645, -(SP) ; PUT NEW PC ON STACK
1925 006406 000002 RTI ; POP NEW PC AND PS
1926 006410
1927 006410 052777 000103 003116 645: BIS #103, 2DXCSA ; SET FUNCTION TO IBM WRITE
1928 AND ENABLE INTERRUPTS
1929 006416 052777 060000 003120 BIS #SELO:HLDO, 2DXMOA ; FORCE DX SELECTION
1930 006424 042777 060000 003112 BIC #SELO:HLDO, 2DXMOA ; DROP SELECT LINES
1931 006432 052777 002000 003104 BIS #CMDO, 2DXMOA ; RAISE COMMAND OUT
1932 006440 042777 002000 003076 BIC #CMDO, 2DXMOA ; DROP COMMAND OUT
1933 006446 052777 000004 003100 BIS #SOSIEN, 2DXESA ; ENABLE FAST SERVICE OUT/SERV IN
1934 TO UNLEASH NPR'S
1935 006454 005037 006506 CLR INTERR ; CLEAR NO INTERRUPT OCCURED FLAG
1936 006460 005001 CLR R1
1937 006462 005301 10$: DEC R1 ; WAIT HERE FOR INTERRUPT
1938 006464 001376 BNE 10$
1939 006466 004537 010746 JSR R5, ERRR ; ERROR, INTERRUPT NEVER CAME
1940 006472 000002 .WORD 2 ; ERROR IDENTIFER
1941 006474 012006 .WORD ERR2 ; ERROR MESSAGE POINTER
1942 006476 012737 000001 006506 MOV #1, INTERR ; SET NO INTERRUPT OCCURED FLAG
1943 006504 000205 RTS R5 ; CONTINUE TESTING
1944 006506 000000 INTERR: .WORD 0 ; SET TO 1 IF DX11 FAILED TO INTERRUPT
1945
1946 ;
1947 ; SUBROUTINE TO CLEAR THE CURRENT BUFFER
1948
1949 006510 013700 011704 BUFCLR: MOV BUFBAS, R0 ; GET BUFFER START
1950 006514 013701 011706 MOV BUFSIZE, R1 ; GET BUFFER SIZE
1951 006520 105020 10$: CLRB (R0)+ ; CLEAR A BYTE
1952 006522 005301 DEC R1 ; COUNT IT
1953 006524 001375 BNE 10$ ; LOOP TO CLEAR BUFFER
1954 006526 000207 RTS PC ; RETURN TO CALLER
1955

```

1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995

006530
006530 012746 000340
006534 012746 006542
006540 000002
006542
006542 005037 011672
006546 010537 011676
006552 005037 006506
006556 000205

006560 052777 000010 002766
006566 042777 000200 002740
006574 042777 000006 002732
006602 052777 000001 002724
006610 042777 001230 002716

006616 012777 014000 002712
006624 005737 011760
006630 001403
006632 012777 016000 002676
006640 032777 100000 002702
006646 001402
006650 000005
006652 000742
006654 000207

```
.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 SPWVD ;;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
```

.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.

1996					
1997					
1998					
1999					
2000					
2001					
2002					
2003					
2004					
2005					
2006					
2007					
2008					
2009					
2010					
2011					
2012					
21 13					
2014	006656	004537	006530		
2015	006662	004737	006560		
2016	006666	004537	010274		
2017	006672	101000			
2018	006674	013700	011634		
2019	006700	012720	006766		
2020	006704	012710	000340		
2021	006710	012737	001000	011706	LOOP2:
2022	006716	004737	006510		
2023	006722	013737	011704	011714	
2024	006730	013737	011716	011720	
2025	006736	053777	011712	002570	LOP2:
2026	006744	012737	100377	011574	
2027					
2028	006752	013777	011720	002560	
2029	006760	004537	006366		
2030	006764	177776			
2031					
2032					
2033	006766	005737	006506		
2034	006772	001006			
2035	006774	012716	007002		
2036	007000	000002			
2037	007002	012705	007010		
2038	007006	000205			
2039	007010	027727	002700	177777	
2040					
2041	007016	001404			
2042	007020	004537	010746		
2043	007024	000003			
2044	007026	012026			
2045	007030	005077	002660		
2046	007034	042777	000200	002472	
2047	007042	042777	000004	002504	
2048	007050	042777	006030	002456	
2049	007056	062737	000002	011720	
2050	007064	062737	000002	011714	
2051	007072	023737	011714	011710	

```

TST2: JSR      R5,INIT          ;INITIALIZE THE SOFTWARE
      JSR      PC,DXRES        ;INITIALIZE THE HARDWARE
      JSR      R5,BUFAD        ;GET STARTING BUFFER ADDRESS
      .WORD    101000          ;PARAM MEANS INIT AND SET UP1000 BYTE BUFFER
      MOV      AVECT,R0        ;GET THE DX VECTOR ADDR
      MOV      #INTR2 (R0)+    ;SET UP RETURN ADDR
      MOV      #340 (R0)      ;SET INTR PSH
      MOV      #1000,BUF SIZE ;ESTABLISH BUFFER SIZE
      JSR      PC,BUF CLR      ;CLEAR THE BUFFER
      MOV      BUF BAS,DATA0   ;INITIALIZE DATA ADDR
      MOV      BFBAPH,BFADPH   ;GET PHYSICAL BASE ADDR
      LOP2:   BIS      EA CLR,DXCSA ;SET EA BITS IN THE DX11
      MOV      #100377,DXM00    ;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      INTR          ;DID WE COME HERE VIA DX INTR
      BNE      B$            ;BR IS NO, STACK IS OK
      MOV      #5$,RSP        ;SET UP RETURN ADDR
      RTI                      ;RELIEVE INTERRUPT STATUS
      MOV      #8$,R5         ;SET UP RETURN ADDR
      RTS      R5             ;RELIEVE SUBROUTINE STATUS
      B$:   CMP      @DATA0,#-1 ;DID DATA GET WHERE IT WAS SUPPOSED TO?
      ;ASSUME CPU CAN ADDRESS MEMORY CORRECTLY
      BEQ      10$            ;BRANCH IF OK
      JSR      R5,ERRR        ;REPORT THE ERROR
      .WORD    3              ;ERROR NUMBER THREE
      .WORD    ERR3           ;ADDR OF ERROR MESSAGE
      10$:  CLR      @DATA0    ;RESTORE THE WORD UNDER TEST
      BIC      #DONE,@DXCSA   ;TURN OFF INTERRUPTS
      BIC      #SOSTEN,@DXESA ;CLEAR FAST MODE
      BIC      #30,@DXCSA     ;CLEAR POSSIBLE CARRY IN EA BITS
      ADD      #2,BFADPH       ;BUMP THE PHYSICAL ADDRESS
      ADD      #2,DATA0        ;BUMP DATA POINTER
      CMP      DATA0,BUFMAX   ;DONE WITH BUFFER?

```

J04

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 47
MODULE 3.0.2 TST2 NPR ADDRESS UNIQUENESS

2052	007100	101716	BLOS	LOP2	;BRANCH IF NO
2053					
2054	007102	004737	JSR	PC, BUFCLR	;RESTORE BUFFER
2055	007106	004537	JSR	RS, BUFAD	;GET NEXT BUFFER
2056	007112	001000	WORD	1000	;SIZE =1000
2057	007114	005701	TST	R1	;DID WE GET ONE?
2058	007116	001674	BEQ	LOOP2	;BRANCH IF YES
2059	007120	000137	JMP	ENDTST	;OTHERWISE TERMINATE TEST

K04

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 48
TST3 SPW ADDRESSING

.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.

2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101 007124 004537 006530
2102 007130 005037 011740
2103 007134 005037 011742
2104 007140 005037 011746
2105 007144 004737 006560
2106 007150 012737 000400 011726
2107 007156 004537 010274
2108 007162 102000
2109 007164 012737 002000 011706
2110 007172 004737 006510
2111 007176 005037 011724
2112 007202 013700 011704
2113 007206 013720 011724
2114 007212 005237 011724
2115 007216 022737 000377 011724

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 DEVC ;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 BUFBAS,R0 ;GET SPW TABLE BASE
10S: 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?

2116	007224	103370				BHIS	10\$: DO MORE IF IT IS NOT
2117	007226	005037	011724			CLR	DEV	: START AT DEV 0
2118	007232	013777	011716	002276	LOP3:	MOV	BFBAPH, 20XOSA	: SET UP OFFSET FOR SPW
2119	007240	053777	011712	002266		BIS	EACUR, 20XCSA	: SET UP EXT ADDR BITS
2120	007246	004537	007514			JSR	RS, PTYGEN	: ASSIGN PARITY TO THE DEVICE BYTE
2121	007252	011724				.WORD	DEV	: ADDR OF DATA TO ASSIGN PARITY TO
2122	007254	113777	011724	002250		MOV	DEV, 20XCRA	: SET DEV TO CUAR
2123	007262	053777	011724	002254		BIS	DEV, 20XMOA	: PUT DEV ADDR ON BUSOUT
2124	007270	052777	004000	002246		BIS	ADRO, 20XMOA	: RAISE ADDR OUT
2125	007276	032777	000001	002244		BIT	#1, 20XCBA	: IS ADRECC ON?
2126	007304	001452				BEQ	20\$: BR IF ADDR NOT RECOGNIZED
2127	007306	032777	000002	002234		BIT	#2, 20XCBA	: IS ADRECC ON?
2128	007314	001446				BEQ	20\$: BR IF ADDR NOT RECOGNIZED
2129	007316	004537	007554			JSR	RS, SAVDEV	: GO SAVE RECOGNIZED DEV DATA
2130	007322	052777	060000	002214		BIS	ADRO!SELO, 20XMOA	: RAISE SEL OUT, HOLD OUT
2131	007330	042777	004000	002206		BIC	ADRO, 20XMOA	: DROP ADRO
2132	007336	043777	011724	002200		BIC	DEV, 20XMOA	: REMOVE DEVICE FROM BUSO
2133	007344	053777	011726	002172		BIS	CMD, 20XMOA	: PUT CMD =0 ON BUSOUT
2134	007352	052777	002000	002164		BIS	BCMD0, 20XMOA	: RAISE CMD OUT
2135	007360	043777	011726	002156		BIC	CMD, 20XMOA	: REMOVE CMD FROM BUSO
2136	007366	042777	002000	002150		BIC	BCMD0, 20XMOA	: DROP CMD OUT
2137	007374	123777	011724	002134		CMPB	DEV, 20XOSA	: IS STATUS EQUAL TO DEV?
2138	007402	001404				BEQ	10\$: BR IF OK
2139	007404	004537	010746			JSR	RS, ERRR	: REPORT ERROR
2140	007410	000004				.WORD	4	: ERROR ID
2141	007412	012065				.WORD	ERRR	: ERROR MESSAGE ADDR
2142	007414	052777	001000	002122	10\$:	BIS	SRV0, 20XMOA	: RAISE SERV CE OUT TO RELIEVE STATUS
2143	007422	042777	001000	002114		BIC	SRV0, 20XMOA	: DROP SERVICE OUT
2144	007430	000402				BR	30\$	
2145	007432	004537	007620		20\$:	JSR	RS, DONTSV	: GO HANDLE RECOGNIZED DEV DATA
2146	007436	004737	006560		30\$:	JSR	PC, DXRES	: RESTORE DX
2147	007442	005237	011724			INC	DEV	: GO TO NEXT DEV
2148	007446	105737	011724			TSTB	DEV	: LAST DEV?
2149	007452	001267				BNE	LOP3	: BR IF MORE DEVICES
2150	007454	004737	006510			JSR	PC, BUFCLR	: CLEAR OLD BUFFER BEFORE GETTING NEW ONE
2151	007460	004537	010274			JSR	RS, BUFAD	: GET NEXT BUFFER BASE
2152	007464	002000				.WORD	2000	: DIVISABLE BY 2000
2153	007466	005701				TST	R1	: DID WE GET ANOTHER BUFFER?
2154	007470	001642				BEQ	LOOP3	: BR IF YES AND DO NEXT ONE
2155	007472	005737	011746			TST	DEVCT	: TEST IS DONE, DID WE EVER FIND A
2156								: RECOGNIZED DEVICE?
2157	007476	001004				BNE	40\$: BR IF YES
2158	007500	004537	010746			JSR	RS, ERRR	: ERROR, NO ADREDD AND ADRECD FOR ANY
2159								: IBM DEVICE NUMBER
2160	007504	000005				.WORD	5	
2161	007506	012114				.WORD	ERRS	: ADDR OF ERROR MSG
2162	007510	000137	001410		40\$:	JMP	ENDTST	: TERMINATE TEST
2163								
2164								
2165								
2166								
2167								
2168								
2169								
2170								
2171								

PARITY ASSIGNMENT SUBROUTINE

CALLLED AS FOLLOWS:
JSR RS,PTYGEN

M04

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 50
TST3 SPW ADDRESSING

```

2172 ; .WORD ADDR ; ADDRESS OF DATA TO WHICH PARITY WILL BE
2173 ; RETURN HERE ; ASSIGNED
2174 ;
2175 ;
2176 ;
2177 ;
2178 ;
2179 007514 012503 PTYGEN: MOV (R5)+,R3 ; GET ADDR OF DATA
2180 007516 011304 MOV (R3),R4 ; GET THE DATA
2181 007520 005002 CLR R2 ; INITIALIZE THE PARTIY SW
2182 007522 106304 PTY2: ASLB R4 ; CHECK A BIT
2183 007524 102001 BVC PTY3 ; BR IF ONES
2184 007526 005102 COM R2 ; OTHERWISE FLIP THE PARITY SW
2185 007530 106304 PTY3: ASLB R4 ; TWO BITS GET CHECKED AT ONCE
2186 007532 001373 BNE PTY2 ; IF R4 = 0 WE'D BE DONE
2187 007534 005702 TST R2 ; DID THE SWITCH END UP SET?
2188 007536 100403 BMI PTY4 ; BR IF NO
2189 007540 052713 000400 BIS #400,(R3) ; SET THE BIT IN THE TARGET DATA
2190 007544 000402 BR PTY5 ; EXIT
2191 007546 042713 000400 PTY4: BIC #400,(R3) ; CLEAR THE BIT IN THE TARGET DATA
2192 007552 000205 PTY5: RTS R5 ; RETURN
2193
2194
2195
2196 007554 005737 011740 SAVDEV: TST RECORD ; ARE WE RECORDING DEVICES FOUND?
2197 007560 001016 BNE 20$ ; BR IF NO AND EXIT
2198 007562 005737 011742 TST FRSTD ; IS THIS THE VERY FIRST DEV RECOGNIZED?
2199 007566 001011 BNE 10$ ; BR IF NO
2200 007570 052737 000001 011742 BIS #1,FRSTD ; SET FIRST DEV FOUND SW
2201 007576 113737 011724 011744 MOVB DEV,FRSTDV ; SAVE FIRST DEVICE FOUND
2202 007604 012737 000001 011756 MOV #1,DEVFND ; SET SWITCH = DEVICE FOUND
2203 007612 005237 011746 10$: INC DEVCT ; COUNT THE DEV
2204 007616 000205 20$: RTS R5
2205
2206
2207
2208
2209 007620 005737 011746 DONT$V: TST DEVCT ; HAVE WE FOUND ANY DEV YET?
2210 007624 001403 BEQ 30$ ; BR IF NO
2211 007626 052737 000001 011740 BIS #1,RECORD ; TURN OFF DEV RECORDING
2212 007634 000205 30$: RTS R5
2213
2214
2215

```

.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 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.

2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238 007636 004537 006530
2239 007642 004737 006560
2240 007646 005737 011756
2241 007652 001006
2242 007654 004537 005134
2243 007660 012211
2244 007662 000000
2245 007664 000137 010270
2246 007670 005037 011724
2247 007674 113737 011744 011724
2248 007702 004537 007514
2249 007706 011724
2250 007710 004537 010274
2251 007714 100400
2252 007716 012737 000400 011706
2253 007724 004737 006510
2254 007730 005000
2255 007732 013701 011704
2256 007736 110021
2257 007740 005200
2258 007742 022700 000377
2259 007746 103373
2260 007750 005000
2261 007752 113700 011724
2262 007756 006300
2263 007760 017701 001552
2264 007764 042701 001777
2265 007770 050100
2266 007772 005737 011646
2267 007776 001412
2268 010000 013701 011712
2269 010004 000301
2270 010006 006201
2271 010010 010137 172352

```
TST4: JSR RS,INIT ;INITIALIZE THE SOFTWARE
        JSR PC,DXRES ;INITIALIZE THE HARDWARE
        TST DEVFND ;DO WE HAVE A DEVICE?
        BNE IOS ;BR IF YES
        JSR RS,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
IOS: CLR DEV ;CLEAN UP CURRENT DEV NO.
      MOVB FRSTDV,DEV ;GET THE VALID DEV
      JSR RS,PTYGEN ;ASSIGN PARITY TO IT
      .WORD DEV
      JSR RS,BUFAD ;GET A BUFFER OF 256 BYTES
      .WORD 100400
      MOV #400,BUFSE ;SET BUFFER SIZE
      JSR PC,BUFCLR ;CLEAR THE BUFFER
      CLR RO ;START WITH CMD = 0
      MOV BUFBAS,R1 ;GET THE DST BASE
      MOVB RO,(R1)+ ;SET CMD IN DST
      INC RO ;NEXT CMD
      CMP #255,RO ;ARE ALL STATUS = TO CMD?
      BHIS IOS ;DO 256 BYTES
      CLR RO
      MOVB DEV,RO ;GET DEVICE
      ASL RO ;MAKE IT A WORD OFFSET
      MOV #DXOSA,R1 ;GET THE SPW BASE
      BIC #1777,R1 ;CLEAN IT UP
      BIS R1,RO ;ADD DEV TO OFFSET
      TST KT ;MM IN USE?
      BEQ IOS ;BR IF NO
      MOV EACUR,R1 ;GET EXT ADDR BITS
      SWAB R1 ;POSITION FOR USE IN PAR
      ASR R1
      MOV R1,#KIPARS ;USE PARS TO GET TO SPW TABLE
```

2272	010014	042700	040000			BIC	#40000, R0	; SET SPW ADDR TO USE PARS
2273	010020	052700	120000			BIS	#120000, R0	
2274	010024	013701	011716		15\$:	MOV	BFBAPH, R1	; SET PHY BUFFER BASE
2275	010030	105001				CLRB	R1	; IMMED STATUS = 0
2276	010032	010110				MOV	R1, (R0)	; SET UP SPW
2277	010034	005037	011726			CLR	CMD	; START WITH CMD = 0
2278	010040	004537	007514		LOP4:	JSR	R5, PTYGEN	; GET COMMAND PARITY
2279	010044	011726				.WORD	CMD	
2280	010046	053777	011712	001460		BIS	EACUR, 20XCSA	; SET UP EXT ADDR BITS
2281	010054	113777	011724	001450		MOVB	DEV, 20XCAA	; SET DEVICE TO CUAR
2282	010062	053777	011724	001454		BIS	DEV, 20XMOA	; PUT DEV ADDR ON BUS OUT
2283	010070	052777	004000	001446		BIS	#ADR0, 20XMOA	; ADDR OUT
2284	010076	052777	060000	001440		BIS	#HLDO, SELO, 20XMOA	; SELECT & HOLD OUT
2285	010104	042777	004000	001432		BIC	#ADR0, 20XMOA	; DROP ADDR OUT
2286	010112	043777	011724	001424		BIC	DEV, 20XMOA	; DROP DEV FROM BUS
2287	010120	053777	011726	001416		BIS	CMD, 20XMOA	; COMMAND TO BUS
2288	010126	052777	002000	001410		BIS	#CMD0, 20XMOA	; COMMAND OUT
2289	010134	043777	011726	001402		BIC	CMD, 20XMOA	; CLR CMD FROM BUS
2290	010142	042777	002000	001374		BIC	#CMD0, 20XMOA	; DROP COMMAND OUT
2291	010150	122737	000004	011726		CMPB	#4, CMD	; IS IT A SENSE COMMAND?
2292	010156	001004				BNE	5\$; BR IF NO
2293	010150	105777	001352			TSTB	20XOSA	; STATUS FORCED TO ZERO ON SENSE CMD
2294	010164	001411				BEQ	10\$; BR IF GOOD
2295	010166	000404				BR	7\$; REPORT ERROR
2296	010170	123777	011726	001340	5\$:	CMPB	CMD, 20XOSA	; IS STATUS EQUAL DEVI.E?
2297	010176	001404				BEQ	10\$; BR IF YES
2298	010200	004537	010746		7\$:	JSR	R5, ERRR	; ERROR IN DST ACCESS
2299	010204	000006				.WORD	6	; ERROR #
2300	010206	012170				.WORD	ERR6	; MESSAGE ADDR
2301	010210	052777	001000	001326	10\$:	BIS	#SRV0, 20XMOA	; RELIEVE STATUS
2302	010216	042777	001000	001320		BIC	#SRV0, 20XMOA	
2303	010224	004737	006560			JSR	PC, DXRES	; CLEAR THE DX
2304	010230	005237	011726			INC	CMD	; TRY NEXT COMMAND
2305	010234	105737	011726			TSTB	CMD	; DONE IF BACK TO ZERO
2306	010240	001277				BNE	LOP4	; BR TO DO NEXT CMD
2307	010242	004737	006510			JSR	PC, BUFCLR	; CLEAR OLD BUFFER
2308	010245	004537	010274		20\$:	JSR	R5, BUFAD	; GET NEXT BUFFER BASE
2309	010252	000400				.WORD	400	
2310	010254	005701				TST	R1	; DID WE GET A BUFFER?
2311	010256	001004				BNE	END4	; BR IF NO END TEST
2312	010260	005737	011716			TST	BFBAPH	; IS BUFFER BASE 0'S?
2313	010264	001770				BEQ	20\$; BR IF YES, THE DX DOES NOT DO A
2314								; DST FETCH IF SPW ENTRY IS ALL ZEROS
2315	010266	000620				BR	LOP4	; DO NEXT BUFFER
2316	010270	000137	001410		END4:	JMP	ENDTST	; END OF TEST 4

```

2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337 010274 012500
2338 010276 005001
2339 010300 005700
2340 010302 100050
2341 010304 005037 011712
2342
2343 010310 012702 016000
2344 010314 042700 100000
2345 010320 060002
2346 010322 162700 000001
2347 010326 040002
2348 010330 010237 011704
2349 010334 010237 011716
2350
2351 010340 005737 011646
2352 010344 001421
2353 010346 012703 000006
2354 010352 005037 172354
2355 010356 006202
2356 010360 077302
2357 010362 010237 172354
2358 010366 042737 177700 011704
2359 010374 052737 140000 011704
2360 010402 012737 000001 177572
2361 010410 063700 011704
2362 010414 005200
2363 010416 010037 011710
2364 010422 000550
2365
2366
2367
2368
2369 010424 060037 011704
2370 010430 042737 000077 011704
2371 010436 060037 011716
2372 010442 042737 000077 011716
    
```

.SBTTL BASE ADDRESS GENERATOR MODULE
MODULE 3.2 BASE ADDRESS GENERATOR

THIS MODULE SETS UP THE STARTING ADDRESS FOR DX BUFFERS AND HAS RESPONSIBILITY FOR ESTABLISHING EXTENDED ADDR BITS, BUFFER BASE, BUFFER SIZE, AND HAS CONTROL OF THE MEMORY MGT. REGISTERS. KIPAR6 IS ALWAYS USED TO ACCESS THE DXBUFFERS IF MEM MGT. IS ENABLED.

THIS MODULE IS CALLED AS FOLLOWS:

JSR RS, BUFAD
.WORD PARM
RETURN HERE WITH R1=0 MEANS BUFFER ALLOCATED
R1=1 MEANS NO MEMORY LEFT

PARM IS DEFINED AS FOLLOWS:
BIT 15=1 INITIALIZE TO FIRST BUFFER LOCATION
BIT 15=0 GET NEXT BUFFER LOCATION
BITS 14:0 BASE ADDR OF BUFFER MUST BE DIVISIBLE BY THIS NUMBER

```

BUFAD: MOV (RS)+, R0 ; GET THE PASSED PARAMETER
      CLR R1 ; RESET "OUT OF MEMORY" INDICATOR
      TST R0 ; CHECK FOR INITIALIZE SW
      BPL NEXTB ; BRANCH IF NOT TO INITIALIZE
      CLR EACUR ; START INIT PROCESS BY CLEARING
      ; THE EA BITS
      MOV #ENDALL, R2 ; GET VIRTUAL END OF PROGRAM ADDR
      BIC #100000, R0 ; CLEAR HIGH BIT OF PARAM
      ADD R0, R2 ; OFFSET ABOVE PROGRAM
      SUB #1, R0 ; DEVELOP A MASK
      BIC R0, R2 ; TO MAKE BASE DIVISIBLE BY PROPER AMT.
      MOV R2, BUFBAS ; SAVE VIRTUAL BUFFER BASE
      MOV R2, BFBAPH ; SAVE PHYSICAL BUFFER BASE, SAME AS VIRT
      ; AT THIS POINT.
      TST KT ; IS KT IN USE?
      BEQ ETCB ; BRANCH IF NO TO END OF MODULE
      MOV #1, R3 ; INIT LOOP COUNTER
      CLR #IPAR6 ; START TO INITIALIZE KIPAR6
105: ASR R2 ; SHIFT ADDRESS RIGHT 6 PLACES
      SOB R3, .05 ; TO ALIGN WITH PAR
      MOV R2, #KIPAR6 ; SET UP PAR6 TO REFERENCE FIRST BUFFER
      BIC #177700, BUFBAS ; CLEAR UP VIRTUAL BUFFER ADDR
      BIS #140000, BUFBAS ; BUFFER ACCESS IS ALWAYS THRU PAR6
      MOV #1, #SR0 ; TURN ON MM
ENDB: ADD BUFBAS, R0 ; FIND TOP OF CURRENT BUFFER
      INC R0 ;
      MOV R0, BUFMAX ; SAVE VIRT END OF BUFFER
      BR ENDB1 ; DONE WITH INITIALIZATION
    
```

COME HERE IF NEXT BUFFER ADDR IS WANTED

```

NEXTB: ADD R0, BUFBAS ; INCR TO NEXT BUFFER SPACE
      BIC #77, BUFBAS ; CLEAR POSSIBLE LOW ORDER BITS
      ADD R0, BFBAPH ; STEP PHYSICAL ADDR
      BIC #77, BFBAPH ; CLEAR POSSIBLE LOW ORDER BITS
    
```


2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483

.SBTTL MODULE 3.3 REPORT ERRORS
MODULE 3.3 ERROR REPORTING

CALLED AS FOLLOWS:

JSR R5,ERRR
.WORD X
.WORD Y
RETURN (UNLESS LOCK ON ERROR FUNCTION OVERRIDES RETURN)
X EQUALS ERROR IDENTIFICATION NUMBER
Y EQUALS ADDR OF UNIQUE ERROR MESSAGE

THIS MODULE FORMATS AND PRINTS ERROR MESSAGES AND HANDLES
ERROR OPTIONS SUCH AS INHIBIT ERROR MESSAGES, SHORT ERROR
REPORT, SCOPE LOOP, HALT ON ERROR, ETC.

010746	032777	100000	170164	ERRR:	BIT	#SW15,@SWR	;HALT ON ERROR ON?
010754	001401				BEQ	5\$;BR IF NO
010756	000000				HALT		;SW15 ON AND ERROR OCCURRED.
010760	013737	011662	011672	5\$:	MOV	CTESTN,ERR	;SET ERROR DETECTED SW
010766	005046				CLR	-(SP)	;PUT NEW PS ON STACK
010770	012746	010776			MOV	#64\$,-(SP)	;PUT NEW PC ON STACK
010774	000002				RTI		;POP NEW PC AND PS
010776				64\$:			
010776	005737	011674			TST	LOCK	;ARE WE LOCKED ON ERROR?
011002	001407				BEQ	10\$;BR IF NO
011007	032777	040000	170126		BIT	#SW14,@SWR	;SCOPE LOOP SW ON?
011012	01413				BEQ	15\$;BR IF NO
011014	013705	011676			MOV	LOCKAD,R5	;GET THE LOOP ADDR
011020	000205				RTS	R5	;GO TO IT
011022	032777	040000	170110	10\$:	BIT	#SW14,@SWR	;SCOPE LOOP ON?
011030	001404				BEQ	15\$;BR IF NO
011032	013737	011662	011674		MOV	CTESTN,LOCK	;SET LOCKED ON ERR SW
011040	000402				BR	20\$	
011042	005037	011674		15\$:	CLR	LOCK	;TURN OFF LOCKED SW
011046	010537	011730		20\$:	MOV	R5,ERRPC	;SAVE ERR PC
011052	012537	011732			MOV	(R5)+,ERRID	;SAVE ERROR ID
011056	012537	011734			MOV	(R5)+,ERRMS	;SAVE ERROR MESSAGE ADDR
011062	032777	020000	170050		BIT	#SW13,@SWR	;INHIBIT PRINT ON?
011070	001402				BEQ	25\$;BR IF NO
011072	000137	011514			JMP	97\$;TERMINATE MODULE
011076	004537	005134		25\$:	JSR	R5,PRINT	;PRINT ERRPC
011102	012763				.WORD	ERRC	;ADDR OF MESSAGE
011104	000000				.WORD	0	;NO SPECIAL PRINT CONTROLS
011106	162737	000004	011730		SUB	#4,ERRPC	;BACKUP ADDR TO CORRECT VALUE
011114	004537	005134			JSR	R5,PRINT	;PRINT THE PC NUMBER ITSELF
011120	011730				.WORD	ERRPC	
011122	100001				.WORD	100001	;NO CRLF, CONVERT TO ASCII
011124	004537	005134			JSR	R5,PRINT	;PRINT ERROR NUMBER ON SAME LINE
011130	012776				.WORD	ERNO	
011132	000001				.WORD	1	;INHIBIT CRLF
011134	004537	005134			JSR	R5,PRINT	;PRINT THE OCT
011140	011732				.WORD	ERRID	
011142	100001				.WORD	100001	

2484	011144	013737	011734	011156		MOV	ERRMS,30\$;GET MESSAGE POINTER
2485	011152	004537	005134			JSR	RS,PRINT	;PRINT THE UNIQUE MESSAGE
2486	011156	000000			30\$:	.WORD	0	;FILLED FROM TWO LINES UP
2487	011160	000000				.WORD	0	;NO SPECIAL CONTROLS
2488	011162	032777	010000	167750		BIT	#SW12,@SWR	;SHORT ERROR REPORT ON?
2489	011170	001145				BNE	95\$;BR IF YES WE'RE DONE
2490	011172	004537	005134			JSR	RS,PRINT	;PRINT TEST NO.
2491	011176	013012				.WORD	ERTN	
2492	011200	000000				.WORD	0	
2493	011202	004537	005134			JSR	RS,PRINT	;PRINT THE OCTAL TEST NO.
2494	011206	011662				.WORD	CTESTN	
2495	011210	100001				.WORD	100001	
2496	011212	022737	000005	011732		CMP	#5,ERRID	;IS IT ERROR #5?
2497	011220	001531				BEQ	95\$;BR IF YES REST OF STUFF IS MEANINGLESS
2498	011222	004537	005134			JSR	RS,PRINT	;PRINT VIRT ADDR
2499	011226	013025				.WORD	ERAOR	
2500	011230	000000				.WORD	0	
2501	011232	004537	005134			JSR	RS,PRINT	;PRINT VIRT ADDR OCTAL
2502	011236	011704				.WORD	BUFBA\$	
2503	011240	100001				.WORD	100001	
2504	011242	005737	011646			TST	KT	;IS MEM MGT IN USE?
2505	011246	001431				BEQ	90\$;BR IF NO WE'RE DONE
2506	011250	004537	005134			JSR	RS,PRINT	;PRINT PHYSICAL ADDR
2507	011254	013060				.WORD	ERAOPH	
2508	011256	000000				.WORD	0	
2509	011260	004537	005134			JSR	RS,PRINT	;PRINT PHYSICAL ADDR OCTAL
2510	011264	011716				.WORD	BFBAPH	
2511	011266	100001				.WORD	100001	
2512	011270	004537	005134			JSR	RS,PRINT	;PRINT EA BITS
2513	011274	013105				.WORD	EABIT	
2514	011276	000000				.WORD	0	
2515	011300	013737	011712	011736		MOV	EACUR,EAERR	;GET THE EA BITS
2516	011306	006237	011736			ASR	EAERR	;POSITION BITS
2517	011312	006237	011736			ASR	EAERR	; TO BITS 1:0
2518	011316	006237	011736			ASR	EAERR	
2519	011322	004537	005134			JSR	RS,PRINT	;PRINT THE OCTAL EA BITS
2520	011326	011736				.WORD	EAERR	
2521	011330	100001				.WORD	100001	
2522	011332	023727	011732	000003	90\$:	CMP	ERRID,#3	;IS IT ERROR NO. 3?
2523	011340	001010				BNE	93\$;BR IF NO
2524	011342	004537	005134			JSR	RS,PRINT	;PRINT DATA ADDRESS ASCII
2525	011346	013126				.WORD	DATAA	
2526	011350	000000				.WORD	0	
2527	011352	004537	005134			JSR	RS,PRINT	;PRINT DATA ADDR OCT
2528	011356	011714				.WORD	DATAD	
2529	011360	100001				.WORD	100001	
2530	011362	023727	011732	000004	93\$:	CMP	ERRID,#4	;IS IT ERROR #4?
2531	011370	001004				BNE	94\$;BR IF NO
2532	011372	113737	011724	011754		MOVB	DEV,EXP	;GET DEV #
2533	011400	000416				BR	96\$	
2534	011402	023727	011732	000006	94\$:	CMP	ERRID,#6	;IS IT ERROR #6?
2535	011410	001035				BNE	95\$;BR IF NO
2536	011412	122737	000004	011726		CMPB	#4,CMD	;WAS IT A SENSE CMD?
2537	011420	001003				BNE	98\$;BR IF NO
2538	011422	005037	011754			CLR	EXP	;EXPECTED STATUS IS ZERO FOR SENSE
2539	011426	000403				BR	96\$	

G05

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
 DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006)
 MODULE 3.3

02-MAY-77 10:36 PAGE 57
 REPORT ERRORS

2540	011430	113737	011726	011754	98\$:	MOVB	CMD,EXP	;GET EXPECTED CMD
2541	011436	117737	000074	011752	96\$:	MOVB	00X05A,ACTU	;GET ACTUAL STATUS
2542	011444	004537	005134			JSR	RS,PRINT	;PRINT ACTUAL MESSAGE
2543	011450	013251				.WORD	ACTUMS	
2544	011456	000000				.WORD	0	
2545	011454	004537	005134			JSR	RS,PRINT	;PRINT STATUS OCTAL
2546	011460	011752				.WORD	ACTU	
2547	011466	100001				.WORD	100001	
2548	011464	004537	005134			JSR	RS,PRINT	;PRINT EXPECTED STATUS
2549	011470	013263				.WORD	EXPMS	
2550	011472	000001				.WORD	1	
2551	011474	004537	005134			JSR	RS,PRINT	;PRINT EXPECTED OCTAL
2552	011500	011754				.WORD	EXP	
2553	011502	100001				.WORD	100001	
2554	011504	004537	005134		95\$:	JSR	RS,PRINT	;SPACE THE PAPER
2555	011510	013301				.WORD	CRLF1	
2556	011512	000000				.WORD	0	
2557	011514	022737	001566	000042	97\$:	CMP	#SENDAD,2#42	;UNDER ACT?
2558	011522	001001				BNE	99\$;BR IF NO
2559	011524	000000				HALT		;HALT ON ACT ERROR
2560								
2561	011526	000205			99\$:	RTS	RS	;RETURN FROM MODULE

2562
2563
2564
2565
2566
2567 011530 000000
2568 011532 000000
2569 011534 000000
2570 011536 000000
2571 011540 000000
2572 011542 000000
2573 011544 000000
2574 011546 000000
2575 011550 000000
2576 011552 000000
2577 011554 000000
2578 011556 000000
2579 011560 000000
2580
2581 011562 000000
2582 011564 000000
2583 011566 000000
2584 011570 000000
2585 011572 000000
2586 011574 000000
2587 011576 000000
2588

```

:
:
: 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 ; NPB 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

```


2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611

011610
011610 000001
011612 000004
011614 160000
011616 177756
011620 000040
011622 000776
011624 000000
011626 000006

.....

INPUT PARAMETER LIMITS TABLE

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

ACTIVE PARAMETER TABLE

THIS TABLE CONTAINS CONFIGURATION PARAMETER CURRENTLY IN USE.

2612
2613
2614
2615
2616
2617
2618 011630
2619 011630 000000
2620 011632 000000
2621 011634 000000
2622 011636 000000
2623 011640 000000
2624 011642 000000
2625 011644 000000
2626 011646 000000
2627 011650 000000
2628 011652 177564
2629 011654 177566
2630 011656 177560
2631 011660 177562
2632 011662 000000
2633 011664 000000
2634 011666 000001
2635 011670 000000
2636 011672 000000
2637 011674 000000
2638 011676 000000
2639 011700 000000
2640 011702 000000
2641 011704 000000
2642 011706 000000
2643 011710 000000
2644 011712 000000
2645 011714 000000
2646 011716 000000
2647 011720 000000
2648 011722 000000
2649 011724 000000
2650 011726 000400
2651 011730 000000
2652 011732 000000
2653 011734 000000
2654 011736 000000
2655 011740 000000
2656 011742 000000
2657 011744 000000
2658 011746 000000
2659 011750 000000
2660 011752 000000
2661 011754 000000
2662 011756 000000
2663 011760 000000

APAR: .WORD 0
ATESTN: .WORD 0
AUBA: .WORD 0
AVECT: .WORD 0
APROR: .WORD 0
MSIZE: .WORD 0
MMAX: .WORD 0
EASIZE: .WORD 0
KT: .WORD 0
FIRSTP: .WORD 0
TPS: .WORD 177564
TPB: .WORD 177566
TKS: .WORD 177560
TDB: .WORD 177562
CTESTN: .WORD 0
PARMV: .WORD 0
COUNT: .WORD 1
CNT: .WORD 0
ERR: .WORD 0
LOCK: .WORD 0
LOCKAD: .WORD 0
EABITS: .WORD 0
PHYMAX: .WORD 0
BUFBAS: .WORD 0
BUFZSE: .WORD 0
BUFMAX: .WORD 0
EACUR: .WORD 0
DATAD: .WORD 0
BFBAPH: .WORD 0
BFAOPH: .WORD 0
SPWPHY: .WORD 0
DEV: .WORD 0
CMD: .WORD 400
ERRPC: .WORD 0
ERRID: .WORD 0
ERRMS: .WORD 0
EAERR: .WORD 0
RECORD: .WORD 0
FRSTD: .WORD 0
FRSTDV: .WORD 0
DEVCT: .WORD 0
DEVMSS: .WORD 0
ACTU: .WORD 0
EXP: .WORD 0
DEVFND: .WORD 0
SPWMVD: .WORD 0

: STARTING TEST NO.
: DX UNIBUS BASE ADDR.
: DX VECTOR ADDRESS
: DX PRIORITY LEVEL
: ADDR OF LAST WORD OF MEM, NOT INCLUDING LOADE' . T OFF
: LAST WORD OF MEM, KT OFF
: LOWEST INVALID SETTING OF PAR
: =0 NO KT IN USE =1 KT IN USE
: 0 IF FIRST PASS, 1 IF NOT FIRST PASS
: TTY PRINTER STATUS REG ADDR
: TTY PRINTER BUFFER REG ADDR
: TTY KEYB D STATUS REG ADDR
: TTY KEYB D BUFFER.
: CURRENT TEST NUMBER
: 0 IF PAR'S NOT VALID, 1 IF THEY ARE VALID
: ITERATION COUNT CONSTANT
: ITERATION VARIABLE
: IF ERROR DETECTED, CONTAINS TN
: SW-TN IF LOOP SW 14 ERR DETECTED
: LOCK ON ERROR ADDR
: HIGHEST VALID SETTING OF ADDR 17 16
: LOWEST INVALID PHYSICAL ADDR, LOW 16 BITS
: START OF CURRENT WORKING BUFFER, VIRTUAL
: SIZE OF CURRENT BUFFER IN OCT BYTES
: MAX VIRTUAL ADDR FOR THIS KT SETTING
: CURRENT SETTING OF EA BITS
: CURRENT DATA ADDR
: CURRENT PHYSICAL BUFFER BASE ADDR
: CURRENT PHYSICAL DATA ADDR
: PHYSICAL ADDR OF SPW START
: CURRENT IBM DEVICE NO.
: CURRENT IBM COMMAND WITH PARITY BIT
: PC WHERE ERROR WAS DETECTED
: ERROR NUMBER
: ADDR OF UNIQUE ERROR MESSAGE
: EABITS ON ERROR
: SWITCH TO CONTROL DEV REC RECORDING
: FIRST DEVICE RECOGNIZED SW
: FIRST DEVICE RECOGNIZED NUMBER
: FIRST DEVICE GROUP COUNT
: DEVICE MESSAGE PRINTED SW
: ACTUAL DATA
: EXPECTED DATA
: = 1 IF DEVICE EVER RECOGNIZED
: = 1 IF SPW TABLE IN ALTERNATE LOCATION

```

2664
2665 011762 ERR1: .NLIST BIN
2666 011770 .ASCIZ /FAST NPR DATA ERROR/
2667 011776
2668 012004
2669 012006 ERR2: .ASCIZ /NO DX INTERRUPT/
2670 012014
2671 012022
2672
2673 012026 ERR3: .ASCIZ /DX ADDRESSING ERROR, DATA XFER/
2674 012034
2675 012042
2676 012050
2677 012056
2678 012064
2679 012065 ERR4: .ASCIZ /SPW ACCESS, ADDR ERROR/
2680 012072
2681 012100
2682 012106
2683 012114 ERR5: .ASCIZ /NO IBM DEV ADR RECOGNIZED. CHECK A20 MODULE/
2684 012122
2685 012130
2686 012136
2687 012144
2688 012152
2689 012160
2690 012166
2691 012170 ERR6: .ASCIZ /DST ACCESS ERROR/
2692 012176
2693 012204
2694 012211 T4MSG: .ASCIZ /TST3 MUST BE RUN FIRST FOR TST 4 TO WORK/
2695 012216
2696 012224
2697 012232
2698 012240
2699 012246
2700 012254
2701 012262 HEADER: .ASCII 'MD-11-DZDXJA DX11B MEMORY ADDRESSING TEST'<15><12>
2702 012270
2703 012276
2704 012304
2705 012312
2706 012320
2707 012326
2708 012334
2709 012336 .ASCII 'TYPE: <D>, DEFAULT PARAMETERS'<15><12>
2710 012344
2711 012352
2712 012360
2713 012366
2714 012374
2715 012375 .ASCII '
2716 012402 <P>, PREVIOUS PARAMETERS'<15><12>
2717 012410
2718 012416
2719 012424

```

2720	012430	.ASCII	'	<S>, SELECT PARAMETERS'<15><12>
2721	012436			
2722	012444			
2723	012452			
2724	012460			
2725	012461	.ASCIZ	'	<N>, START AT THIS TEST NO.'<15><12>
2726	012466			
2727	012474			
2728	012502			
2729	012510			
2730	012516			
2731	012520	INVM:	.ASCIZ	'INVALID ENTRY'
2732	012526			
2733	012534			
2734	012536	INVPRM:	.ASCIZ	'NEED MORE PARMS'
2735	012544			
2736	012552			
2737	012556	PROMPT:	.ASCIZ	'D,P,S,N? '
2738	012564			
2739	012570	TESTN:	.ASCIZ	'STARTING TEST NO. '
2740	012576			
2741	012604			
2742	012612			
2743	012613	BASEA:	.ASCIZ	'BASE ADDRESS '
2744	012620			
2745	012626			
2746	012631	VECTA:	.ASCIZ	'VECTOR ADDRESS '
2747	012636			
2748	012644			
2749	012651	PRILV:	.ASCIZ	'DX PRIORITY LEVEL '
2750	012656			
2751	012664			
2752	012672			
2753	012674	SETSW:	.ASCIZ	'SET SWITCHES '
2754	012702			
2755	012710			
2756	012712	BELL:	.ASCIZ	<207> 'END PASS'
2757	012720			
2758	012724	BAOP:	.ASCIZ	'INPUT PARAMETER OUTSIDE LIMITS'
2759	012732			
2760	012740			
2761	012746			
2762	012754			
2763	012762			
2764	012763	ERPC:	.ASCIZ	'ERROR PC: '
2765	012770			
2766	012776	ERNO:	.ASCIZ	' ERR NO.: '
2767	013004			
2768	013012	ERTN:	.ASCIZ	'TEST NO.: '
2769	013020			
2770	013025	ERADR:	.ASCIZ	'VIRT ADDR OF BUFFER BASE: '
2771	013032			
2772	013040			
2773	013046			
2774	013054			
2775	013060	ERADPH:	.ASCIZ	'PHY ADDR OF BUFFER: '

```

2776 013066
2777 013074
2778 013102
2779 013105 EABIT: .ASCIZ 'EA BITS IN OCT: '
2780 013112
2781 013120
2782 013126 DATA: .ASCIZ 'VIRT ADDR OF ACCESS ERROR: '
2783 013134
2784 013142
2785 013150
2786 013156
2787 013162 DEVMS: .ASCIZ '1ST IBM DEV RCGNZD. OCTAL: '
2788 013170
2789 013176
2790 013204
2791 013212
2792 013220 DEVCMS: .ASCIZ 'SIZE OF 1ST DEV GROUP: '
2793 013226
2794 013234
2795 013242
2796 013250
2797 013251 ACTUMS: .ASCIZ 'ACTUAL: '
2798 013256
2799 013263 EXPMS: .ASCIZ ' EXPECTED: '
2800 013270
2801 013276
2802 013301 CRLF1: .ASCIZ / /<15><12>
2803
2804 013306 .LIST BIN
2805 .EVEN
2806 013306 012 INBUF: .BYTE 12 ;INBUF MUST BE ON AN EVEN WORD BOUNDARY
2807 014000 014000 .=14000 ;KEYBOARD INPUT BUFFER
2808
2809 014000 SPW: .START OF STATIONARY SPW TABLE USED ONLY TO
2810 014000 000002 .WORD 2 ;FIELD UNEXPECTED ERRORS.
2811 014002 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2812 014004 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2813 014006 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2814 014010 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2815 014012 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2816 014014 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2817 014016 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2818 014020 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2819 014022 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2820 014024 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2821 014026 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2822 014030 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2823 014032 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2824 014034 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2825 014036 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2826 014040 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2827 014042 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2828 014044 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2829 014046 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2830 014050 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS
2831 014052 000002 .WORD ;STATUS OF UNIT CHECK WITH NO DST ACCESS

```


3000	014574	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3001	014576	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3002	014600	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3003	014602	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3004	014604	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3005	014606	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3006	014610	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3007	014612	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3008	014614	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3009	014616	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3010	014620	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3011	014622	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3012	014624	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3013	014626	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3014	014630	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3015	014632	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3016	014634	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3017	014636	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3018	014640	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3019	014642	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3020	014644	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3021	014646	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3022	014650	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3023	014652	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3024	014654	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3025	014656	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3026	014660	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3027	014662	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3028	014664	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3029	014666	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3030	014670	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3031	014672	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3032	014674	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3033	014676	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3034	014700	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3035	014702	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3036	014704	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3037	014706	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3038	014710	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3039	014712	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3040	014714	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3041	014716	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3042	014720	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3043	014722	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3044	014724	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3045	014726	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3046	014730	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3047	014732	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3048	014734	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3049	014736	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3050	014740	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3051	014742	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3052	014744	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3053	014746	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3054	014750	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS
3055	014752	000002	.WORD	STATUS	OF	UNIT	CHECK	WITH	NO	DST	ACCESS

N06

MD-11-DZDXJ-A DX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 78
CROSS REFERENCE TABLE -- USER SYMBOLS

. = 016000	613#	617#	626	627#	629#	631#	632#	652#	686	715	1047	1051#	1052
	1053	1290#	1291	1292	1299#	1353	1650	1827	1851	2804#	2807#		

CKSMR	444#																				
COMMEN	569#																				
ENDCOM	569#																				
ERROR	463#																				
ESCAPE	569#																				
GETPRI	569#	1428																			
GETSMR	444#	569#																			
MULT	569#																				
NEWTST	569#																				
POP	446#	569#	1038	1342	1575	1836	1837														
PUSH	446#	569#	1316	1558	1817	1823															
REPORT	569#																				
SCOPE	464#																				
SETPRI	445#	569#	755	1219	1923	1967	2450														
SETTRA	1766#	1791	1800	1801	1802	1804	1806	1807	1808	1809											
SETUP	569#	709																			
SKIP	569#																				
SLASH	569#																				
SPACE	569#																				
STARS	569#	624	648	686	1046	1122	1137	1208	1232	1302	1406	1582	1653	1741	1813						
	1824																				
SMRSU	569#	722#																			
TRNTRP	1788#																				
TYPBIN	569#																				
TYPDEC	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#																				
TYPNUM	569#	1150																			
TYPNUM	569#																				
SSCHRE	633#																				
SSCHTH	640#																				
SSESCA	569#																				
SSNEWT	569#																				
SSSET	1772#	1791	1800	1801	1802	1804	1806	1807	1808	1809											
SSSKIP	569#																				
.EQUAT	445#	459																			
.HEADE	444#	447																			
.KT11	446#	569																			
.SETUP	444#	709																			
.SACT1	446#	622																			
.SCATC	444#	611																			
.SCHTR	444#	633																			
.SPOWE	445#	1811																			
.SROOC	445#	1300																			
.SREAD	445#	1044																			
.SSIZE	446#	1404																			
.STRAP	445#	1739																			
.STYPE	444#	1580																			
.STYPO	445#	1651																			

. ABS. 016000 000

ERRORS DETECTED: 0

C07

MD-11-DZDXJ-A OX11-B ADDRESSING TEST
DZDXJA.P11 28-APR-77 09:59

MACY11 27(1006) 02-MAY-77 10:36 PAGE 81
CROSS REFERENCE TABLE -- MACRO NAMES

DEFAULT GLOBALS GENERATED: 0

DZDXJA DZDXJA/CRF/SOL=DZDXJA.P11(400,3274),SYSMAC.SML(400,1066)

RUN-TIME: 13 7 .7 SECONDS
RUN-TIME RATIO: 749/21=34.2
CORE USED: 20K (40 PAGES)