

PDP-11

UNIBUS SYS EXER
CZKUAE0

AH-8856E-MC

COPYRIGHT 75-80

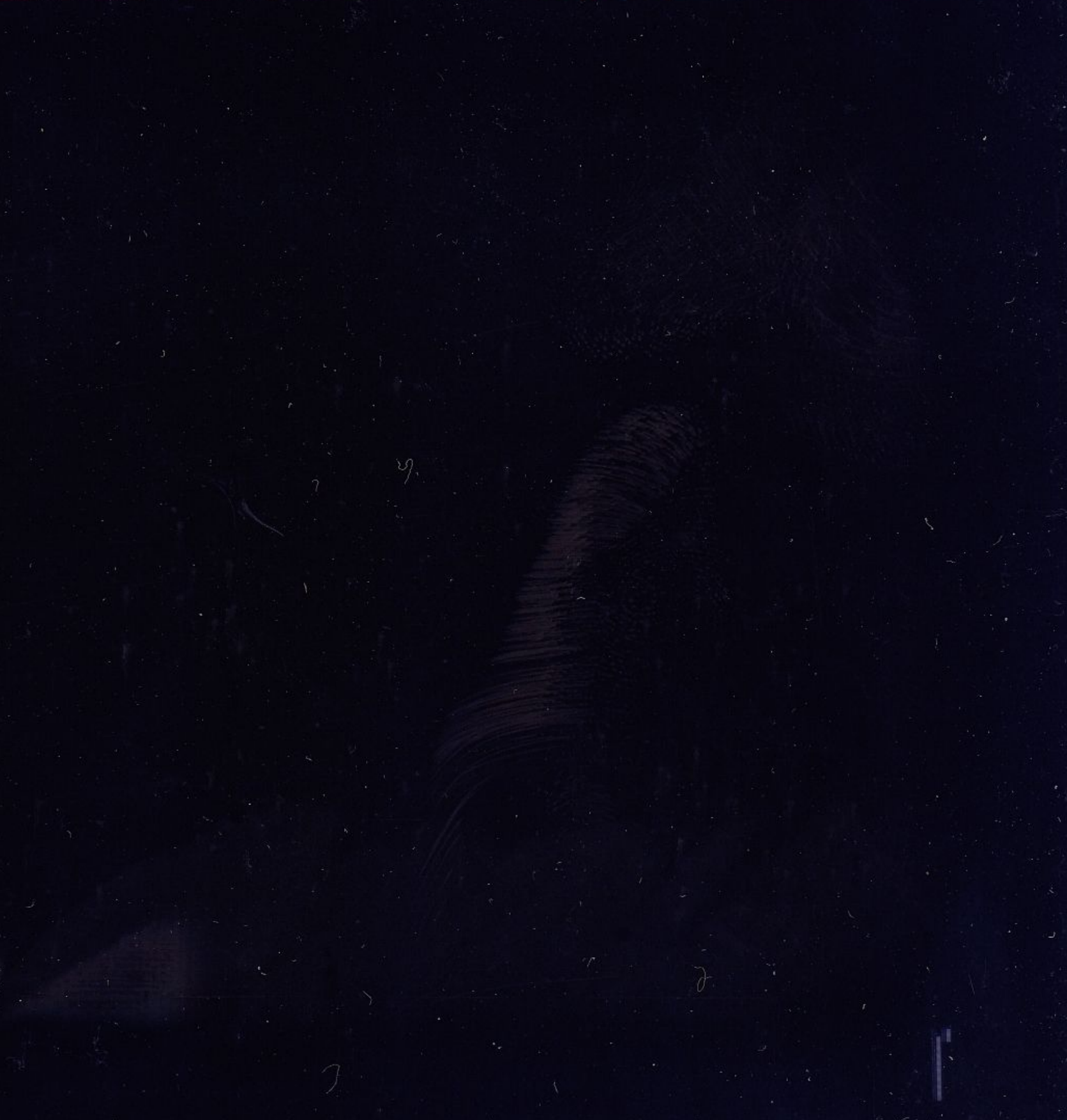
FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

The left side of the page contains a grid of 60 small, illegible diagrams or tables arranged in 10 rows and 6 columns. Each cell in the grid appears to contain a small schematic or data table, but the text is too small to read. The diagrams are arranged in a regular grid pattern, with some cells appearing slightly darker than others, possibly due to the scanning process or the original document's layout.



Identification

SEQ 0001

Product Code: AC-8855E-MC
Product Name: CZKUAEO Unibus Systems Exerciser Diagnostic
DATE: NOV 79
Maintainer: Diagnostic Group
Author: Manuel Soares
MODIFIED BY: BILL SCHLITZKUS

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright (C) 1975, 1979 Digital Equipment Corporation

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECJS	DECTAPE	

Table of Contents

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	Hardware
2.2	Software
3.0	PROGRAM DESCRIPTION
3.1	Switch Options
3.2	Test 1 thru Test 16
3.3	Sysmac Routines
4.0	ERROR REPORTING

1.0 ABSTRACT

This program was created to test PDP-11's CPU interface circuitry. It uses the Unibus Exerciser(s) (UBE) to insure proper operation by simulating peripherals which would require the 11-CPU to produce the necessary signals. It should be noted that the UBE is a powerful tool and if it is not programmed correctly could cause various problems on the Bus.

2.0 REQUIREMENTS

2.1 Hardware

This program assumes the following in proper working condition: 1. The Unibus, 2. Memory (8K minimum), and 3. UBE(s) (4 maximum). If a fourth UBE is being used, its time delay should be set at 100us to prevent latency problems in one of the tests.

With two or more UBE(s), all should have W1 jumpers except the one furthest electrically from the CPU. If there are more than 4 UBE(s) on the Unibus the program is not responsible for any problems which might occur, since it is programmed to handle a maximum of only 4.

2.2 Software

After loading the program the starting address must be 200, so that the first time through, the available UBE(s) are determined. In addition if one or more UBE(s) are added or removed, the program again must be started at 200. Otherwise, to avoid duplicating some printouts, the program can be restarted at address 220.

A SOFTWARE HALT CAN BE CAUSED BY DEPRESSING CONTROL-H ON THE CONSOLE.
IF THE PROGRAM IS HALTED THIS WAY, AND THE PROGRAM IS RESTARTED,
DEPRESS ANY CONSOLE KEY TO REMOVE THE SOFTWARE HALT CONDITION.

3.0 PROGRAM DESCRIPTION

This program was assembled with MACY11 using PDP-11 Maindec Sysmac package .

3.1 Switch Options

The use of this program on processors having a software switch register necessitates operator interaction: the operator must set up location 176 with the switch register values desired.

Switch -----	Use ---
15	Halt on Error
14	Loop on test
13	Inhibit error typeouts
11	Inhibit iterations
10	Bell on error
9	Loop on error
8	Loop on test in SWR<5:0>

NOTE: If you wish to inhibit all typing except "end of pass" you must put down switch 7, after loading 200.
6 WHEN SET, INHIBIT TEST 14

3.2 Test 1 through Test 16

- TEST 1 - No Bus grants issued with processor at higher priority than bus request. This test is to insure that any request is not honored as long as the processor is at the same or higher priority.
- TEST 2 - Issuing of non-processor grants and arbitration tests. This test will request on NPR through BR4 levels with the processor status initially at level 7 and make sure the device exercises an NPG to do a fun 1-dati, then the requests will be repeated while sequentially lowering the processor status from 7 to 0 to allow arbitration of all requests and the issuing of NPG.
- TEST 3 - Issuing of Bus grant 7 and arbitration tests. This test will arbitrate for a BG7. The requests will be on levels BR7 thru BR4, doing fun 1-dati transfers, and the processor status lowered sequentially from 7 to 0.
- TEST 4 - Issuing of Bus grant 6 and arbitration tests. This test will arbitrate for a BG6, the requests will be on levels BR6 thru BR4, doing fun 1-dati transfers, and the processor status lowered sequentially from 6 to 0.
- TEST 5 - Issuing of Bus grant 5 and arbitration tests. This test will arbitrate for a BG5, the requests will be on levels BR5 thru BR4, doing fun 1-dati transfers, and the processor status lowered sequentially from 5 to 0.

- TEST 6 - Issuing of Bus grant 4 and arbitration tests. This test will arbitrate for a BG4, the requests will be on level BR4, doing func 1-dati transfers, and the processor status lowered sequentially from 4 to 0.
- TEST 7 - CPU test for no sack time out. This test will check that the CPU times out and drops a grant if no sack signal is received. If the CPU time out is inoperative, the Bus exerciser will time out and send the sack signal to prevent a Bus hang and set an error flag in CR2.
- TEST 10 - CPU test for receiving sack. This test is to insure that the CPU can receive the sack signal; The time delay will be set on device 1 and several dati transfers made. If there is not bus late error, the CPU received sack correctly. It is assumed that dev 1 time delay is set for 10us.
- TEST 11 - Passing of grants and interrupt test. This test will set off all available devices simultaneously whose only functions will be to interrupt, the requests will all be at level 7 so that the device closest to the CPU should receive BG7 first and interrupt first, the next closest should interrupt next and so on.
- TEST 12 - Address lines (14 - 17) check. This test will check Bus address lines 14 thru 17 by doing a fun 1-dati-npr to those addresses. If the addresses don't exist the interrupt routine will ignore any no ssyn error.
- TEST 13 - CPU test for ACLO/DCLO sequence. This test checks the assertion of ACLO and DCLO and that the CPU traps to the correct service routine. If this program is running under ACT11 this test will be skipped.
- TEST 14 - Parity error test. This test will cause parity error and checks that the CPU traps to the correct vector.
THIS TEST IS SKIPPED ON MACHINES THAT DON'T HAVE THE SXT INSTRUCTION (EG., 1/05 AND 11/20).
THIS TEST SHOULD BE DESELECTED IF THE MEMORY PARITY OPTION IS NOT PRESENT OR NOT ENABLED.
- SW06=1 INHIBIT TEST 14
- TEST 15 - Multitransfers I. This test will cause any Bus exercisers, up to 4, to create a lot of traffic on the Bus and check that the CPU can handle it; all devices are set off simultaneously.
- TEST 16 - Multitransfers II. This test will have the available exercisers doing various transfers and/or interrupts at different request levels to further check CPU handling capabilities.

TEST 17 - DUMMY END OF PROGRAM. This portion of the program is just to see if '^H' has been typed on the console to cause a program halt. If there is no '^H' the program continues by jumping to \$EOP (end-of-pass routine).
IF THE PROGRAM IS HALTED THIS WAY, AND THE PROGRAM IS RESTARTED, DEPRESS ANY CONSOLE KEY TO REMOVE THE SOFTWARE HALT CONDITION.

3.3 Sysmac Routines

The 'END OF PASS ROUTINE' thru 'Power Down and Up Routines', as listed in the program listing, are the Sysmac package macros. They are called out in the source program, some with arguments and some without, and are expanded in the listing. Some macros are necessary for the operation of others, so for a complete explanation of all available Sysmac Macros see PDP-11 Maindec Sysmac Package (DZQAC-B-D).

4.0 ERROR REPORTING

The minimum amount of information given when an error occurs is the PC of the error call and the Test number in which it occurred. Other pertinent data will be typed out depending on the test being run at that time.

17	OPERATIONAL SWITCH SETTINGS
29	BASIC DEFINITIONS
139	MEMORY MANAGEMENT DEFINITIONS
196	TRAP CATCHER
226	ACT11 HOOKS
237	COMMON TAGS
295	ERROR POINTER TABLE
528	INITIALIZE THE COMMON TAGS
713	T1 NO BUS GRANTS ISSUED WITH PROCESSOR AT HIGHER PRIORITY THAN BUS REQUEST
744	T2 ISSUING OF NON-PROCESSOR GRANTS AND ARBITRATION TESTS
782	T3 ISSUING OF BUS GRANT 7 AND ARBITRATION TESTS
818	T4 ISSUING OF BUS GRANT 6 AND ARBITRATION TESTS
854	T5 ISSUING OF BUS GRANT 5 AND ARBITRATION TESTS
890	T6 ISSUING OF BUS GRANT 4 AND ARBITRATION TESTS
926	T7 CPU TEST FOR NO SACK TIME OUT
966	T10 CPU TEST FOR RECEIVING SACK
1007	T11 PASSING OF GRANTS AND INTERRUPT TEST
1080	T12 ADDRESS LINES (14 - 17) CHECK
1123	T13 CPU TEST FOR ACLO/DCLO SEQUENCE
1165	T14 PARITY ERROR TEST
1217	T15 MULTITRANSFERS I
1280	T16 MULTITRANSFERS II
1406	T17 DUMMY END OF PROGRAM
1803	END OF PASS ROUTINE
1852	SCOPE HANDLER ROUTINE
1917	ERROR HANDLER ROUTINE
1962	ERROR MESSAGE TIMEOUT ROUTINE
2009	TTY INPUT ROUTINE
2083	ROUTINE TO SIZE MEMORY
2197	SAVE AND RESTORE R0-R5 ROUTINES
2242	TYPE ROUTINE
2331	BINARY TO OCTAL (ASCII) AND TYPE
2408	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2475	TRAP DECODER
2498	TRAP TABLE
2517	POWER DOWN AND UP ROUTINES

1

```

2          167400          $SWR=167400
3          000300          $SWRMK=300
4                                     .TITLE UNIBUS EXERCISER
5                                     : *COPYRIGHT (C) SEPT 79
6                                     : *DIGITAL EQUIPMENT CORP.
7                                     : *MAYNARD, MASS. 01754
8                                     : *
9                                     : *PROGRAM BY DIAG. ENG.
10                                    : *
11                                    : *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
12                                    : *PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
13                                    : *
14          000001        $TN-1
15          .SBTTL OPERATIONAL SWITCH SETTINGS
16          : *
17          : *          SWITCH          USE
18          : *          -----          -----
19          : *          15          HALT ON ERROR
20          : *          14          LOOP ON TEST
21          : *          13          INHIBIT ERROR TYPEOUTS
22          : *          11          INHIBIT ITERATIONS
23          : *          10          BELL ON ERROR
24          : *          9          LOOP ON ERROR
25          : *          8          LOOP ON TEST IN SWR<5:0>
26          : *          6          WHEN SET, INHIBIT TEST 14
27          .SBTTL BASIC DEFINITIONS
28          : *
29          : *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
30          001100        STACK= 1100
31          .EQUIV EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
32          .EQUIV IOT,SCOPE          ;;BASIC DEFINITION OF SCOPE CALL
33          : *
34          : *MISCELLANEOUS DEFINITIONS
35          000011        HT= 11          ;;CODE FOR HORIZONTAL TAB
36          000012        LF= 12          ;;CODE FOR LINE FEED
37          000015        CR= 15          ;;CODE FOR CARRIAGE RETURN
38          000200        CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
39          177776        PS= 177776     ;;PROCESSOR STATUS WORD
40          .EQUIV PS,PSW
41          177774        STKLMT= 177774  ;;STACK LIMIT REGISTER
42          177772        PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
43          177570        DSWR= 177570   ;;HARDWARE SWITCH REGISTER
44          177570        DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
45          : *
46          : *GENERAL PURPOSE REGISTER DEFINITIONS
47          000000        R0= %0          ;;GENERAL REGISTER
48          000001        R1= %1          ;;GENERAL REGISTER
49          000002        R2= %2          ;;GENERAL REGISTER
50          000003        R3= %3          ;;GENERAL REGISTER
51          000004        R4= %4          ;;GENERAL REGISTER
52          000005        R5= %5          ;;GENERAL REGISTER
53          000006        R6= %6          ;;GENERAL REGISTER
54          000007        R7= %7          ;;GENERAL REGISTER
55          000006        SP= %6         ;;STACK POINTER
56          000007        PC= %7         ;;PROGRAM COUNTER
57

```

```
58          : *PRIORITY LEVEL DEFINITIONS
59          000000 PR0= 0          :: PRIORITY LEVEL 0
60          000040 PR1= 40         :: PRIORITY LEVEL 1
61          000100 PR2= 100        :: PRIORITY LEVEL 2
62          000140 PR3= 140        :: PRIORITY LEVEL 3
63          000200 PR4= 200        :: PRIORITY LEVEL 4
64          000240 PR5= 240        :: PRIORITY LEVEL 5
65          000300 PR6= 300        :: PRIORITY LEVEL 6
66          000340 PR7= 340        :: PRIORITY LEVEL 7
67
68          : *'SWITCH REGISTER' SWITCH DEFINITIONS
69          100000 SW15= 100000
70          040000 SW14= 40000
71          020000 SW13= 20000
72          010000 SW12= 10000
73          004000 SW11= 4000
74          002000 SW10= 2000
75          001000 SW09= 1000
76          000400 SW08= 400
77          000200 SW07= 200
78          000100 SW06= 100
79          000040 SW05= 40
80          000020 SW04= 20
81          000010 SW03= 10
82          000004 SW02= 4
83          000002 SW01= 2
84          000001 SW00= 1
85          .EQUIV SW09,SW9
86          .EQUIV SW08,SW8
87          .EQUIV SW07,SW7
88          .EQUIV SW06,SW6
89          .EQUIV SW05,SW5
90          .EQUIV SW04,SW4
91          .EQUIV SW03,SW3
92          .EQUIV SW02,SW2
93          .EQUIV SW01,SW1
94          .EQUIV SW00,SW0
95
96          : *DATA BIT DEFINITIONS (BIT00 TO BIT15)
97          100000 BIT15= 100000
98          040000 BIT14= 40000
99          020000 BIT13= 20000
100         010000 BIT12= 10000
101         004000 BIT11= 4000
102         002000 BIT10= 2000
103         001000 BIT09= 1000
104         000400 BIT08= 400
105         000200 BIT07= 200
106         000100 BIT06= 100
107         000040 BIT05= 40
108         000020 BIT04= 20
109         000010 BIT03= 10
110         000004 BIT02= 4
111         000002 BIT01= 2
112         000001 BIT00= 1
113         .EQUIV BIT09,BIT9
```

```
114      .EQUIV BIT08,BIT8
115      .EQUIV BIT07,BIT7
116      .EQUIV BIT06,BIT6
117      .EQUIV BIT05,BIT5
118      .EQUIV BIT04,BIT4
119      .EQUIV BIT03,BIT3
120      .EQUIV BIT02,BIT2
121      .EQUIV BIT01,BIT1
122      .EQUIV BIT00,BIT0
123
124      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
125      000004  ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
126      000010  RESVEC= 10       ;;RESERVED AND ILLEGAL INSTRUCTIONS
127      000014  TBITVEC=14      ;;'T' BIT
128      000014  TRTVEC= 14       ;;TRACE TRAP
129      000014  BPTVEC= 14       ;;BREAKPOINT TRAP (BPT)
130      000020  IOTVEC= 20       ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
131      000024  PWRVEC= 24       ;;POWER FAIL
132      000030  EMTVEC= 30       ;;EMULATOR TRAP (EMT) **ERROR**
133      000034  TRAPVEC=34      ;;'TRAP' TRAP
134      000060  TKVEC= 60        ;;TTY KEYBOARD VECTOR
135      000064  TPVEC= 64        ;;TTY PRINTER VECTOR
136      000240  PIRQVEC=240     ;;PROGRAM INTERRUPT REQUEST VECTOR
137
138      .SBTTL MEMORY MANAGEMENT DEFINITIONS
139
140      ;*KT11 VECTOR ADDRESS
141      000250  MMVEC= 250
142
143      ;*KT11 STATUS REGISTER ADDRESSES
144
145      177572  SR0= 177572
146      177574  SR1= 177574
147      177576  SR2= 177576
148      172516  SR3= 172516
149
150      ;*USER "I" PAGE DESCRIPTOR REGISTERS
151
152      177600  JIPDR0= 177600
153      177602  UIPDR1= 177602
154      177604  UIPDR2= 177604
155      177606  UIPDR3= 177606
156      177610  UIPDR4= 177610
157      177612  UIPDR5= 177612
158      177614  UIPDR6= 177614
159      177616  UIPDR7= 177616
160
161      ;*USER "I" PAGE ADDRESS REGISTERS
162
163      177640  UIPAR0= 177640
164      177642  UIPAR1= 177642
165      177644  UIPAR2= 177644
166      177646  UIPAR3= 177646
167      177650  UIPAR4= 177650
168      177652  UIPAR5= 177652
169      177654  UIPAR6= 177654
```


226
227
228
229
230
231
232
233
234

000232
000046
015376
000052
000052 040000
000232

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


```
235 .SBTTL COMMON TAGS
236
237 ::*****
238 ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
239 ::*USED IN THE PROGRAM.
240
241 001100 .=1100
242 001100 $CMTAG: .WORD 0 :: START OF COMMON TAGS
243 001100 000000 $PASS: .WORD 0 :: CONTAINS PASS COUNT
244 001102 000 $STSTM: .BYTE 0 :: CONTAINS THE TEST NUMBER
245 001103 000 $ERFLG: .BYTE 0 :: CONTAINS ERROR FLAG
246 001104 000000 $ICNT: .WORD 0 :: CONTAINS SUBTEST ITERATION COUNT
247 001106 000000 $LPADR: .WORD 0 :: CONTAINS SCOPE LOOP ADDRESS
248 001110 000000 $LPERR: .WORD 0 :: CONTAINS SCOPE RETURN FOR ERRORS
249 001112 000000 $ERTTL: .WORD 0 :: CONTAINS TOTAL ERRORS DETECTED
250 001114 000 $ITEMB: .BYTE 0 :: CONTAINS ITEM CONTROL BYTE
251 001115 001 $ERMAX: .BYTE 1 :: CONTAINS MAX. ERRORS PER TEST
252 001116 000000 $ERRPC: .WORD 0 :: CONTAINS PC OF LAST ERROR INSTRUCTION
253 001120 000000 $GDADR: .WORD 0 :: CONTAINS ADDRESS OF 'GOOD' DATA
254 001122 000000 $BDADR: .WORD 0 :: CONTAINS ADDRESS OF 'BAD' DATA
255 001124 000000 $GDDAT: .WORD 0 :: CONTAINS 'GOOD' DATA
256 001126 000000 $BDDAT: .WORD 0 :: CONTAINS 'BAD' DATA
257 001130 000000 .WORD 0 :: RESERVED--NOT TO BE USED
258 001132 000000 .WORD 0
259 001134 000 $AUTOB: .BYTE 0 :: AUTOMATIC MODE INDICATOR
260 001135 000 $INTAG: .BYTE 0 :: INTERRUPT MODE INDICATOR
261 001136 000000 .WORD 0
262 001140 177570 $SWR: .WORD DSWR :: ADDRESS OF SWITCH REGISTER
263 001142 177570 DISPLAY: .WORD DDISP :: ADDRESS OF DISPLAY REGISTER
264 001144 177560 $TKS: 177560 :: TTY KBD STATUS
265 001146 177562 $TKB: 177562 :: TTY KBD BUFFER
266 001150 177564 $TPS: 177564 :: TTY PRINTER STATUS REG. ADDRESS
267 001152 177566 $TPB: 177566 :: TTY PRINTER BUFFER REG. ADDRESS
268 001154 000 $NULL: .BYTE 0 :: CONTAINS NULL CHARACTER FOR FILLS
269 001155 002 $FILLS: .BYTE 2 :: CONTAINS # OF FILLER CHARACTERS REQUIRED
270 001156 012 $FILLC: .BYTE 12 :: INSERT FILL CHARS. AFTER A 'LINE FFED'
271 001157 000 $TPFLG: .BYTE 0 :: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
272 001160 000000 $REGAD: .WORD 0 :: CONTAINS THE ADDRESS FROM
273 :: WHICH ($REGO) WAS OBTAINED
274 001162 000000 $REG0: .WORD 0 :: CONTAINS (($REGAD)+0)
275 001164 000000 $REG1: .WORD 0 :: CONTAINS (($REGAD)+2)
276 001166 000000 $REG2: .WORD 0 :: CONTAINS (($REGAD)+4)
277 001170 000000 $REG3: .WORD 0 :: CONTAINS (($REGAD)+6)
278 001172 000000 $REG4: .WORD 0 :: CONTAINS (($REGAD)+10)
279 001174 000000 $REG5: .WORD 0 :: CONTAINS (($REGAD)+12)
280 001176 000000 $TMP0: .WORD 0 :: USER DEFINED
281 001200 000000 $TMP1: .WORD 0 :: USER DEFINED
282 001202 000000 $TMP2: .WORD 0 :: USER DEFINED
283 001204 000000 $TMP3: .WORD 0 :: USER DEFINED
284 001206 000000 $TMP4: .WORD 0 :: USER DEFINED
285 001210 000000 $TMP5: .WORD 0 :: USER DEFINED
286 001212 000000 $TIMES: 0 :: MAX. NUMBER OF ITERATIONS
287 001214 000000 $ESCAPE: 0 :: ESCAPE ON ERROR ADDRESS
288 001216 177607 000377 $BELL: .ASCII <207><377><377> :: CODE FOR BELL
289 001222 077 $QUES: .ASCII /?/ :: QUESTION MARK
290 001223 015 $CRLF: .ASCII <15> :: CARRIAGE RETURN
```

291 001224 000012
292

\$LF: .ASCIZ <12> ;:LINE FEED
;:*****

293
294
295
296
297
298
299
300
301
302
303
304
305
306
307 001226
308
309
310
311 001226 011144
312 001230 011212
313 001232 015022
314 001234 000000
315
316 001236 011243
317
318 001240 011356
319 001242 015032
320 001244 000000
321
322 001246 011446
323 001250 011500
324 001252 015052
325 001254 000000
326
327 001256 011561
328 001260 011500
329 001262 015052
330 001264 000000
331
332 001266 011617
333 001270 011500
334 001272 015052
335 001274 000000
336
337 001276 011655
338 001300 011500
339 001302 015052
340 001304 000000
341
342 001306 011713
343 001310 011500
344 001312 015052
345 001314 000000
346
347 001316 011751
348 001320 012017

```
.SBTTL ERROR POINTER TABLE

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

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

$ERRTB:
:*****
:*****
:ITEM 1
      EM1      :CPU TRAPPED THRU LOC 4 -TIME OUT
      DH1      : ADDR $ERRPC #ERR/TST#
      DT1      :$REG2,$ERRPC,$STSTM,0
      0
:ITEM 2
      EM2      :CPU ISSUED A BUS GRANT WITH PSW = 7
      DH2      :DEV 1 SHOULD NOT HAVE BECOME BUS MASTER
      DT2      :BE1DB BE1CC BE1BA BE1A PSW $ERRPC #ERR/TST#
      0      :$REG0,$REG1,$REG2,$REG3,$REG4,$ERRPC,$STSTM,0
:ITEM 3
      EM3      :CPU DID NOT ISSUE A BUS NPG
      DH3      :BE1CR1 BE1CC FM-PS TO-PS $ERRPC #ERR/TST#
      DT3      :$REG0,$REG1,$REG2,$REG3,$ERRPC,$STSTM,0
      0
:ITEM 4
      EM4      :CPU DID NOT ISSUE BUS GRANT 7
      DH3
      DT3
      0
:ITEM 5
      EM5      :CPU DID NOT ISSUE BUS GRANT 6
      DH3
      DT3
      0
:ITEM 6
      EM6      :CPU DID NOT ISSUE BUS GRANT 5
      DH3
      DT3
      0
:ITEM 7
      EM7      :CPU DID NOT ISSUE BUS GRANT 4
      DH3
      DT3
      0
:ITEM 10
      EM10     :ONE OR MORE DEVICES DID NOT INTERRUPT
      DH10     :THIS IS THE ORDER IN WHICH THEY INTERRUPTED
```

349				: 1ST 2ND 3RD 4TH \$ERRPC #ERR/TST#
350	001322	015070	DT10	;\$REG1,\$REG2,\$REG3,\$REG4,\$ERRPC,\$STSTNM,0
351	001324	000000	0	
352			:ITEM 11	
353	001326	012155	EM11	:BUS ADDRESS LINES <A17:A14> DID NOT FUNCTION PROPERLY
354	001330	012243	DH11	:BE1CR1 BE1CR2 BE1BA \$ERRPC #ERR/TST#
355	001332	015106	DT11	;\$REG1,\$REG2,\$REG3,\$ERRPC,\$STSTNM,0
356	001334	000000	0	
357			:ITEM 12	
358	001336	012314	EM12	:CPU NO SACK TIME OUT LOGIC FAILED(TO NEGATE BUS GRANT)
359	001340	012402	DH12	:BE1CR1 BE1CR2 \$ERRPC #ERR/TST#
360	001342	015122	DT12	;\$REG0,\$REG1,\$ERRPC,\$STSTNM,0
361	001344	000000	0	
362			:ITEM 13	
363	001346	012443	EM13	:CPU DID NOT PROPERLY EXECUTE AN ACLO/DCLO SEQUENCE
364	001350	012526	DH13	;\$ERRPC #ERR/TST#
365	001352	015134	DT13	;\$ERRPC,\$STSTNM,0
366	001354	000000	0	
367			:ITEM 14	
368	001356	012547	EM14	:CPU DID NOT TRAP FROM BUS PARITY ERR PA/PB 0/1
369	001360	012526	DH13	
370	001362	015134	DT13	
371	001364	000000	0	
372			:ITEM 15	
373	001366	012632	EM15	:DEV 1 DID DATIP WIT' ROL ON DATOB TO MEMORY
374				:THE TRANSFER TO THE FOLLOWING LOC WAS INCORRECT
375	001370	012775	DH15	:MEMORY ACTUAL CORRECT
376				: LOC DATA DATA \$ERRPC #ERR/TST# \$ICNT #
377	001372	015142	DT15	;\$REG0,\$REG1,\$REG3,\$ERRPC,\$STSTNM,\$ICNT,0
378	001374	000000	0	
379			:ITEM 16	
380	001376	013107	EM16	:DEV 3'S DATO TO MEMORY DID NOT EQUAL PATTERN IN R3
381	001400	012775	DH15	
382	001402	015142	DT15	
383	001404	000000	0	
384			:ITEM 17	
385	001406	013175	EM17	:DEV 4'S DATO TO MEMORY DID NOT EQUAL PATERN IN R4
386	001410	012775	DH15	
387	001412	015142	DT15	
388	001414	000000	0	
389			:ITEM 20	
390	001416	013263	EM20	:DEV 1 DID FUN 1-NPR-DATIP;INCORRECT PATTERN IN MEMORY
391	001420	012775	DH15	
392	001422	015142	DT15	
393	001424	000000	0	
394			:ITEM 21	
395	001426	013357	EM21	:DEV 2 DID FUN 2-NPR-DATOB;INCORRECT PATTERN IN MEMORY
396	001430	012775	DH15	
397	001432	015142	DT15	
398	001434	000000	0	
399			:ITEM 22	
400	001436	013453	EM22	:BIT 7 OF CR2 SET-CPU DID NOT TIME OUT WITH SACK INHIBITED
401	001440	013545	DH22	:DEV # PC \$ERRPC #ERR/TST#
402	001442	015160	DT22	;\$TMP4,\$REG5,\$ERRPC,\$STSTNM,0
403	001444	000000	0	
404			:ITEM 23	

405	001446	013607	EM23	;BIT 11 OF CR2 SET-NO SSYN ON INTR SIGNAL
406	001450	013545	DH22	
407	001452	015160	DT22	
408	001454	000000	0	
409			:ITEM 24	
410	001456	013660	EM24	;BIT 5 OF CR2 SET-RECEIVED WRONG GRANT
411	001460	013545	DH22	
412	001462	015160	DT22	
413	001464	000000	0	
414			:ITEM 25	
415	001466	013726	EM25	;BIT 6 OF CR2 SET-BUS LATE
416	001470	013545	DH22	
417	001472	015160	DT22	
418	001474	000000	0	
419			:ITEM 26	
420	001476	013760	EM26	;BIT 8 OF CR2 SET-DEV DID NOT RECEIVE SSYN
421	001500	013545	DH22	
422	001502	015160	DT22	
423	001504	000000	0	
424			:ITEM 27	
425	001506	014022	EM27	;BIT 9 OF CR2 SET-WRONG ADDR ON BUS
426	001510	013545	DH22	
427	001512	015160	DT22	
428	001514	000000	0	
429			:ITEM 30	
430	001516	014071	EM30	;BIT 10 OF CR2 SET-DEV RECEIVED OTHER THAN ONE GRANT
431	001520	013545	DH22	
432	001522	015160	DT22	
433	001524	000000	0	
434			:ITEM 31	
435	001526	014160	EM31	;BKGRND RTN INSTRUCTIONS OF NEGB'S WERE NOT DONE
436				;CORRECTLY TO \$REG1 DURING MULTITRANFERS II
437	001530	014320	DH31	;ACTUAL CORRECT
438				;DATA DATA \$ERRPC #ERR/TST# \$ICNT #
439	001532	015172	DT31	; \$REG1,146463,\$ERRPC,\$TSTNM,\$ICNT,0
440	001534	000000	0	
441			:ITEM 32	
442	001536	014413	EM32	;DEV 3 DID DATI BUT HAS INCORRECT
443				;VALUES IN DATA REGISTER
444	001540	014320	DH31	
445	001542	015172	DT31	
446	001544	000000	0	
447			:ITEM 33	
448	001546	014477	EM33	;DEV 4 DID NOT INTR THE CORRECT # OF TIMES
449	001550	014320	DH31	
450	001552	015172	DT31	
451	001554	000000	0	
452			:ITEM 34	
453	001556	014551	EM34	;LAST DATI XFER BY DEV 1 WAS INCORRECT-
454				;EITHER DEV DID NOT WORK OR WRONG DATA WASSET UP
455	001560	014320	DH31	
456	001562	015172	DT31	
457	001564	000000	0	
458			:ITEM 35	
459	001566	014725	EM35	;CPU TRAPPED THRU LOC 0 TO CATCH
460				;IMPROPERLY LOADED VECTORS

461 001570 011212
462 001572 015022
463 001574 000000
464
465
466
467 001576 007740
468 001600 170014
469 000114
470 000116
471 001602 000000
472 001604 000000
473 001606 000000
474 001610 000000
475 001612 000000
476 001614 000000
477 001616 000000
478 001620 000000
479 001622 000000
480 001624 000000
481 001626 000000
482 001630 000000
483 001632 000000
484 001634 000000
485 001636 000000
486 001640 000000
487 001642 000000
488 001644 000000
489 001646 000000
490 001650 000000
491 001652 000000
492 001654 000000
493 001656 000000
494 001660 000000
495 001662 000000
496 001664 000000
497 001666 000000
498 001670 000000
499 001672 000000
500 001674 000000
501 001676 000000
502 001700 000000
503 001702 000000
504 001704 000000 000000 000000
505 001712 000000
506 001714 000000
507 001716 000000
508 001720 000000
509 001722 000000
510 001724 000000
511
512
513 001726
514 001726 012703 001602
515 001732 005023
516 001734 022703 001724

DH1 : ADDR \$ERRPC #ERR/TST#
DT1 : \$REG2,\$ERRPC,\$TSTNM,0
0
:*****
:*****
:*****
ALLERR :7740 :ALL ERR BITS OF CR2
SIMLGO :170014 :ADDR TO SET OFF ALL DEVICES SIMOLTANEOUSLY
PBVEC =114 :TRAP VEC FOR PARITY ERROR
PBPSW =116 :PSW ADDR FOR TRAP ON PARITY ERR
BE1DB :0 :DATA REG ADDR FOR DEVICE 1
BE1CC :0 :CYCLE COUNT REG ADDR FOR DEV 1
BE1BA :0 :ADDR REG ADDR FOR DEV 1
BE1CR1 :0 :CONTROL REG 1 ADDR FOR DEV 1
BE1CLR :0 :CLEAR ERRS REG ADDR FOR DEV 1
BE1CR2 :0 :CONTROL REG 2 ADDR FOR DEV 1
BE2DB :0 :DATA REG ADDR FOR DEV 2
BE2CC :0 :CYCLE COUNT REG ADDR FOR DEV 2
BE2BA :0 :ADDR REG ADDR FOR DEV 2
BE2CR1 :0 :CONTROL REG 1 ADDR FOR DEV 2
BE2CLR :0 :CLEAR ERRS REG ADDR FOR DEV 2
BE2CR2 :0 :CONTROL REG 2 ADDR FOR DEV 2
BE3DB :0 :DATA REG ADDR FOR DEV 3
BE3CC :0 :CYCLE COUNT REG ADDR FOR DEV 3
BE3BA :0 :ADDR REG ADDR FOR DEV 3
BE3CR1 :0 :CONTROL REG 1 ADDR FOR DEV 3
BE3CLR :0 :CLEAR ERRS REG ADDR FOR DEV 3
BE3CR2 :0 :CONTROL REG 2 ADDR FOR DEV 3
BE4DB :0 :DATA REG ADDR FOR DEV 4
BE4CC :0 :CYCLE COUNT REG ADDR FOR DEV 4
BE4BA :0 :ADDR REG ADDR FOR DEV 4
BE4CR1 :0 :CONTROL REG 1 ADDR FOR DEV 4
BE4CLR :0 :CLEAR ERRS REG ADDR FOR DEV 4
BE4CR2 :0 :CONTROL REG 2 ADDR FOR DEV 4
BE1VEC :0 :TRAP VEC ADDR FOR DEV 1
BE1PSW :0 :PSW ADDR FOR TRAP THRU BE1VEC
BE2VEC :0 :TRAP VEC ADDR FOR DEV 2
BE2PSW :0 :PSW ADDR FOR TRAP THRU BE2VEC
BE3VEC :0 :TRAP VEC ADDR FOR DEV 3
BE3PSW :0 :PSW ADDR FOR TRAP THRU BE3VEC
BE4VEC :0 :TRAP VEC ADDR FOR DEV 4
BE4PSW :0 :PSW ADDR FOR TRAP THRU BE4VEC
DEVCNT :0 :CONTAINS # OF DEVICES ON BUS
DEVS :0,0,0,0 :WILL CONTAIN ADDR(S) OF INTR'G DEVS
DATA1 :0 :MAX ADDR TO WHICH DATA XFERRD BY DEV 1
DATA2 :0 :MAX ADDR TO WHICH DATA XFERRD BY DEV 2
DATA3 :0 :MAX ADDR TO WHICH DATA XFERRD BY DEV 3
DATA4 :0 :MAX ADDR TO WHICH DATA XFERRD BY DEV 4
ENDMEM :0 :TAG ENDING DEFINED LABELS
:*****
:*****
CLRRTN:
1S: MOV #BE1DB,R3 :R3 IS POINTER TO BUFFER AREAS
CLR (R3)+ :CLEAR BUFFER THEN INCREMENT ADDR
CMP #ENDMEM,R3 :IF POINTER AT LAST BUFFER, EXIT


```

517 001740 100374          BPL      1$          ;IF PLUS, GO BACK AND CLEAR NEXT ADDR
518 001742 012703 001162    MOV      #$REG0,R3    ;NOW START TO CLEAR TEMP REGISTERS
519 001746 005023          CLR      (R3)+        ;CLEAR CURRENT ADDR
520 001750 022703 001210    2$:     CMP      #$TMP5,R3 ;CHECK FOR LAST TEMP REG ADDR
521 001754 101374          BHI     2$           ;IF NOT, CLEAR NEXT TEMP REG
522 001756 000207          RTS      PC          ;EXIT
523                                     ;:*****
524                                     ;:*****
525 001760          START:
526          .SBTTL  INITIALIZE THE COMMON TAGS
527          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
528 001760 012706 001100    MOV      #$CMTAG,R6   ;;FIRST LOCATION TO BE CLEARED
529 001764 005026          CLR      (R6)+        ;;CLEAR MEMORY LOCATION
530 001766 022706 001140    CMP      #SWR,R6     ;;DONE?
531 001772 001374          BNE     .-6          ;;LOOP BACK IF NO
532 001774 012706 001100    MOV      #STACK,SP   ;;SETUP THE STACK POINTER
533          ;;INITIALIZE A FEW VECTORS
534 002000 012737 015416 000020  MOV      #$SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
535 002006 012737 000340 000022  MOV      #340,@IOTVEC+2 ;;LEVEL 7
536 002014 012737 015674 000030  MOV      #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
537 002022 012737 000340 000032  MOV      #340,@EMTVEC+2 ;;LEVEL 7
538 002030 012737 020252 000034  MOV      #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
539 002036 012737 000340 000036  MOV      #340,@TRAPVEC+2 ;;LEVEL 7
540 002044 012737 020332 000024  MOV      #SPURDN,@PWRVEC ;;POWER FAILURE VECTOR
541 002052 012737 000340 000026  MOV      #340,@PWRVEC+2 ;;LEVEL 7
542 002060 013737 015242 015234  MOV      $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
543 002066 005037 001212          CLR      $TIMES      ;;INITIALIZE NUMBER OF ITERATIONS
544 002072 005037 001214          CLR      $ESCAPE     ;;CLEAR THE ESCAPE ON ERROR ADDRESS
545 002076 112737 000001 001115  MOVB    #1,$ERMAX    ;;ALLOW ONE ERROR PER TEST
546 002104 012737 002104 001106  MOV      #.,$LPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
547 002112 012737 002112 001110  MOV      #.,$LPERR    ;;SETUP THE ERROR LOOP ADDRESS
548          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
549          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
550 002120 013746 000004          MOV      @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
551 002124 012737 002160 000004  MOV      #64$,@ERRVEC ;;SET UP ERROR VECTOR
552 002132 012737 177570 001140  MOV      #DSWR,SWR    ;;SETUP FOR A HARDWARE SWICH REGISTER
553 002140 012737 177570 001142  MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
554 002146 022777 177777 176764  CMP      #-1,@SWR    ;;TRY TO REFERENCE HARDWARE SWR
555 002154 001012          BNE     66$         ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
556          ;;AND THE HARDWARE SWR IS NOT - -1
557 002156 000403          BR      65$         ;;BRANCH IF NO TIMEOUT
558 002160 012716 002166          64$:   MOV      #65$, (SP) ;;SET UP FOR TRAP RETURN
559 002164 000002          RTJ
560 002166 012737 000176 001140  65$:   MOV      #SWREG,SWR  ;;POINT TO SOFTWARE SWR
561 002174 012737 000174 001142  MOV      #DISPREG,DISPLAY
562 002202 012637 000004          66$:   MOV      (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
563
564 002206 032777 000200 176724  BIT      #BIT07,@SWR  ;;IS SWITCH 7 UP?
565 002214 001402          BEQ     3$          ;IF NOT, SKIP TIMEOUT
566 002216 104401 010754          TYPE    ,QNO
567 002222
568 002222 022737 000777 001176  3$:     CMP      #777,$TMP0  ;IS THIS RESTART FROM LOC 220?
569 002230 001002          BNE     5$          ;IF NOT,SKIP THE JMP INSTR
570 002232 000137 003252          JMP     @TST1      ;ELSE JUMP TO TEST 1
571
572 002236          5$:

```

```

573 002236 012737 010342 000000      MOV      #THRU0,0      ;SET UP FOR TRAP THRU LOC 0
574 002244 012737 000340 000002      MOV      #PR7,2       ;SET UP PSW FOR TRAP THRU 0
575 002252 032777 000200 176660      BIT      #BIT07,@SWR  ;IS SWITCH 7 UP?
576 002260 001400                BEQ      33$          ;IF NOT, SKIP TYPEOUT
577 002262                33$:
578 002262 004737 001726                JSR      PC,CLRRTN    ;CLEAR BUFFER AREAS
579 002266 012737 000340 000006      MOV      #PR7,ERRVEC+2 ;PS=7 FOR TRAP THRU LOC 4
580 002274 012700 170000                MOV      #170000,R0   ;SET UP POINTER FOR 1ST POSSIBLE DEV ADDR
581 002300 012702 000510                MOV      #510,R2      ;SET UP POINTER FOR 1ST POSSIBLE VEC ADDR
582 002304 012701 001602                MOV      #BE1DB,R1    ;SET UP POINTER FOR DEVICE ADDR LOCATION
583 002310 012703 001662                MOV      #BE1VEC,R3   ;SET UP POINTER FOR INTR ADDR LOCATION
584 002314                LODDEV:
585 002314 022700 170060                CMP      #170060,R0   ;IS R0 > LAST POSSIBLE DEV ADDR?
586 002320 002002                BGE      10$          ;IF NOT,GO TO 10$
587 002322 000137 002624                JMP      BGIN         ;ELSE GO TO BGIN
588 002326                10$:
589 002326 012737 002432 000004      MOV      #NODEV,ERRVEC ;SET UP TRAP VECTOR FOR TIME OUT
590 002334 005710                TST      (R0)         ;SEE IF ACTUAL DEVICE ADDRESS EXISTS
591 002336 012737 002550 000004      MOV      #TYMOUT,ERRVEC ;CHANGE TRAP VECTOR FOR ERROR CONDITION
592 002344 005237 001702                INC      DEVCNT       ;COUNT DEVICES
593 002350 010021                MOVREG: MOV      R0,(R1)+ ;MOVE ACTUAL DEVICE ADDR TO DEVICE NAME
594 002352 010037 001174                MOV      R0,$REG5     ;REG5 CONTAINS LAST DEVICE ADDR
595 002356 062700 000002                ADD      #2,R0         ;INCREMENT POINTER BY 2
596 002362 105237 001176                INCB    $TMP0         ;COUNT # OF REGISTERS PER DEVICE
597 002366 122737 000005 001176      CMPB    #5,$TMP0     ;AFTER 5 REGISTERS
598 002374 001365                BNE     MOVREG        ;ARE RECORDED
599 002376 105037 001176                CLRB    $TMP0        ;CLEAR THE COUNTING REGISTER
600 002402 062700 000004                ADD      #4,R0         ;ADD 4 TO THE POINTER THEN
601 002406 010021                MOV      R0,(R1)+     ;RECORD THE LAST REGISTER ADDRESS
602 002410 062700 000002                ADD      #2,R0         ;INCREMENT POINTER BY 2
603 002414                MOVVEC:
604 002414 010223                MOV      R2,(R3)+     ;NOW START RECORDING
605 002416 062702 000002                ADD      #2,R2         ;THE INTR VECTORS
606 002422 010223                MOV      R2,(R3)+     ;INCREMENT POINTER BY 2
607 002424 062702 000002                ADD      #2,R2         ;THE INTR VECTORS
608 002430 000731                BR       LODDEV       ;INCREMENT POINTER BY 2
609                                ;AND GO SEE IF THER'S ANOTHER DEVICE
610                                ;*****
611                                ;*****
612 002432 022700 170060                NODEV: CMP      #170060,R0 ;TIME OUT ROUTINE FOR DEVICE CHECK
613 002436 003035                BGT      ADD20        ;IF ALL POSSIBLE ADDR'S HAVE NOT BEEN CHECKED
614 002440 012716 002624                MOV      #BGIN,(SP)   ;OUT-GO BACK AND CHECK FOR MORE,
615 002444 022737 000000 001702      CMP      #0,DEVCNT    ;ELSE CHANGE STACK POINTER
616 002452 001035                BNE     EXNO          ;CHECK FOR NO EXERCISERS
617 002454 104401 002462                TYPE    ,65$         ;IF ONE OR MORE EXERCISERS, EXIT
618 002460 000423                BR       ,64$         ;:TYPE ASCIZ STRING
619                                ;:GET OVER THE ASCIZ
620 002530                ;:65$: .ASCIZ <15><12>/THERE ARE NO EXERCISERS ON THE BUS/
621 002530 000000                64$:
622 002532 062700 000020                ADD20: HALT
623 002536 062702 000004                ADD      #20,R0       ;ADD 20 TO POINTER
624 002542 012716 002314                ADD      #4,R2        ;POINTER=NEXT DEV'S VEC LOCATIONS
625 002546                EXNO: MOV      #LODDEV,(SP) ;GO BACK TO LODDEV
626 002546 000002                RTI
627                                ;EXIT
628                                ;*****

```

```

629 002550 TYMOUT: ;TIME OUT ROUTINE
630
631 002550 011637 001166 MOV (SP), $REG2 ;THE MOVE IS FOR TYPEOUT REASONS
632 002554 162737 000002 001166 SUB #2, $REG2 ;SUBTRACT 2 TO FIND ACTUAL ADDR
633 002562 104001 ERROR 1 ;ERR MESSG FOR ILLEGAL TIME OUT
634 002564 000002 RTI
635
636 ;*****
648 002614 020237 001702 ;ADD 1 TO R2(DEVICE COUNTER)
649 002620 101766 CMP R2, DEVCNT ;SEE IF IT = PREVIOUS COUNT
650 002622 000207 BLOS 1$ ;IF NOT, CLEAR NEXT DEV REGS
651 ;EXIT
652
653 ;////////////////////////////////////
654 002624 BGIN:
655 002624 012737 010122 000024 MOV #PWRFAL, PWRVEC ;TAKE CARE OF BIT 4(S) BEING SET RANDOMLY IN CR2(S)
656 002632 004737 010020 JSR PC, STVEC ;SET UP VEC(S) FOR RANDOM ERRS
657 002636 004737 002566 JSR PC, CLRREG ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
658 002642 005037 001176 CLR $TMP0 ;CLEAR TEMPORARY REG
659 002646 005037 001162 CLR $REG0 ;CLEAR COUNTER
660 002652 032777 000200 176260 BIT #BIT07, @SWR ;IS SWITCH 7 UP?
661 002660 001002 BNE 2$ ;IF UP, GO TO 2$
662 002662 000137 003252 JMP 5$ ;ELSE SKIP THE TYPEOUTS
663 002666
664 002666 104401 002674 2$:
665 002672 000431 TYPE ,65$ ;:TYPE ASCIZ STRING
666 BR 64$ ;:GET OVER THE ASCIZ
667 002756 ;:65$: .ASCIZ <15><12>/THE FOLLOWING # OF EXERCISERS ARE ON THE BUS: /
668 002756 013746 001702 64$:
669 002762 104403 MOV DEVCNT, -(SP) ;:SAVE DEVCNT FOR TYPEOUT
670 002764 001 TYPOS ;:GO TYPE--OCTAL ASCII
671 002765 000 .BYTE 1 ;:TYPE 1 DIGIT(S)
672 002766 104401 002774 .BYTE 0 ;:SUPPRESS LEADING ZEROS
673 002772 000436 TYPE ,67$ ;:TYPE ASCIZ STRING
674 ;:67$: .ASCIZ <15><12>/THE LOWEST ELECT. PRIORITY UBE SHOULD NOT HAVE W1 JUMPER/
675 003070 66$:
676 003070 104401 003076 TYPE ,69$ ;:TYPE ASCIZ STRING
677 003074 000415 BR 68$ ;:GET OVER THE ASCIZ
678 ;:69$: .ASCIZ <15><12>/DEVICE ADDRESS(ES): /<15><12>
679 003130 68$:
680 003130 005037 001176 CLR $TMP0 ;:CLEAR TMP0(USED AS COUNTER)
681 003134 012700 001602 MOV #BE1DB, R0 ;:USE R0 AS POINTER TO ADDRESSES
682 003140
683 003140 005237 001176 4$:
684 003144 011037 001162 INC $TMP0 ;:ADD 1 TO TMP0
MOV (R0), $REG0 ;:MOVE FOR TYPEOUT REASONS

```

```

685 003150 104401 003156          TYPE      71$          ;;TYPE ASCIZ STRING
686 003154 000403          BR        70$          ;;GET OVER THE ASCIZ
687          ;;71$: .ASCIZ / DEV/
688 003164          70$:
689 003164 013746 001176          MOV      $TMP0,-(SP)    ;;SAVE $TMP0 FOR TYPEOUT
690 003170 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
691 003172      002          .BYTE      2          ;;TYPE 2 DIGIT(S)
692 003173      000          .BYTE      0          ;;SUPPRESS LEADING ZEROS
693 003174 104401 003202          TYPE      73$          ;;TYPE ASCIZ STRING
694 003200 000402          BR        72$          ;;GET OVER THE ASCIZ
695          ;;73$: .ASCIZ / = /
696 003206          72$:
697 003206 013746 001162          MOV      $REG0,-(SP)    ;;SAVE $REG0 FOR TYPEOUT
698 003212 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
699 003214      006          .BYTE      6          ;;TYPE 6 DIGIT(S)
700 003215      000          .BYTE      0          ;;SUPPRESS LEADING ZEROS
701 003216 062700 000014          ADD      #14,R0        ;;ADD 14 FOR NEXT ADDR
702 003222 023737 001176 001702          CMP      $TMP0,DEVcnt  ;;SEE IF TMP0 = # OF DEVICES
703 003230 001343          BNE      4$           ;;IF NOT, GO TYPE NEXT ADDR
704 003232 104401 001223          TYPE     $,SCRLF      ;;TYPE <CR><LF>
705 003236 022737 000004 001702          CMP      #4,DEVcnt    ;;SEE IF THERE ARE 4 DEVICES
706 003244 001002          BNE      5$           ;;IF NOT,SKIP THE TYPE OUT
707 003246 104401 011037          TYPE     ,FOR4        ;;ELSE TYPE MSSG FOR 4TH DEV
708 003252          5$:
709
710
711          ;;*****
712          ;*TEST 1          NO BUS GRANTS ISSUED WITH PROCESSOR AT HIGHER PRIORITY THAN BUS REQUEST
713          ;*THIS TEST IS TO INSURE THAT ANY REQUEST IS NOT
714          ;*HONORED AS LONG AS THE PROCESSOR IS AT THE SAME OR
715          ;*HIGHER PRIORITY
716          ;;*****
717 003252 000004          TEST1: SCOPE
718 003254 004737 002566          JSR      PC,CLRREG    ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
719 003260          NG:
720
721 003260 012777 004506 176374          MOV      #ERRCHK,@BE1VEC ;SET UP DEVICE 1 INTR VECTOR
722 003266 012777 000340 176370          MOV      #PR7,@BE1PSW  ;SET UP DEVICE 1 PSW VECTOR
723 003274 012737 002550 000004          MOV      #TYMOUT,ERRVEC ;SET UP TRAP THRU LOC 4(TIME OUT VEC)
724 003302 012700 000340          MOV      #PR7,R0       ;MOVE PS=7 TO R0
725 003306 012701 002021          MOV      #2021,R1      ;MOVE FUN 1-DATI-BR7 TO R1
726 003312 004737 010416          JSR      PC,NOG        ;DO NOG
727 003316 012700 000300          MOV      #PR6,R0       ;MOVE PS=6 TO R0
728 003322 012701 002011          MOV      #2011,R1      ;MOVE FUN 1-DATI-BR6 TO R1
729 003326 004737 010416          JSR      PC,NOG        ;DO NOG
730 003332 012700 000240          MOV      #PR5,R0       ;MOVE PS=5 TO R0
731 003336 012701 002005          MOV      #2005,R1      ;MOVE FUN 1-DATI-BR5 TO R1
732 003342 004737 010416          JSR      PC,NOG        ;DO NOG
733 003346 012700 000200          MOV      #PR4,R0       ;MOVE PS=4 TO R0
734 003352 012701 002003          MOV      #2003,R1      ;MOVE FUN 1-DATI-BR4 TO R1
735 003356 004737 010416          JSR      PC,NOG        ;DO NOG
736 003362 052777 004000 176220          BIS      #BIT11,@BE1CR1 ;SET BIT 11 TO DO FUN 3
737 003370 052777 000040 176212          BIS      #BIT05,@BE1CR1 ;SET OFF DEV AT NPR LEVEL
738 003376 000240          NOP                    ;ALLOW TIME FOR XFER
739
740
    
```

```

741
742
743
744
745
746
747
748
749
750 003400 000004
751
752 003402
753 003402 012700 000340
754 003406
755 003406 123737 001115 001103
756 003414 100452
757 003416 012737 000340 177776
758 003424 012777 004506 176230
759 003432 012777 000340 176224
760 003440 012777 020510 176140
761 003446 012777 177777 176130
762 003454 012777 002077 176126
763 003462 010037 177776
764 003466 000240
765
766 003470 022777 177777 176106
767 003476 001014
768 003500 017737 176104 001162
769 003506 017737 176072 001164
770 003514 012737 000340 001166
771 003522 010037 001170
772 003526 104003
773 003530
774 003530 162700 000040
775 003534 020027 000000
776 003540 100322
777
778
779
780
781
782
783
784
785 003542 000004
786 003544
787 003544 012700 000300
788 003550
789 003550 123737 001115 001103
790 003556 100452
791 003560 012737 000340 177776
792 003566 012777 004506 176066
793 003574 012777 000340 176062
794 003602 012777 020510 175776
795 003610 012777 177777 175766
796 003616 012777 002037 175764

```

```

*****
*TEST 2      ISSUING OF NON-PROCESSOR GRANTS AND ARBITRATION TESTS
*THIS TEST WILL REQUEST ON NPR THRU BR4 LEVELS
*WITH THE PROCESSOR STATUS INITIALLY AT LEVEL 7 AND MAKE
*SURE THE DEVICE EXERCISES AN NPG TO DO A FUN 1-DATI,
*THEN THE REQUESTS WILL BE REPEATED WHILE SEQUENTIALLY
*LOWERING THE PROCESSOR STATUS FROM 7 TO 0 TO ALLOW
*ARBITRATION OF ALL REQUESTS AND THE ISSUING OF NPG
*****
TST2:  SCOPE
NPRTST:
2$:  MOV      #PR7,R0
      CMPB    $ERMAX,$ERFLG ;MAX ERRS FOR THIS TEST OCCURRED?
      BMI     TST3          ;:BR IF YES TO NEXT TEST
      MOV     #PR7,PSW      ;INITIAL PS
      MOV     #ERRCHK,@BE1VEC ;SET UP VECTOR LOCATION
      MOV     #PR7,@BE1PSW  ;SET UP DEVICE INTR PSW
      MOV     #ATEND,@BE1BA  ;SET UP ADDR REG
      MOV     #-1,@BE1CC    ;SET CYCLE COUNT = 1
      MOV     #2077,@BE1CR1 ;LOAD #2077 FUNCTIONS
      MOV     R0,PSW        ;LOWER PROC STATUS
      NOP                    ;ALLOW TIME FOR INTERRUPT
      CMP     #-1,@BE1CC    ;SEE IF DEVICE WENT OFF
      BNE     5$           ;IF IT DID,SKIP ERR TYPEOUT
      MOV     @BE1CR1,$REG0 ;NEXT MOVES ARE FOR TYPEOUTS
      MOV     @BE1CC,$REG1
      MOV     #PR7,$REG2
      MOV     R0,$REG3
      ERROR   3            ;TYPE ERROR MESSG
5$:  SUB     #40,R0         ;LOWER PS BY 1 LEVEL
      CMP     R0,#PRO ;SEE IF R0 IS LESS THAN 0
      BPL    2$           ;IF PLUS ,GO BACK AND DO ANOTHER CYCLE
*****
*TEST 3      ISSUING OF BUS GRANT 7 AND ARBITRATION TESTS
*THIS TEST WILL ARBITRATE FOR A BG7,
*THE REQUESTS WILL BE ON LEVELS BR7 THRU BR4, DOING
*FUN 1-DATI TRANSFERS, AND THE PROCESSOR STATUS
*LOWERED SEQUENTIALLY FROM 7 TO 0.
*****
TST3:  SCOPE
BR7TST:
2$:  MOV     #PR6,R0        ;2ND PS WILL = 6
      CMPB    $ERMAX,$ERFLG ;MAX ERRS FOR THIS TEST OCCURRED?
      BMI     TST4          ;:BR IF YES TO NEXT TEST
      MOV     #PR7,PSW      ;INITIAL PS
      MOV     #ERRCHK,@BE1VEC ;SET UP VECTOR LOCATION
      MOV     #PR7,@BE1PSW  ;SET UP DEVICE INTR PSW
      MOV     #ATEND,@BE1BA  ;SET UP ADDR REG
      MOV     #-1,@BE1CC    ;SET CYCLE COUNT = 1
      MOV     #2037,@BE1CR1 ;LOAD #2037 FUNCTIONS

```

UNIBUS EXERCISER
CZKUA.E.P11 27-SEP-79

MACY11 30A(1052)
09:25

04-OCT-79 12:49 PAGE 19
T3 ISSUING OF BUS GRANT 7 AND ARBITRATION TESTS

SEQ 0025

```

797 003624 010037 177776      MOV    R0,PSW          ;LOWER PROC STATUS
798 003630 000240              NOP                    ;ALLOW TIME FOR INTERUPT
799
800 003632 022777 177777 175744  CMP    #-1,@BE1CC      ;SEE IF DEVICE WENT OFF
801 003640 001014              BNE    5$              ;IF IT DID,SKIP ERR TYPEOUT
802 003642 017737 175742 001162  MOV    @BE1CR1,$REG0   ;NEXT MOVES ARE FOR TYPEOUTS
803 003650 017737 175730 001164  MOV    @BE1CC,$REG1
804 003656 012737 000340 001166  MOV    #PR7,$REG2
805 003664 010037 001170      MOV    R0,$REG3
306 003670 104004              ERROR   4              ;TYPE ERROR MESSG
807 003672
808 003672 162700 000040      5$:  SUB    #40,R0          ;LOWER PS BY 1 LEVEL
809 003676 020027 000000      CMP    R0,#PRO ;SEE IF R0 IS LESS THAN 0
810 003702 100322      BPL    2$              ;IF PLUS ,GO BACK AND DO ANOTHER CYCLE

```

```

*****
:*TEST 4      ISSUING OF BUS GRANT 6 AND ARBITRATION TESTS
;*THIS TEST WILL ARBITRATE FOR A BG6,
;*THE REQUESTS WILL BE ON LEVELS BR6 THRU BR4, DOING
;*FUN 1-DATI TRANSFERS, AND THE PROCESSOR STATUS
;*LOWERED SEQUENTIALLY FROM 6 TO 0.
*****

```

```

820 003704 000004      TST4:  SCOPE
821 003706
822 003706 012700 000240      BR6TST: MOV    #PR5,R0          ;2ND PS WILL = 5
823 003712
824 003712 123737 001115 001103  2$:  CMPB   $ERMAX,$ERFLG  ;MAX ERRS FOR THIS TEST OCCURRED?
825 003720 100452              BMI    TST5            ;:BR IF YES TO NEXT TEST
826 003722 012737 000300 177776  MOV    #PR6,PSW        ;INITIAL PS
827 003730 012777 004506 175724  MOV    #ERRCHK,@BE1VEC ;SET UP VECTOR LOCATION
828 003736 012777 000340 175720  MOV    #PR7,@BE1PSW    ;SET UP DEVICE INTR PSW
829 003744 012777 020510 175634  MOV    #ATEND,@BE1BA   ;SET UP ADDR REG
830 003752 012777 177777 175624  MOV    #-1,@BE1CC      ;SET CYCLE COUNT = 1
831 003760 012777 002017 175622  MOV    #2017,@BE1CR1   ;LOAD #2017 FUNTIONS
832 003766 010037 177776      MOV    R0,PSW          ;LOWER PROC STATUS
833 003772 000240              NOP                    ;ALLOW TIME FOR INTERUPT
834

```

```

835 003774 022777 177777 175602  CMP    #-1,@BE1CC      ;SEE IF DEVICE WENT OFF
836 004002 001014              BNE    5$              ;IF IT DID,SKIP ERR TYPEOUT
837 004004 017737 175600 001162  MOV    @BE1CR1,$REG0   ;NEXT MOVES ARE FOR TYPEOUTS
838 004012 017737 175566 001164  MOV    @BE1CC,$REG1
839 004020 012737 000300 001166  MOV    #PR6,$REG2
840 004026 010037 001170      MOV    R0,$REG3
841 004032 104005              ERROR   5              ;TYPE ERROR MESSG
842 004034
843 004034 162700 000040      5$:  SUB    #40,R0          ;LOWER PS BY 1 LEVEL
844 004040 020027 000000      CMP    R0,#PRO ;SEE IF R0 IS LESS THAN 0
845 004044 100322      BPL    2$              ;IF PLUS ,GO BACK AND DO ANOTHER CYCLE

```

```

*****
:*TEST 5      ISSUING OF BUS GRANT 5 AND ARBITRATION TESTS
;*THIS TEST WILL ARBITRATE FOR A BG5,
;*THE REQUESTS WILL BE ON LEVELS BR5 THRU BR4, DOING
;*FUN 1-DATI TRANSFERS, AND THE PROCESSOR STATUS
*****

```

846
847
848
849
850
851
852


```

853
854
855 004046 000004
856 004050
857 004050 012700 000200
858 004054
859 004054 123737 001115 001103
860 004062 100452
861 004064 012737 000240 177776
862 004072 012777 004506 175562
863 004100 012777 000340 175556
864 004106 012777 020510 175472
865 004114 012777 177777 175462
866 004122 012777 002007 175460
867 004130 010037 177776
868 004134 000240
869
870 004136 022777 177777 175440
871 004144 001014
872 004146 017737 175436 001162
873 004154 017737 175424 001164
874 004162 012737 000240 001166
875 004170 010037 001170
876 004174 104006
877 004176
878 004176 162700 000040
879 004202 020027 000000
880 004206 100322
881
882
883
884
885
886
887
888
889
890 004210 000004
891 004212
892 004212 012700 000140
893 004216
894 004216 123737 001115 001103
895 004224 100452
896 004226 012737 000200 177776
897 004234 012777 004506 175420
898 004242 012777 000340 175414
899 004250 012777 020510 175330
900 004256 012777 177777 175320
901 004264 012777 002003 175316
902 004272 010037 177776
903 004276 000240
904
905 004300 022777 177777 175276
906 004306 001014
907 004310 017737 175274 001162
908 004316 017737 175262 001164

```

```

; *LOWERED SEQUENTIALLY FROM 5 TO 0.
;*****
TST5: SCOPE
BR5TST:
2$: MOV #PR4,R0 ;2ND PS WILL = 4
CMPB $ERMAX,$ERFLG ;MAX ERRS FOR THIS TEST OCCURRED?
BMI TST6 ;:BR IF YES TO NEXT TEST
MOV #PR5,PSW ;INITIAL PS
MOV #ERRCHK,@BE1VEC ;SET UP VECTOR LOCATION
MOV #PR7,@BE1PSW ;SET UP DEVICE INTR PSW
MOV #ATEND,@BE1BA ;SET UP ADDR REG
MOV #-1,@BE1CC ;SET CYCLE COUNT = 1
MOV #2007,@BE1CR1 ;LOAD #2007 FUNTIONS
MOV R0,PSW ;LOWER PROC STATUS
NOP ;ALLOW TIME FOR INTERUPT

CMP #-1,@BE1CC ;SEE IF DEVICE WENT OFF
BNE 5$ ;IF IT DID,SKIP ERR TYPEOUT
MOV @BE1CR1,$REG0 ;NEXT MOVES ARE FOR TYPEOUTS
MOV @BE1CC,$REG1
MOV #PR5,$REG2
MOV R0,$REG3
ERROR 6 ;TYPE ERROR MESSG

5$: SUB #40,R0 ;LOWER PS BY 1 LEVEL
CMP R0,#PRO ;SEE IF R0 IS LESS THAN 0
BPL 2$ ;IF PLUS ,GO BACK AND DO ANOTHER CYCLE

;*****
; *TEST 6 ISSUING OF BUS GRANT 4 AND ARBITRATION TESTS
; *THIS TEST WILL ARBITRATE FOR A BG4,
; *THE REQUESTS WILL BE ON LEVEL BR4, DOING
; *FUNC 1-DATI TRANSFERS, AND THE PROCESSOR ST TUS
; *LOWERED SEQUENTIALLY FROM 4 TO 0.
;*****
TST6: SCOPE
BR4TST:
2$: MOV #PR3,R0 ;2ND PS WILL = 3
CMPB $ERMAX,$ERFLG ;MAX ERRS FOR THIS TEST OCCURRED?
BMI TST7 ;:BR IF YES TO NEXT TEST
MOV #PR4,PSW ;INITIAL PS
MOV #ERRCHK,@BE1VEC ;SET UP VECTOR LOCATION
MOV #PR7,@BE1PSW ;SET UP DEVICE INTR PSW
MOV #ATEND,@BE1BA ;SET UP ADDR REG
MOV #-1,@BE1CC ;SET CYCLE COUNT = 1
MOV #2003,@BE1CR1 ;LOAD #2003 FUNTIONS
MOV R0,PSW ;LOWER PROC STATUS
NOP ;ALLOW TIME FOR INTERUPT

CMP #-1,@BE1CC ;SEE IF DEVICE WENT OFF
BNE 5$ ;IF IT DID,SKIP ERR TYPEOUT
MOV @BE1CR1,$REG0 ;NEXT MOVES ARE FOR TYPEOUTS
MOV @BE1CC,$REG1

```

909 004324 012737 000200 001166
 910 004332 010037 001170
 911 004336 104007
 912 004340
 913 004340 162700 000040
 914 004344 020027 000000
 915 004350 100322
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926 004352 000004
 927 004354 004737 002566
 928 004360 012777 177777 175216
 929 004366 012777 020510 175212
 930 004374 012737 000340 177776
 931 004402 012737 002550 000004
 932 004410 012777 004506 175244
 933 004416 012777 000340 175240
 934 004424 052777 000010 175162
 935 004432 012777 006003 175150
 936 004440 012737 000140 177776
 937 004446 004737 010626
 938 004452 042777 000010 175134
 939 004460 105777 175130
 940 004464 100024
 941 004466 017737 175116 001162
 942 004474 017737 175114 001164
 943 004502 104012
 944 004504 000414
 945
 946
 947 004506
 948 004506 033777 001576 175100
 949 004514 001407
 950 004516 011637 001174
 951 004522 012737 000001 001206
 952 004530 004737 010166
 953 004534
 954 004534 000002
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964

```

MOV #PR4,$REG2
MOV RO,$REG3
ERROR 7 ;TYPE ERROR MESSG
5$:
SUB #40,RO ;LOWER PS BY 1 LEVEL
CMP RO,#PRO ;SEE IF RO IS LESS THAN 0
BPL 2$ ;IF PLUS ,GO BACK AND DO ANOTHER CYCLE
;*****
;*****
;*TEST 7 CPU TEST FOR NO SACK TIME OUT
; *THIS TEST WILL CHECK THAT THE CPU TIMES OUT AND
; *DROPS A GRANT IF NO SACK SIGNAL IS RECEIVED
; *IF THE CPU TIME OUT IS INOPERATIVE, THE BUS EXERCISER
; *WILL TIME OUT AND SEND THE SACK SIGNAL TO PREVENT
; *A BUS HANG AND SET AN ERROR FLAG IN CR2
;*****
TST7: SCOPE
JSR PC,CLRREG ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
MOV #-1,@BE1CC ;SET CYCLE COUNT = 1
MOV #ATEND,@BE1BA ;SET UP DEVICE REG ADDR
MOV #PR7,PSW ;SET PS=7
MOV #TYMOUT,ERRVEC ;SET UP TIME OUT VECTOR
MOV #ERRCHK,@BE1VEC ;SET UP DEVICE INTR VECTOR
MOV #PR7,@BE1PSW ;SET UP DEVICE INTR PSW
BIS #BIT03,@BE1CR2 ;INHIBIT SACK RETURN
MOV #6003,@BE1CR1 ;DO FUN 3--BR4
MOV #PR3,PSW ;LOWER PROC. STATUS TO 3
JSR PC,CNTR ;DELAY FOR TIMEOUT
BIC #BIT03,@BE1CR2 ;ALLOW FUTURE SACKS
TSTB @BE1CR2 ;CHECK IF NO-NO SACK BIT IS SET
BPL TST10 ;:IF NOT SET, GO TO NEXT TEST
MOV @BE1CR1,$REG0 ;MOVE FOR TYPEOUT REASONS
MOV @BE1CR2,$REG1 ;MOVE FOR TYPEOUT
ERROR 12 ;ERROR IF SET-DEVICE FORCED TO SEND SACK
BR TST10 ;:GO TO NEXT TEST
;*****
;*****
ERRCHK:
BIT ALLERR,@BE1CR2 ;CHECK FOR ANY ERRS IN CR2
BEQ 5$ ;IF NONE, EXIT
MOV (SP),$REG5 ;FOR TYPEOUT OF PC
MOV #1,$TMP4 ;INDICATOR FOR DEVICE 1
JSR PC,ERRTN ;CHECK TO SEE IF ANY ERRORS OCCURED
5$:
RTI ;EXIT TRAP
;*****
;*TEST 10 CPU TEST FOR RECEIVING SACK
; *THIS TEST IS TO INSURE THAT THE CPU CAN RECEIVE THE
; *SACK SIGNAL; THE TIME DELAY WILL BE SET ON DEVICE 1
; *AND SEVERAL DATI TRANSFERS MADE, IF THERE IS NO BUS
; *LATE ERROR, THE CPU RECEIVED SACK CORRECTLY
; *IT IS ASSUMED THAT DEV 1 TIME DELAY IS SET FOR 10 US
;*****

```

```

965 004536 000004          TST10: SCOPE
966
967 004540 012737 000340 177776          MOV    #PR7,PSW          :PS = 7
968 004546 004737 002566          JSR    PC,CLRREG        :CLEAR ALL DEVICE REGISTERS
969 004552 012702 020510          MOV    #ATEND,R2        :R2 WILL POINT TO END OF PROG
970 004556 012705 000010          MOV    #10,R5           :R5 = # OF TEST WORDS TO CREATE
971 004562 004737 010610          JSR    PC,DOUP          :CREATE THOSE TEST WORDS
972
973 004566 012777 004506 175066          MOV    #ERRCHK,@BE1VEC  :SET UP VECTOR LOCATION
974 004574 012777 000340 175062          MOV    #PR7,@BE1PSW    :SET UP DEVICE INTR PSW
975 004602 012777 177770 174774          MOV    #-10,@BE1CC     :SET UP CYCLE COUNT
976 004610 012777 020510 174770          MOV    #ATEND,@BE1BA   :SET UP ADDR REGISTER
977 004616 052777 040000 174770          BIS    #BIT14,@BE1CR2  :SET BIT 14 OF CR2 FOR TIME DELAY
978 004624 012777 024441 174756          MOV    #24441,@BE1CR1 :DO FUN 2-DATIP/NO ROL-NPR
979 004632 012737 000000 177776          MOV    #PRO,PSW       :LOWER PS TO ALLOW INTERRUPTS
980 004640
981 004640 000240          5$: NOP                   :ALLOW FOR INTERUPT
982 004642 105777 174742          TSTB   @BE1CR1         :SEE IF DONE BIT SET
983 004646 100374          BPL    5$              :IF NOT, GO BACK AND WAIT
984 004650 042777 040000 174736          BIC    #BIT14,@BE1CR2  :ELSE CLEAR BIT 14 OF CR2
985 004656 022777 000010 174716          CMP    #10,@BE1DB      :DID LAST XFER MOVE 10 INTO DB
986 004664 001407          BEQ    10$            :IF IT DID,GO TO 10$
987 004666 017737 174710 001164          MOV    @BE1DB,$REG1    :ELSE MOVE FOR ERR TYPE OUT
988 004674 012737 000010 001166          MOV    #10,$REG2
989 004702 104034          ERROR  34             :TYPE ERR MSSG
990 004704
991 004704 032777 004000 174702          10$: BIT    #BIT11,@BE1CR2 :SEE IF NO Ssyn ON INTR ERR SET
992 004712 001402          BEQ    TST11          :;IF NOT SET, GO TO NEXT TEST
993 004714 104023          ERROR  23             :;ELSE TYPE ERR MSSG
994 004716 000400          BR     TST11          :;THEN GO TO NEXT TEST
995
996
997
998
999
1000
1001
1002
1003
1004
1005 004720 000004          :*****
1006 004722 012737 000340 177776          :*TEST 11  PASSING OF GRANTS AND INTERRUPT TEST
1007 004730 004737 002566          :*THIS TEST WILL SET OFF ALL AVAILABLE DEVICES SIMULTANEOUSLY
1008 004734          :*WHOSE ONLY FUNCTIONS WILL BE TO INTERRUPT, THE REQUESTS
1009 004734 012704 001704          :*WILL ALL BE AT LEVEL 7 SO THAT THE DEVICE CLOSEST TO THE CPU
1010 004740 012777 005162 174714          :*SHOULD RECEIVE BG7 FIRST AND INTERRUPT FIRST, THE NEXT
1011 004746 012777 000340 174710          :*CLOSEST SHOULD INTERRUPT NEXT AND SO ON.
1012 004754 012777 000036 174626          :*****
1013 004762 122737 000001 001702          TST11: SCOPE
1014 004770 001443          MOV    #PR7,PSW          :PS=7
1015 004772 012777 005200 174666          JSR    PC,CLRREG        :CLEAR CONTENTS OF ALL AVAILABLE DEVS
1016 005000 012777 000340 174662          LOAD1: MOV    #DEVS,R4      :DEVS CONTAINS SEQUENCE OF INTR'G DEVICE ADDRS
1017 005006 012777 000036 174610          MOV    #INTR1,@BE1VEC  :SET UP DEVICE 1 INTR VECTOR
1018 005014 122737 000002 001702          MOV    #PR7,@BE1PSW    :SET UP INTR PSW
1019 005022 001426          MOV    #36,@BE1CR1     :DO FUN 0 - BR7 THRU BR4
1020 005024 012777 005216 174640          CMPB   #1,DEV CNT      :IF ONLY 1 DEVICE ON BUS
          BEQ    GO           :BRANCH TO GO
          LOAD2: MOV    #INTR2,@BE2VEC :SET UP DEVICE 2 INTR VECTOR
          MOV    #PR7,@BE2PSW :SET UP DEVICE 2 PSW VECTOR
          MOV    #36,@BE2CR1  :DO FUN 0 - BR7 THRU BR4
          CMPB   #2,DEV CNT  :IF ONLY 2 DEVICES ON BUS
          BEQ    GO           :BRANCH TO GO
          LOAD3: MOV    #INTR3,@BE3VEC :SET UP DEVICE 3 INTR VECTOR

```

```

1021 005032 012777 000340 174634      MOV      #PR7,@BE3PSW      ;SET UP DEVICE 3 PSW VECTOR
1022 005040 012777 000036 174572      MOV      #36,@BE3CR1      ;DO FUN 0 - BR7 THRU BR4
1023 005046 122737 000003 001702      CMPB     #3,DEV CNT       ;IF ONLY 3 DEVICES ON BUS
1024 005054 001411                BEQ      GO               ;BRANCH TO GO
1025 005056                LOAD4:
1026 005056 012777 005234 174612      MOV      #INTR4,@BE4VEC   ;SET UP DEVICE 4 INTR VECTOR
1027 005064 012777 000340 174606      MOV      #PR7,@BE4PSW     ;SET UP DEVICE 4 PSW VECTOR
1028 005072 012777 000036 174554      MOV      #36,@BE4CR1     ;DO FUN 0 - BR7 THRU BR4
1029 005100                GO:
1030 005100 005001                CLR      R1               ;CLEAR R1 FOR COUNTING
1031 005102 005277 174472                INC      @SIMLGO          ;SET SIMULTANEOUS GO REGISTER
1032 005106 012737 000000 177776      MOV      #PRO,PSW        ;LOWER PS TO ALLOW INTERRUPTS
1033 005114 004737 010626                JSR     PC,CNTR          ;ALLOW TIME FOR INTERRUPTS BY COUNTING
1034 005120 020137 001702      CMPARE: CMP      R1,DEV CNT ;COMPARE THE TWO
1035 005124 001456                BEQ      TST12           ;:IF BUFFERS INCREMENTED IN CORRECT SEQUENCE, GO TO NEXT
1036 005126 013737 001704 001164      MOV      DEVS,$REG1      ;MOVE FOR TYPEOUT REASONS
1037 005134 013737 001706 001166      MOV      DEVS+2,$REG2    ;MOVE FOR TYPEOUT REASONS
1038 005142 013737 001710 001170      MOV      DEVS+4,$REG3    ;MOVE FOR TYPEOUT REASONS
1039 005150 013737 001712 001172      MOV      DEVS+6,$REG4    ;MOVE FOR TYPEOUT REASONS
1040 005156 104010                ERROR   10               ;TYPE ERR MSSG
1041 005160 000440                BR      TST12           ;:GO TO NEXT TEST
1042                ;:*****
1043                ;:*****
1044 005162                INTR1:
1045 005162 005201                INC      R1               ;ADD 1 TO COUNTER ON INTR
1046 005164 013724 001602                MOV      BE1DB,(R4)+     ;MOVE ADDR FOR TYPEOUT
1047 005170 012737 000001 001206      MOV      #1,$TMP4        ;INDICATOR FOR DEVICE 1
1048 005176 000424                BR      INTRTN          ;BRANCH TO REST OF INTR RTN
1049 005200                INTR2:
1050 005200 005201                INC      R1               ;ADD 1 TO COUNTER ON INTR
1051 005202 013724 001616                MOV      BE2DB,(R4)+     ;MOVE ADDR FOR TYPEOUT
1052 005206 012737 000002 001206      MOV      #2,$TMP4        ;INDICATOR FOR DEVICE 2
1053 005214 000415                BR      INTRTN          ;BRANCH TO REST OF INTR RTN
1054 005216                INTR3:
1055 005216 005201                INC      R1               ;ADD 1 TO COUNTER ON INTR
1056 005220 013724 001632                MOV      BE3DB,(R4)+     ;MOVE ADDR FOR TYPEOUT
1057 005224 012737 000003 001206      MOV      #3,$TMP4        ;INDICATOR FOR DEVICE 3
1058 005232 000406                BR      INTRTN          ;BRANCH TO REST OF INTR RTN
1059 005234                INTR4:
1060 005234 005201                INC      R1               ;ADD 1 TO COUNTER ON INTR
1061 005236 013724 001646                MOV      BE4DB,(R4)+     ;MOVE ADDR FOR TYPEOUT
1062 005242 012737 000004 001206      MOV      #4,$TMP4        ;INDICATOR FOR DEVICE 4
1063 005250                INTRTN:
1064 005250 011637 001174                MOV      (SP),$REG5      ;FOR TYPEOUT OF PC
1065 005254 004737 010166                JSR     PC,ERRTN        ;SEE IF ERROR CAUSED INTR
1066 005260 000002                RTI                      ;EXIT
1067
1068
1069                ;:*****
1070                ;*TEST 12 ADDRESS LINES (14 - 17) CHECK
1071                ;*THIS TEST WILL CHECK BUS ADDRESS LINES 14 THRU 17
1072                ;*BY DOING A FUN 1-DATI-NPR TO THOSE ADDRESSES
1073                ;*IF THE ADDRESSES DON'T EXIST THE INTERRUPT ROUTINE
1074                ;*WILL IGNORE ANY NO SSYN ERROR.
1075                ;:*****
1076 005262 000004      TST12: SCOPE

```

```

1077
1078 005264 004737 002566 JSR PC,CLRREG ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
1079 005270 012737 000140 177776 MOV #PR3,PSW ;PS=3
1080 005276 012777 005364 174356 MOV #BRK,@BE1VEC ;SET UP DEVICE INTR VEC
1081 005304 012777 000340 174352 MOV #PR7,@BE1PSW ;SET UP DEVICE PSW VEC
1082 005312
1083 005312 004737 010654 D014: JSR PC,ADLI ;TEST ADDR LINES 14 &15
1084 005316 052777 000001 174270 BIS #1,@BE1CR2 ;ELSE SET BIT 0 OF CR2(ADDR LINE 16)
1085 005324 004737 010654 JSR PC,ADLI
1086 005330 042777 000001 174256 BIC #1,@BE1CR2 ;CLEAR BIT 0(ADDR LINE 16)
1087 005336 052777 000002 174250 BIS #2,@BE1CR2 ;SET BIT 1 OF CR2(ADDR LINE 17)
1088 005344 004737 010654 JSR PC,ADLI
1089 005350 052777 000003 174236 BIS #3,@BE1CR2 ;ELSE SET BITS 0 AND 1 OF CR2
1090 ;SETS ADDR LINES 16 & 17
1091 005356 004737 010654 JSR PC,ADLI
1092 005362 000431 BR TST13 ;GO TO NEXT TEST
1093 ;:*****
1094 ;:*****
1095 005364 BRK:
1096 005364 011637 001174 MOV (SP),$REG5 ;FOR TYPEOUT OF PC
1097 005370 012737 000001 001206 MOV #1,$TMP4 ;INDICATOR FOR DEVICE 1
1098 005376 032777 007340 174210 BIT #7340,@BE1CR2 ;CHECK FOR ALL ERRS EXCEPT NO SSYN ERR
1099 005404 001003 BNE 1$ ;IF ANY ARE SET,SEE WHICH ONES
1100 005406 005077 174200 CLR @BE1CLR ;ELSE CLEAR THE NO SSYN ERR
1101 005412 000414 BR EXBRK ;AND EXIT
1102 005414
1103 005414 017737 174170 001164 1$: MOV @BE1CR1,$REG1 ;MOVES ARE FOR TYPEOUTS
1104 005422 017737 174166 001166 MOV @BE1CR2,$REG2
1105 005430 017737 174152 001170 MOV @BE1BA,$REG3
1106 005436 104011 ERROR 11 ;ERR ON ACCESSING A14 - A17
1107 005440 004737 010166 JSR PC,ERRTN ;DO ERR CHECK SUB-ROUTINE
1108 005444 000002 EXBRK: RTI ;EXIT
1109
1110
1111 ;:*****
1112 ;*TEST 13 CPU TEST FOR ACLO/DCLO SEQUENCE
1113 ;*THIS TEST CHECKS THE ASSERTION OF ACLO AND DCLO
1114 ;*AND THAT THE CPU TRAPS TO THE CORRECT SERVICE ROUTINE.
1115 ;*IF THIS PROGRAM IS RUNNING UNDER ACT11 THIS TEST
1116 ;*WILL BE SKIPPED.
1117 ;:*****
1118 005446 000004 TST13: SCOPE
1119 005450 012737 000001 001212 MOV #1,$TIMES ;DO 1 ITERATION
1120 005456 005737 000042 TST 42 ;SEE IF PROGRAM IS UNDER ACT11
1121 005462 001061 BNE TST14 ;IF UNDER ACT, DO NOT PERFORM THIS TEST
1122 005464 012705 000001 MOV #1,R5 ;INIT R5 WITH A VALUE OF 1
1123 005470
1124 005470 005205 6$: INC R5 ;ADD 1 TO R5
1125 005472 100376 BPL 6$ ;KEEP ADDING AS LONG AS R5 POS
1126 005474 012737 000001 001206 MOV #1,$TMP4 ;INDICATOR FOR DEVICE 1
1127 005502 012737 000340 177776 MOV #PR7,PSW ;SET PS=7
1128 005510 012777 004506 174144 MOV #ERRCHK,@BE1VEC ;SET UP INTR VECTOR
1129 005516 012777 000340 174140 MOV #PR7,@BE1PSW ;SET UP DEVICE INTR PSW
1130 005524 005037 001200 CLR $TMP1 ;CLEAR TEMPORARY REGISTER(TMP1)
1131 005530 012737 005606 000024 MOV #TMPPWR,PWRVEC ;SET UP SPECIAL POWER RTN
1132 005536 052777 000020 174050 BIS #BIT04,@BE1CR2 ;INDICATE PWR FAILURE BY SETTING BIT 4
    
```

```

1133 005544 004737 010626          JSR    PC,CNTR      ;PAUSE FOR TIME
1134 005550 012737 010122 000024    MOV    #PWFAL,PWRVEC ;RESTORE PWFAL SEQ FOR A PWR FAIL
1135 005556 042777 000020 174030    BIC    #BIT04,@BE1CR2 ;MAKE SURE BIT 4 IS CLEARED
1136 005564                                FAILCK:
1137 005564 022737 001154 020466    CMP    #SNULL,$PWRMG ;IF THIS TEST IS CAUSE OF
1138                                ;PWR FAIL --TYPE NULL CHAR
1139 005572 001401                                BEQ    XTST         ;IF EQUAL, EXIT TEST
1140 005574 104013                                ERROR  13          ;TYPE ERR MSSG IF FAILURE
1141 005576                                XTST:
1142 005576 012737 020500 020466    MOV    #$POWER,$PWRMG ;RESTORE TYPE OUT OF 'POWER'
1143 005604 000410                                BR     TST14       ;GO TO NEXT TEST
1144                                ;*****
1145                                ;*****
1146 005606                                TMPPWR:
1147 005606 012737 001154 020466    MOV    #SNULL,$PWRMG ;SPECIAL PWR RTN; OTHER THAN SYSMAC'S
1148 005614 042777 000020 173772    BIC    #BIT04,@BE1CR2 ;CHANGE PWR MSSG TO NULL CHAR
1149 005622 000137 020332                                JMP    $PWRDN      ;CLEAR POWER DOWN/UP BIT
1150                                ;GO TO THAT RTN
1151
1152                                ;*****
1153                                ;*TEST 14      PARITY ERROR TEST
1154                                ;*THIS TEST WILL CAUSE PARITY ERROR AND CHECKS
1155                                ;*THAT THE CPU TRAPS TO THE CORRECT VECTOR.
1156                                ;*THIS TEST SHOULD BE DESELECTED IF THE MEMORY
1157                                ;*PARITY OPTION IS NOT PRESENT, ELSE AN
1158                                ;*ERROR WILL BE REPORTED ALTHOUGH HARDWARE IS
1159                                ;*FUNCTIONING PROPERLY.
1160                                ;*SW06=1      INHIBIT TEST 14 AND GO TO NEXT TEST
1161                                ;*****
1162 005626 000004                                TST14: SCOPE
1163 005630 032777 000100 173302    BIT    #BIT06,@SWR   ;INHIBIT TEST 14?
1164 005636 001105                                BNE    TST15        ;GO TO NEXT TEST
1165 005640 012737 006050 000010    MOV    #NODO,10     ;SET UP RESERVED INSTR VECTOR
1166 005646 012737 000340 000012    MOV    #PR7,12      ;PSW=7
1167 005654 005037 001204                                CLR    $TMP3        ;SET $TMP3 = 0
1168 005660 000270                                SEN    ;SET N BIT OF CC
1169 005662 006737 001204                                SXT    $TMP3       ;IF VALID INSTR, $TMP3 WILL = -1
1170 005666 005737 001204                                TST    $TMP3       ;IF INVALID, $TMP3 WILL REMAIN 0
1171 005672 100033                                BPL    NXT         ;IF CP NOT= 35,40,45,OR 70,GO TO NEXT TEST
1172 005674 012737 000140 177776    MOV    #PR3,PSW     ;PS=3
1173 005702 012777 006014 173752    MOV    #PBERR,@BE1VEC ;SET UP DEVICE INTR
1174 005710 012737 006036 000114    MOV    #PBRTN,PBVEC ;SET UP PARITY BIT VECTOR
1175 005716 012737 000340 000116    MOV    #340,PBPSW   ;SET UP PARITY BIT PSW
1176 005724 012777 020510 173654    MOV    #ATEND,@BE1BA ;SET UP ADDo REG
1177 005732 012777 177777 173644    MOV    #-1,@BE1CC   ;SET UP CYCLE COUNT
1178 005740 052777 010000 173646    BIS    #BIT12,@BE1CR2 ;SET BIT 12 FOR PARITY ERROR
1179 005746 005777 173642                                TST    @BE1CR2     ;SET OFF PARITY ERR SEQUENCE
1180 005752 012777 013161 173630    MOV    #13161,@BE1CR1 ;TRY FUN 1-DATO FROM CC-NPR-INTR ON DONE(7)
1181 005760 000240                                NOP
1182                                ;ALLOW TIME FOR ATTEMPTED XFER
1183 005762 012737 000116 000114    NXT:  MOV    #PBPSW,PBVEC ;RESTORE
1184 005770 012737 000000 000116    MOV    #0,PBPSW     ;TRAP CATCHER HERE AND
1185 005776 012737 000012 000010    MOV    #12,10       ;AT RESERVED
1186 006004 012737 000000 000012    MOV    #0,12        ;INSTRUCTION VECTOR
1187 006012 000417                                BR     TST15       ;BRANCH TO NEXT TEST IF PARITY TRAP OCCURRED
1188 006014                                PBERR:

```



```

1189 006014 011637 001174      MOV      (SP),SREG5      ;FOR TYPEOUT OF PC
1190 006020 104014      ERROR    14              ;TYPE ERR MSSG IF DEVICE INTERRUPTED
1191 006022 012737 000001 001206  MOV      #1,$TMP4        ;INDICATOR FOR DEVICE 1
1192 006030 004737 010166      JSR      PC,ERRTN        ;CHECK TO SEE IF ANY ERRORS OCCURED
1193 006034 000002      RTI                          ;EXIT TRAP
1194
1195
1196 006036      PBRTN:
1197 006036 012777 000000 173550  MOV      #0,@BE1CR2      ;PARITY BIT TRAP RTN
1198      ;CLEAR PARITY BIT ERROR-MUST BE DONE
1199 006044 012716 005762      MOV      #NXT,(SP)      ;BY MOVING 0(S) TO BE1CR2
1200 006050 000002      NODO:  RTI              ;SET STACK FOR NEXT TEST
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210 006052 000004      TST15: SCOPE
1211 006054 004737 002566      JSR      PC,CLRREG      ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
1212 006060 012703 000000      MOV      #0,R3          ;SET DATA PATTERN = 0
1213 006064 012704 177777      MOV      #177777,R4     ;SET DATA PATTERN = ALL 1'S
1214 006070 004737 007260      JSR      PC,MULT1       ;LOAD & EXECUTE ALL DEVICES
1215 006074 022737 000002 001702  CMP      #2,DEV CNT     ;ARE THERE MORE THAN 2 DEVICES?
1216 006102 100115      BPL      TST16          ;IF 2 OR LESS, GO TO NEXT TEST
1217 006104 012703 161610      MOV      #161610,R3     ;ELSE LOAD R3 AND R4 WITH
1218 006110 012704 016161      MOV      #016161,R4     ;ANOTHER PATTERN
1219 006114
1220 006114 123737 001115 001103  5$:     CMPB    $ERMAX,$ERFLG   ;MAX ERRS FOR THIS TEST OCCURRED?
1221 006122 100505      BMI     TST16           ;BR IF YES TO NEXT TEST
1222 006124 004737 010360      JSR     PC,ROTATE       ;ROTATE DATA PATTERNS
1223 006130 004737 007260      JSR     PC,MULT1        ;LOAD & EXECUTE ALL DEVICES
1224 006134 022703 107070      CMP     #107070,R3     ;IS R3 = 107070?
1225 006140 001365      BNE     5$              ;IF NOT,ROTATE AND DO AGAIN
1226 006142 012703 167777      MOV     #167777,R3     ;ELSE MOVE NEW PATTERNS
1227 006146 012704 010000      MOV     #010000,R4     ;INTO R3 AND R4
1228 006152
1229 006152 123737 001115 001103  10$:    CMPB    $ERMAX,$ERFLG   ;MAX ERRS FOR THIS TEST OCCURRED?
1230 006160 100466      BMI     TST16           ;BR IF YES TO NEXT TEST
1231 006162 004737 010360      JSR     PC,ROTATE       ;ROTATE DATA PATTERNS
1232 006166 004737 007260      JSR     PC,MULT1        ;LOAD & EXECUTE ALL DEVICES
1233 006172 022703 167777      CMP     #167777,R3     ;IS R3 = 167777 AGAIN?
1234 006176 001365      BNE     10$            ;IF NOT,ROTATE AND DO AGAIN
1235 006200 000456      BR      TST16           ;GO TO NEXT TEST
1236
1237
1238
1239 006202      SERV1:
1240 006202 017737 173400 001714  MOV     @BE1BA,DATA1    ;MOVE ADDR IN BE1BA TO DATA1 AND
1241 006210 162737 000001 001714  SUB     #1,DATA1        ;SUB 1 TO GET ACTUAL ADDR
1242 006216 012737 000001 001206  MOV     #1,$TMP4        ;INDICATOR FOR DEVICE 1
1243 006224 000435      BR      INK            ;BRANCH TO INK
1244 006226      SERV2:

```

```

1245 006226 017737 173370 001716      MOV    @BE2BA,DATA2    ;MOVE ADDR IN BE2BA TO DATA2 AND
1246 006234 162737 000002 001716      SUB    #2,DATA2       ;SUB 2 TO GET ACTUAL ADDR
1247 006242 012737 000002 001206      MOV    #2,$TMP4       ;INDICATOR FOR DEVICE 2
1248 006250 000423                    BR     INK             ;BRANCH TO INK
1249 006252
1250 006252 017737 173360 001720      MOV    @BE3BA,DATA3    ;MOVE ADDR IN BE3BA TO DATA3 AND
1251 006260 162737 000002 001720      SUB    #2,DATA3       ;SUB 2 TO GET ACTUAL ADDR
1252 006266 012737 000003 001206      MOV    #3,$TMP4       ;INDICATOR FOR DEVICE 3
1253 006274 000411                    BR     INK             ;BRANCH TO INK
1254 006276
1255 006276 017737 173350 001722      MOV    @BE4BA,DATA4    ;MOVE ADDR IN BE4BA TO DATA4 AND
1256 006304 162737 000002 001722      SUB    #2,DATA4       ;SUB 2 TO GET ACTUAL ADDR
1257 006312 012737 000004 001206      MOV    #4,$TMP4       ;INDICATOR FOR DEVICE 4
1258 006320
1259 006320 005237 001166      INC    $REG2           ;INCREMENT REG
1260 006324 011637 001174      MOV    (SP),$REG5     ;FOR TYPEOUT OF PC
1261 006330 004737 010166      JSR    PC,ERRTN       ;CHECK FOR ANY ERRS
1262 006334 000002                    RTI                    ;EXIT
1263
1264
1265
1266
1267
1268
1269
1270
1271 006336 000004
1272 006340 012702 020510      MOV    #ATEND,R2      ;R2 = END OF PROG
1273 006344 012705 005000      MOV    #5000,R5       ;R5 = THE # OF DATA WORDS
1274 006350 004737 010610      JSR    PC,DOUP        ;CREATE THOSE WORDS IN BUFFER MEMORY
1275 006354 004737 010722      JSR    PC,TSTOVR     ;SET UP PATTERN IN MEMORY BUFFER AREA
1276 006360 004737 002566      JSR    PC,CLRREG     ;CLEAR CONTENTS OF ALL AVAILABLE DEVS
1277
1278 006364 012737 000000 177776      MOV    #PRO,PSW       ;PS=0
1279 006372 012777 007124 173262      MOV    #S1,@BE1VEC    ;SET UP DEVICE 1 INTR VECTOR
1280 006400 012777 000340 173256      MOV    #PR7,@BE1PSW   ;SET UP DEVICE 1 PSW VECTOR
1281 006406 012777 022510 173172      MOV    #ATEND+2000,@BE1BA ;SET UP ADDR REG
1282 006414 012777 176000 173162      MOV    #-2000,@BE1CC  ;SET UP CYCLE COUNT
1283 006422 012777 015551 173160      MOV    #15551,@BE1CR1 ;DO FUN 2-DATOB FROM CC-NPR-INTR ON DONE(6)
1284 006430 022737 000001 001702      CMP    #1,DEVCNT     ;CHECK FOR MORE THAN 1 DEVICE
1285 006436 001467                    BEQ    1$            ;IF NOT, GO CHECK RESULTS
1286
1287 006440 012777 007156 173220      MOV    #S2,@BE2VEC    ;SET UP DEVICE 2 INTR VECTOR
1288 006446 012777 000340 173214      MOV    #PR7,@BE2PSW   ;SET UP DEVICE 2 PSW VECTOR
1289 006454 012777 177000 173136      MOV    #-1000,@BE2CC  ;SET UP CYCLE COUNT FOR 1000 XFERS
1290 006462 012777 020510 173132      MOV    #ATEND,@BE2BA  ;SET UP ADDR REG=1ST LOCATION AFTER PROG
1291 006470 012777 002561 173126      MOV    #2561,@BE2CR1  ;DO FUN 1-DATIP-NPR-INTR ON DONE(7)
1292 006476 022737 000002 001702      CMP    #2,DEVCNT     ;CHECK FOR MORE THAN 2 DEVICES
1293 006504 001444                    BEQ    1$            ;IF NOT, GO CHECK RESULTS
1294
1295 006506 012777 007210 173156      MOV    #S3,@BE3VEC    ;SET UP DEVICE 3 INTR VECTOR
1296 006514 012777 000340 173152      MOV    #PR7,@BE3PSW   ;SET UP PSW VECTOR
1297 006522 012777 176776 173104      MOV    #-1002,@BE3CC  ;SET UP CYCLE COUNT
1298 006530 012777 020510 173100      MOV    #ATEND,@BE3BA  ;SET UP ADDR REG
1299 006536 012777 004005 173074      MOV    #4005,@BE3CR1  ;DO FUN 2-DATI-BR5
1300 006544 022737 000003 001702      CMP    #3,DEVCNT     ;CHECK FOR MORE THAN 3 DEVICES

```

::*****

::*****

*TEST 16 MULTITRANSFERS II
;THIS TEST WILL HAVE THE AVAILABLE EXERCISERS DOING
;VARIOUS TRANSFERS AND/OR INTERRUPTS AT DIFFERENT
;REQUEST LEVELS TO FURTHER CHECK CPU HANDLING CAPABILITIES

::*****

```

TST16: SCOPE
MOV    #ATEND,R2      ;R2 = END OF PROG
MOV    #5000,R5       ;R5 = THE # OF DATA WORDS
JSR    PC,DOUP        ;CREATE THOSE WORDS IN BUFFER MEMORY
JSR    PC,TSTOVR     ;SET UP PATTERN IN MEMORY BUFFER AREA
JSR    PC,CLRREG     ;CLEAR CONTENTS OF ALL AVAILABLE DEVS

MOV    #PRO,PSW       ;PS=0
MOV    #S1,@BE1VEC    ;SET UP DEVICE 1 INTR VECTOR
MOV    #PR7,@BE1PSW   ;SET UP DEVICE 1 PSW VECTOR
MOV    #ATEND+2000,@BE1BA ;SET UP ADDR REG
MOV    #-2000,@BE1CC  ;SET UP CYCLE COUNT
MOV    #15551,@BE1CR1 ;DO FUN 2-DATOB FROM CC-NPR-INTR ON DONE(6)
CMP    #1,DEVCNT     ;CHECK FOR MORE THAN 1 DEVICE
BEQ    1$            ;IF NOT, GO CHECK RESULTS

MOV    #S2,@BE2VEC    ;SET UP DEVICE 2 INTR VECTOR
MOV    #PR7,@BE2PSW   ;SET UP DEVICE 2 PSW VECTOR
MOV    #-1000,@BE2CC  ;SET UP CYCLE COUNT FOR 1000 XFERS
MOV    #ATEND,@BE2BA  ;SET UP ADDR REG=1ST LOCATION AFTER PROG
MOV    #2561,@BE2CR1  ;DO FUN 1-DATIP-NPR-INTR ON DONE(7)
CMP    #2,DEVCNT     ;CHECK FOR MORE THAN 2 DEVICES
BEQ    1$            ;IF NOT, GO CHECK RESULTS

MOV    #S3,@BE3VEC    ;SET UP DEVICE 3 INTR VECTOR
MOV    #PR7,@BE3PSW   ;SET UP PSW VECTOR
MOV    #-1002,@BE3CC  ;SET UP CYCLE COUNT
MOV    #ATEND,@BE3BA  ;SET UP ADDR REG
MOV    #4005,@BE3CR1  ;DO FUN 2-DATI-BR5
CMP    #3,DEVCNT     ;CHECK FOR MORE THAN 3 DEVICES

```


:BE3 CHECK ROUTINE

:/*/:
:/*/:

```

1361 006764 105777 172650          16$:  TSTB  @BE3CR1      ;CHECK IF DEVICE 3 DONE
1362 006770 100401                   BMI  20$           ;IF YES, CHECK NEXT DEVICE
1363 006772 000774                   BR   16$           ;AND GO BACK TO SEE IF DONE YET
1364 006774                                20$:
1365 006774 022777 176002 172630     CMP  #-1776,@BE3DB ;CHECK FOR CORR VALUE IN BE3DB
1366 007002 001407                   BEQ  25$           ;IF EQUAL, SKIP ERR TYPE OUT
1367 007004 017737 172622 001164     MOV  @BE3DB,$REG1 ;MOVE FOR ERR TYPE OUT
1368 007012 012737 176002 001166     MOV  #-1776,$REG2
1369 007020 104032                   ERROR 32          ;TYPE ERR MSSG
1370 007022                                25$:
1371 007022 022737 000003 001702     CMP  #3,DEV CNT   ;CHECK IF ONLY 3 DEVICES OPERATED
1372 007030 001421                   BEQ  TST17        ;<IF EQUAL, GO TO NEXT TEST>

```

:/*/:
:/*/:

:BE4 CHECK ROUTINE

:/*/:
:/*/:

```

1380 007032                                22$:
1381 007032 105777 172616           TSTB  @BE4CR1      ;TEST IF DEVICE 4 IS DONE
1382 007036 100375                   BPL  22$           ;IF NOT, GO BACK TO DELAY RTN
1383 007040 022737 003000 001172     CMP  #3000,$REG4  ;CHECK IF REG4 COUNTED 3000 INTRS
1384 007046 001407                   BEQ  26$           ;IF EQUAL SKIP ERR TYPE OUT
1385 007050 013737 001172 001164     MOV  $REG4,$REG1  ;MOVE FOR TYPE OUT
1386 007056 012737 003000 001166     MOV  #3000,$REG2
1387 007064 104033                   ERROR 33          ;TYPE ERR MSSG
1388 007066                                26$:
1389 007066 042777 040000 172564     BIC  #BIT14,@BE4CR2 ;ELSE CLEAR TIME DELAY BIT
1390
1391                                ;*****
1392                                ;*TEST 17 DUMMY END OF PROGRAM
1393                                ;*****
1393 007074 000004                                TST17: SCOPE
1394 007076 012737 000001 001212     MOV  #1,$TIMES    ;:DO 1 ITERATION
1395
1396 007104 017700 172036           MOV  @$TKB,R0      ;MOVE READ BUFF CONTENTS TO R0
1397 007110 022700 000210           CMP  #210,R0       ;DOES THE VALUE = 'H' ?
1398 007114 001001                   BNE  10$           ;IF NOT GO TO 10$
1399 007116 000000                   HALT                ;ELSE HALT THE PROGRAM
1400
1401                                10$:
1401 007120 000137 015206           JMP  $EOP
1402
1403                                ;*****
1404                                ;SET UP DEVICE INTR VECTOR
1405                                ;*****
1405 007124                                S1:
1406
1407 007124 017737 172456 001714     MOV  @BE1BA,DATA1 ;MOVE ADDR IN BE1BA TO DATA1 AND
1408 007132 162737 000002 001714     SUB  #2,DATA1      ;SUB 2 TO GET ACTUAL ADDR
1409 007140 012737 000001 001206     MOV  #1,$TMP4      ;SET INDICATOR FOR DEVICE 1
1410 007146 005777 172436           TST  @BE1CR1       ;TEST FOR ERROR
1411 007152 100041                   BPL  EXS           ;IF PLUS, EXIT
1412 007154 000434                   BR   CHEX          ;ELSE FIND CAUSE OF INTR

```

1413	007156				S2:			
1414	007156	017737	172440	001716		MOV	@BE2BA,DATA2	;MOVE ADDR IN BE2BA TO DATA2 AND
1415	007164	162737	000002	001716		SUB	#2,DATA2	;SUB 2 TO GET ACTUAL ADDR
1416	007172	012737	000002	001206		MOV	#2,\$TMP4	;SET INDICATOR FOR DEVICE 2
1417	007200	005777	172420			TST	@BE2CR1	;TEST FOR ERROR
1418	007204	100024				BPL	EXS	;IF PLUS EXIT
1419	007206	000417				BR	CHEX	;ELSE FIND CAUSE OF INTR

1420 007210
1421 007210 012737 000003 001206
1422 007216 005777 172416
1423 007222 100015

S3:

MOV #3,\$TMP4
TST @BE3CR1
BPL EXS

→:SET INDICATOR FOR DEVICE 3
;TEST FOR ERROR
;IF PLUS, EXIT

UNIBUS EXERCISER MACY11 30A(1052) 04-OCT-79 12:49 PAGE 32
CZKUA.E.P11 27-SEP-79 09:25 T17 DUMMY END OF PROGRAM

SEQ 0038

1424	007224	000410		BR	CHEX	;ELSE FIND CAUSE OF INTR
1425	007226		S4:			
1426	007226	005237 001172		INC	\$REG4	;COUNT DEVICE 4'S INTR

```

1427 007232 012737 000004 001206      MOV    #4,$TMP4      ;SET INDICATOR FOR DEVICE 4
1428 007240 005777 172410      TST    @BE4CR1      ;TEST FOR ERROR
1429 007244 100004      BPL    EXS          ;IF PLUS, EXIT
1430 007246      CHEX:
1431 007246 011637 001174      MOV    (SP),$REG5   ;FOR TYPEOUT OF PC
1432 007252 004737 010166      JSR    PC,ERRTN     ;ELSE FIND CAUSE OF INTR
1433 007256      EXS:
1434 007256 000002      RTI
1435      ;:*****
1436      ;:*****
1437 007260      MULT1:
1438 007260 012702 020510      MOV    #ATEND,R2    ;R2 = END OF PROG
1439 007264 012705 005000      MOV    #5000,R5     ;R5 = THE # OF DATA WORDS
1440 007270 004737 010610      JSR    PC,DOUP      ;CREATE THOSE WORDS IN BUFFER MEMORY
1441 007274 004737 010722      JSR    PC,TSTOVR    ;SET UP PATTERN IN MEMORY BUFFER AREA
1442 007300 012777 020510 172300      MOV    #ATEND,@BE1BA ;SET REG ADDR= 1ST LOCATION AFTER END OF PROGRAM
1443 007306 012777 176000 172270      MOV    #-2000,@BE1CC ;SET CYCLE COUNT FOR 2000 XFERS
1444 007314 012777 006202 172340      MOV    #SERV1,@BE1VEC ;SET UP DEVICE INTR VECTOR
1445 007322 012777 000340 172334      MOV    #PR7,@BE1PSW ;SET UP DEVICE PSW VECTOR
1446 007330 052777 040000 172256      BIS    #BIT14,@BE1CR2 ;SET BIT 14 FOR TIME DELAY ENABLE
1447 007336 012777 042560 172244      MOV    #42560,@BE1CR1 ;DO DATIP/DATOB-FUN 1-NPR-INTR ON DONE(7)
1448 007344 122737 000001 001702      CMPB  #1,DEV CNT   ;IF MORE THAN 1 DEVICE, LOAD THEIR REGISTERS
1449 007352 001474      BEQ    6$          ;OTHERWISE BEGIN TESTING
1450 007354      3$:
1451 007354 012777 006226 172304      MOV    #SERV2,@BE2VEC ;SET UP DEVICE 2 INTR VECTOR
1452 007362 012777 000340 172300      MOV    #PR7,@BE2PSW ;SET UP DEVICE 2 PSW VECTOR
1453 007370 012777 020510 172224      MOV    #ATEND,@BE2BA ;SET UP ADDR REG FOR SAME LOCATIONS AS DEVICE 1
1454 007376 012777 177000 172214      MOV    #-1000,@BE2CC ;SET CYCLE COUNT FOR A 1000 XFERS
1455 007404 012777 024510 172212      MOV    #24510,@BE2CR1 ;DO DATIP/NO ROTATE-FUN 2-BR6-INTR ON DONE(6)
1456 007412 122737 000002 001702      CMPB  #2,DEV CNT   ;IF MORE THAN 2 DEVICES, LOAD THEIR REGISTERS
1457 007420 001451      BEQ    6$          ;OTHERWISE BEGIN TESTING
1458 007422      4$:
1459 007422 012777 000340 172244      MOV    #PR7,@BE3PSW ;SET UP DEVICE 3 PSW VECTOR
1460 007430 010377 172176      MOV    R3,@BE3DB    ;MOVE PATTERN IN R3 TO DEVICE DATA REG
1461 007434 012777 006252 172230      MOV    #SERV3,@BE3VEC ;SET UP DEVICE INTR VECTOR
1462 007442 012777 022510 172166      MOV    #ATEND+2000,@BE3BA ;SET UP ADDR REG
1463 007450 012777 177000 172156      MOV    #-1000,@BE3CC ;SET UP FOR 1000 XFERS
1464 007456 052777 040000 172160      BIS    #BIT14,@BE3CR2 ;SET BIT 14 FOR TIME DELAY ENABLE
1465 007464 012777 003160 172146      MOV    #3160,@BE3CR1 ;DO DATO-FUN 1-FROM DB-NPR-INTR ON DONE(7)
1466 007472 122737 000003 001702      CMPB  #3,DEV CNT   ;IF A 4TH DEVICE, GO AND LOAD REGISTERS
1467 007500 001421      BEQ    6$          ;OTHERWISE BEGIN TESTING
1468 007502      5$:
1469 007502 010477 172140      MOV    R4,@BE4DB    ;MOVE PATTERN IN R4 TO DEVICE DATA REG
1470 007506 012777 000340 172164      MOV    #PR7,@BE4PSW ;SET UP DEVICE 4 PSW VECTOR
1471 007514 012777 006276 172154      MOV    #SERV4,@BE4VEC ;SET UP DEVICE 4 INTR VECTOR
1472 007522 012777 024510 172122      MOV    #ATEND+4000,@BE4BA ;SET UP ADDR REG
1473 007530 012777 177000 172112      MOV    #-1000,@BE4CC ;SET CYCLE COUNT FOR 1000 XFERS
1474 007536 012777 003104 172110      MOV    #3104,@BE4CR1 ;DO DATO-FUN 1-BR5-INTR ON DONE
1475 007544      6$:
1476 007544 005277 172030      INC    @SIMLGO      ;START DEVICES SIMULTANEOUSLY
    
```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;BACKGROUND ROUTINE FOR MULTITRANSFERS I
    
```

```

1482 007550 012737 000001 001164      MOV    #1,$REG1     ;MOVE 1 TO TEMPORARY REG
    
```



```

1483 007556 012701 001164      MOV    #SREG1,R1      ;SET UP R1 AS POINTER
1484 007562 005121              7$:  COM    (R1)+      ;COMPLEMENT TEMP REG
1485 007564 006041              ROR    -(R1)         ;ROTATE CONTENTS RIGHT
1486 007566 123737 001702 001166  CMPB   DEVCNT,$REG2   ;CHECK IF ALL DEVICES ARE DONE
1487 007574 101372              BHI    7$            ;IF NOT, CONTINUE WITH BACKGROUND RTN
1488                               ;:*****

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; DEVICE 1 TRANSFER CHECKS
; THERE ARE NO CHECKS FOR BE2
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

1497 007576 042777 040000 172010      BIC    #BIT14,@BE1CR2 ;CLEAR TIME DELAY BIT
1498 007604 012700 020510              MOV    #ATEND,R0     ;START CHECKING FOR CORRECT XFERS
1499 007610                          8$:
1500 007610 122720 000125      CMPB   #125,(R0)+    ;COMPARE LOWER BYTE
1501 007614 001012      BNE    20$          ;IF NOT EQUAL, BR TO ERR MSSG
1502 007616 023700 001714      CMP    DATA1,R0    ;IS THIS LAST BYTE COMPARE?
1503 007622 001422      BEQ    9$           ;IF SO, BR TO 9$
1504 007624 122720 000124      CMPB   #124,(R0)+    ;ELSE COMPARE UPPER BYTE
1505 007630 001006      BNE    22$          ;IF NOT EQUAL, BR TO ERR MSSG
1506 007632 023700 001714      CMP    DATA1,R0    ;IS THIS LAST BYTE COMPARE?
1507 007636 001414      BEQ    9$           ;IF SO, BR TO 9$
1508 007640 000763      BR     8$           ;ELSE CHECK NEXT ADDR
1509 007642                          20$:
1510 007642 105740      TSTB   -(R0)        ;SUB 1 TO GET ERR ADDR
1511 007644 000401      BR     24$         ;GO DO MOVES FOR ERR MSSG
1512 007646                          22$:
1513 007646 005740      TST    -(R0)        ;SUB 2 TO GET ERR ADDR
1514 007650                          24$:
1515 007650 011037 001164      MOV    (R0),$REG1    ;MOVES ARE FOR ERR MSSG
1516 007654 010037 001162      MOV    R0,$REG0
1517 007660 012737 052125 001170      MOV    #052125,$REG3
1518 007666 104015      ERROR  15           ;ELSE TYPE ERR MSSG
1519 007670                          9$:
1520 007670 022737 000001 001702      CMP    #1,DEVCNT    ;IF ONLY ONE DEVICE
1521 007676 001447      BEQ    13$          ;IF NO MORE DEVICES,EXIT RTN
1522 007700 122737 000002 001702      CMPB   #2,DEVCNT    ;CHECK FOR MORE THAN 2 DEVICES
1523 007706 001773      BEQ    25$          ;IF NOT, EXIT TEST

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; BE3 TRANSFER CHECKS
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

1531 007710 042777 040000 171726      BIC    #BIT14,@BE3CR2 ;CLEAR TIME DELAY BIT OF DEVICE 3
1532 007716 012700 022510              MOV    #ATEND+2000,R0 ;CHECK NEXT 2000 LOCATIONS
1533 007722 023700 001720      10$:  CMP    DATA3,R0    ;CHECK FOR 1000 XFERS
1534 007726 001411      BEQ    11$          ;IF SO, CHECK NEXT BLOCK
1535 007730 020320      CMP    R3,(R0)+    ;TEST FOR CORRECT PATTERNS
1536 007732 001773      BEQ    10$          ;IF NO ERR, CHECK ANOTHER LOC
1537 007734 010337 001170      MOV    R3,$REG3     ;THE MOVE IS FOR TYPEOUT REASONS
1538 007740 014037 001164      MOV    -(R0),$REG1

```



```

1595 010106 012737 000004 001206      MOV    #4,$TMP4      ;INDICATOR FOR DEV 4
1596 010114                               XQ:
1597 010114 004737 010166      JSR    PC,ERRTN     ;GO TO ERR ROUTINE
1598 010120 000002      RTI
1599
1600
1601 010122      PWRFAL:
1602 010122 010146      MOV    R1,-(SP)     ;SAVE CONTENTS OF R1
1603 010124 010046      MOV    R0,-(SP)     ;SAVE CONTENTS OF R0
1604 010126 012700 001614      MOV    #BE1CR2,R0   ;R0 POINTS TO DEV 1 CR2 ADDR
1605 010132 005001      CLR    R1           ;CLEAR R1
1606 010134
1607 010134 042770 000020 000000 5$:      BIC    #BIT04,@(R0) ;CLR BIT 4 OF CURRENT CR2
1608 010142 062700 000014      ADD    #14,R0       ;ADD 14 TO POINT TO NEXT CR2
1609 010146 005201      INC    R1           ;COUNT THE NUMBER OF DEVS
1610 010150 020137 001702      CMP    R1,DEV CNT   ;REACHED MAX # ON BUS?
1611 010154 103767      BLO    5$           ;IF NOT, CLR NEXT CR2
1612 010156 012600      MOV    (SP)+,R0     ;ELSE RESTORE R0
1613 010160 012601      MOV    (SP)+,R1     ;AND R1
1614 010162 000137 020332      JMP    $PWRDN       ;THEN DO REGULAR PWR DOWN RTN
1615
1616
1617 010166      ERRTN:
1618 010166 104410      SAVREG              ;SAVE REGISTERS
1619 010170 012700 001600      MOV    #BE1CR2-14,R0 ;INITIALIZE R0
1620 010174 105005      CLRB   R5           ;CLEAR DEVICE COUNTER
1621 010176
1622 010176 105205      1$:      INCB   R5           ;ADD 1 TO COUNTER
1623 010200 062700 000014      ADD    #14,R0       ;SET R0=ADDR OF CR2 OF NEXT DEVICE
1624 010204 120537 001206      CMPB  R5,$TMP4     ;IF COUNTER NOT EQUAL TO INDICATOR
1625 010210 001372      BNE    1$           ;ADD 1 TO COUNTER & CHECK AGAIN
1626 010212
1627 010212 105770 000000      CHKERR: TSTB  @ (R0)    ;CHECK FOR NO NOSACK TIMEOUT
1628 010216 100001      BPL    1$           ;IF NOT, SEE IF THERE ARE ANY ERRS
1629 010220 104022      ERROR  22          ;TYPE ERR MESSG FOR NO NOSACK
1630 010222
1631 010222 032770 007540 000000 1$:      BIT    #7540,@(R0)  ;CHECK FOR OTHER ERRORS
1632 010230 001436      BEQ    LEEV         ;IF NO ERRORS,EXIT
1633 010232 032770 004000 000000      BIT    #BIT11,@(R0) ;CHECK FOR NO SSYN ON INTR
1634 010240 001401      BEQ    10$         ;IF NOT SET, CHECK FOR NEXT ERR
1635 010242 104023      ERROR  23          ;TYPE ERR MSSG FOR NO SSYN ON INTR
1636 010244
1637 010244 132770 000040 000000 10$:     BITB  #BIT05,@(R0)  ;CHECK FOR WRONG GRANT ERR
1638 010252 001401      BEQ    2$           ;IF NOT, CHECK BIT 6
1639 010254 104024      ERROR  24          ;ELSE TYPE ERR MESSG FOR WRONG GRANT
1640 010256
1641 010256 132770 000100 000000 2$:      BITB  #BIT06,@(R0)  ;CHECK FOR BUS LATE ERR
1642 010264 001401      BEQ    3$           ;IF NOT, CHECK BIT 8
1643 010266 104025      ERROR  25          ;TYPE ERR MSSG FOR BUS LATE
1644 010270
1645 010270 032770 000400 000000 3$:      BIT    #BIT08,@(R0) ;CHECK FOR NO SSYN ERR
1646 010276 001401      BEQ    4$           ;IF NOT, CHECK BIT 9
1647 010300 104026      ERROR  26          ;TYPE ERR MSSG FOR NO SSYN
1648 010302
1649 010302 032770 001000 000000 4$:      BIT    #BIT09,@(R0) ;CHECK FOR WRONG ADDR ERR
1650 010310 001401      BEQ    5$           ;IF NOT, CHECK BIT 10

```

```

1651 010312 104027          ERROR 27          ;TYPE ERR MSSG FOR WRONG ADDR
1652 010314                5$:
1653 010314 032770 002000 000000 BIT #BIT10,@(R0) ;CHECK FOR NO GRANT ERR
1654 010322 001401          BEQ LEEV          ;IF NOT, EXIT
1655 010324 104030          ERROR 30          ;TYPE ERR MSSG FOR NO GRANT
1656 010326                LEEV:
1657 010326 162700 000002 SUB #2,R0          ;POINT TO DEVICE CLEAR REG
1658 010332 005070 000000 CLR @(R0)         ;CLEAR ALL ERRORS
1659 010336 104411          RESREG          ;RESTORE REGISTERS
1660 010340 000207          RTS PC           ;EXIT
1661
1662
1663
1664 010342                *****
1665 010342 011637 001166 THRU: MOV (SP),SREG2 ;MOVE FOR ERR TYPE OUT
1666 010346 162737 000002 001166 SUB #2,SREG2 ;SUB 2 FOR ACTUAL ADDR
1667 010354 104035          ERROR 35          ;TYPE ERR MSSG
1668 010356 000002          RTI
1669
1670
1671 010360                ROTATE:
1672 010360 032703 000001 BIT #BIT00,R3      ;IS LSB A 1 OR 0 ?
1673 010364 001402          BEQ 5$           ;IF 0, GO TO 5$
1674 010366 000261          SEC           ;ELSE SET C BIT OF COND CODES
1675 010370 000401          BR 10$          ;AND GO ROTATE
1676 010372                5$:
1677 010372 000241          CLC           ;CLEAR C BIT OF COND CODES
1678 010374                10$:
1679 010374 006003          ROR R3          ;ROTATE R3
1680 010376 032704 000001 BIT #BIT00,R4      ;IS LSB A 1 OR 0 ?
1681 010402 001402          BEQ 15$          ;IF 0, GO TO 15$
1682 010404 000261          SEC           ;ELSE SET C BIT OF COND CODES
1683 010406 000401          BR 20$          ;AND GO ROTATE
1684 010410                15$:
1685 010410 000241          CLC           ;CLEAR C BIT OF COND CODES
1686 010412                20$:
1687 010412 006004          ROR R4          ;ROTATE R4
1688 010414 000207          RTS PC
1689
1690
1691 010416                NOG:
1692 010416                2$:
1693 010416 010037 177776 MOV R0,PSW        ;SET UP PROCESSOR STATUS
1694 010422 012777 020510 171156 MOV #ATEND,@BE1BA ;SET UP ADDR REG
1695 010430 012777 177777 171146 MOV #-1,@BE1CC    ;SET UP CYCLE COUNT FOR 1 CYCLE
1696 010436 010177 171146 MOV R1,@BE1CR1     ;DO FUN 1 ON BR LEVELS IN R1
1697 010442 000240          NOP           ;WAIT FOR DEVICE TO ATTEMPT TO DO XFER
1698 010444 022777 177777 171132 CMP #-1,@BE1CC    ;SEE IF DEVICE OPERATED
1699 010452 001005          BNE 4$           ;IF IT DID,GO TYPE ERR MSSG
1700 010454 106201          ASRB R1         ;SHIFT BYTE RIGHT TO LOWER BRRO
1701 010456 122701 000001 CMPB #1,R1        ;IF BYTE IS NOT EQUAL TO 1
1702 010462 001355          BNE 2$           ;GO TO 2$
1703 010464 000402          BR EXNOG        ;EXIT
1704 010466                4$:
1705 010466 004737 010474 JSR PC,ERRS      ;EXIT SUB RTN
1706 010472 000207          EXNOG: RTS PC
    
```

1707
1708 010474
1709 010474 017737 171102 001162
1710 010502 017737 171076 001164
1711 010510 017737 171072 001166
1712 010516 017737 171066 001170
1713 010524 010037 001172
1714 010530 104002
1715 010532 000207
1716
1717
1718 010534
1719 010534 012737 031463 001164
1720 010542 012701 001165
1721 010546 105441
1722 010550 105421
1723 010552 105421
1724 010554 105770 000000
1725 010560 100402
1726 010562 105441
1727 010564 000770
1728 010566
1729
1730 010566 005741
1731 010570 022711 146463
1732 010574 001404
1733 010576 012737 146463 001166
1734 010604 104031
1735 010606 000207
1736
1737
1738 010610
1739 010610 012701 000001
1740 010614
1741 010614 010122
1742 010616 005201
1743 010620 020105
1744 010622 101774
1745 010624 000207
1746
1747
1748
1749 010626
1750 010626 012737 000001 001172
1751 010634 062737 000001 001172
1752 010642 022737 000106 001172
1753 010650 001371
1754 010652 000207
1755
1756
1757 010654
1758 010654 012700 040000
1759 010660
1760 010660 012777 177777 170716
1761 010666 010077 170714
1762 010672 012777 002041 170710

```
*****  
ERRS:  MOV @BE1DB,$REG0 ;MOVES ARE FOR TYPEOUTS  
      MOV @BE1CC,$REG1  
      MOV @BE1BA,$REG2  
      MOV @BE1CR1,$REG3  
      MOV R0,$REG4  
      ERROR 2  
      RTS PC ;EXIT ERROR RTN  
*****  
BKGD:  MOV #031463,$REG1 ;START OF BACKGROUND ROUTINE  
      MOV #R1+1,R1 ;USE R1 TO POINT TO TEST PATTERN  
1$:   NEG B -(R1) ;DECREMENT LOC AND NEGATE BYTE=(031715)  
      NEG B (R1)+ ;NEGATE BYTE THEN INCREMENT LOC=(031463)  
      NEG B (R1)+ ;NEGATE BYTE THEN INCREMENT LOC=(146463)  
      TSTB @R0 ;TEST FOR DONE BIT OF DEVICE IN R0  
      BMI 2$ ;IF DONE, GO CHECK RESULTS  
      NEG B -(R1) ;ELSE DECREMENT LOC AND NEGATE BYTE=(031463)  
      BR 1$ ;CONTINUE WITH BACKGROUND  
2$:   TST -(R1) ;BRING POINTER DOWN TO REG1  
      CMP #146463,(R1) ;COMPARE EXPECTED PATTERN WITH THAT IN R1  
      BEQ BKEX ;IF EQUAL, EXIT THIS RTN  
      MOV #146463,$REG2 ;MOVE FOR TYPE OUT  
      ERROR 31 ;ELSE TYPE ERR MSSG  
      RTS PC  
*****  
DOUP:  MOV #1,R1 ;INIT R1 TO 1  
5$:   MOV R1,(R2)+ ;MOVE CONTENTS OF R1 TO AREA IN R2  
      INC R1 ;ADD 1 TO R1  
      CMP R1,R5 ;IS # OF MOVES = TO # IN R5?  
      BLOS 5$ ;IF NOT, DO ANOTHER MOVE  
      RTS PC ;ELSE EXIT  
*****  
CNTR:  MOV #1,$REG4 ;INITIALIZE COUNTER REG  
1$:   ADD #1,$REG4 ;ADD 1 TO IT  
      CMP #70.,$REG4 ;DELAY AT LEAST 41 US  
      BNE 1$ ;IF NOT, GO BACK AND ADD 1 TO REG4  
      RTS PC ;EXIT  
*****  
ADLI:  MOV #40000,R0 ;USE R0 TO SET BIT 14  
1$:   MOV #-1,@BE1CC ;SET CYCLE COUNT = 1 XFER  
      MOV R0,@BE1BA ;SET ADDR AS SPECIFIED IN R0  
      MOV #2041,@BE1CR1 ;DO DATI-FUN 1-NPR
```

```

1763 010700 004737 010626 JSR PC,CNTR ;ALLOW TIME FOR RDY BIT TO SET
1764 010704 022700 100000 CMP #100000,R0 ;CHECK IF BIT 15 OF R0 IS SET
1765 010710 001403 BEQ EXAD ;IF SET, GO SET NEXT ADDR LINE
1766 010712 012700 100000 MOV #100000,R0 ;ELSE, NOW SET BIT 15 OF R0
1767 010716 000760 BR 1$ ;GO BACK AND CHECK THAT ADDR
1768 010720 EXAD:
1769 010720 000207 RTS PC ;EXIT SUB ROUTINE
1770
1771
1772 010722
1773 010722 012737 000140 177776 TSTOVR: MOV #PR3,PSW ;PS=3
1774 010730 005037 001166 CLR $REG2 ;CLEAR REG FOR INTR ON DONE COUNTER
1775 010734 012700 020510 MOV #ATEND,R0 ;SET UP R0 AS POINTER
1776 010740 012720 125252 1$: MOV #125252,(R0)+ ;MOVE DATA PATTERN TO AVAILABLE MEMORY
1777 010744 022700 022510 CMP #ATEND+2000,R0 ;CHECK FOR A 2000 MOVES
1778 010750 001373 BNE 1$ ;IF NOT, GO BACK AND MOVE AGAIN
1779 010752 000207 RTS PC ;EXIT
1780
1781
1782
1783 010754 005015 005015 047125 QNO: .ASCIZ <15><12><15><12>\UNIBUS SYSTEMS EXERCISER DIAGNOSTIC - CZKUA-E \
011037 015 042012 053105 FOR4: .ASCIZ <15><12>\DEV 4 MUST HAVE TIME DELAY SET @ 100 US OR LATENCY ERR MAY OCCU
011144 050103 020125 051124 EM1: .ASCIZ \CPU TRAPPED THRU LOCATION 4 -TIMEOUT \
011212 040440 042104 020122 DH1: .ASCIZ \ ADDR $ERRPC #ERR/TST#\
011243 103 052520 044440 EM2: .ASCII \CPU ISSUED A BUS GRANT WITH PSW = 7\
011306 042504 020126 020061 .ASCIZ \DEV 1 SHOULD NOT HAVE BECOME BUS MASTER\
011356 042502 042061 020102 DH2: .ASCIZ \BE1DB BE1CC BE1BA BE1CR1 PSW $ERRPC #ERR/TST#\
011446 050103 020125 044504 EM3: .ASCIZ \CPU DID NOT ISSUE BUS NPG\
011500 042502 041461 030522 DH3: .ASCIZ \BE1CR1 BE1CC FM-PS TO-PS $ERRPC #ERR/TST#\
011561 103 052520 042040 EM4: .ASCIZ \CPU DID NOT ISSUE BUS GRANT 7\
011617 103 052520 042040 EM5: .ASCIZ \CPU DID NOT ISSUE BUS GRANT 6\
011655 103 052520 042040 EM6: .ASCIZ \CPU DID NOT ISSUE BUS GRANT 5\
011713 103 052520 042040 EM7: .ASCIZ \CPU DID NOT ISSUE BUS GRANT 4\
011751 117 042516 047440 EM10: .ASCIZ \ONE OR MORE DEVICES DID NOT INTERRUPT\
012017 124 044510 020123 DH10: .ASCII \THIS IS THE ORDER IN WHICH THEY INTERRUPTED\<15><12>
012074 020040 051461 020124 .ASCIZ \ 1ST 2ND 3RD 4TH $ERRPC #ERR/TST#\
012155 102 051525 040440 EM11: .ASCIZ \BUS ADDRESS LINES <A17:A14> DID NOT FUNCTION PROPERLY\
012243 102 030505 051103 DH11: .ASCIZ \BE1CR1 BE1CR2 BE1BA $ERRPC #ERR/TST#\
012314 050103 020125 047516 EM12: .ASCIZ \CPU NO SACK TIMEOUT LOGIC FAILED(TO NEGATE BUS GRANT)\
012402 042502 041461 030522 DH12: .ASCIZ \BE1CR1 BE1CR2 $ERRPC #ERR/TST#\
012443 103 052520 042040 EM13: .ASCIZ \CPU DID NOT PROPERLY EXECUTE AN ACLO/DCLO SEQUENCE\
012526 042444 051122 041520 DH13: .ASCIZ \$ERRPC #ERR/TST#\
012547 103 052520 042040 EM14: .ASCIZ \CPU DID NOT TRAP FROM BUS PARITY ERROR PA/PB = 0/1\
012632 042504 020126 020061 EM15: .ASCII \DEV 1 DID DATIP WITH ROL ON DATOB TO MEMORY\<15><12>
012707 124 042510 052040 .ASCIZ \THE TRANSFER TO THE FOLLOWING LOCATION WAS INCORRECT\
012775 115 046505 051117 DH15: .ASCII \MEMORY ACTUAL CORRECT\<15><12>
013026 046040 041517 020040 .ASCIZ \ LOC DATA DATA $ERRPC #ERR/TST# $ICNT #\
013107 104 053105 041511 EM16: .ASCIZ \DEVICE 3'S DATO TO MEMORY DID NOT EQUAL PATTERN IN R3\
013175 104 053105 041511 EM17: .ASCIZ \DEVICE 4'S DATO TO MEMORY DID NOT EQUAL PATTERN IN R4\
013263 104 053105 041511 EM20: .ASCIZ \DEVICE 1 DOING FUN 1-NPR-DATIP; INCORRECT PATTERN IN MEMORY\
013357 104 053105 041511 EM21: .ASCIZ \DEVICE 2 DOING FUN 2-NPR-DATOB; INCORRECT PATTERN IN MEMORY\
013453 102 052111 033440 EM22: .ASCIZ \BIT 7 OF CR2 SET-CPU DID NOT TIME OUT WITH SACK INHIBITED\
013545 104 053105 021440 DH22: .ASCIZ \DEV # PC $ERRPC #ERR/TST#\
013607 102 052111 030440 EM23: .ASCIZ \BIT 11 OF CR2 SET-NO SSYN ON INTR SIGNAL\
013660 044502 020124 020065 EM24: .ASCIZ \BIT 5 OF CR2 SET-RECEIVED WRONG GRANT\
013726 044502 020124 020066 EM25: .ASCIZ \BIT 6 OF CR2 SET-BUS LATE\

```

```

013760 J44502 020124 020070 EM26: .ASCIZ \BIT 8 OF CR2 SET-NO SSYN OCCURRED\
014022 044502 020124 020071 EM27: .ASCIZ \BIT 9 OF CR2 SET-WRONG ADDRESS ON BUS\
014071 102 052111 030440 EM30: .ASCIZ \BIT 10 OF CR2 SET-DEVICE RECEIVED OTHER THAN ONE GRANT\
014160 045502 047107 020104 EM31: .ASCII \BKGND ROUTINE INSTRUCTIONS OF NEGB'S WERE NOT DONE\<15><12>
014244 047503 051122 041505 .ASCIZ \CORRECTLY TO $REG1 DURING MULTITRANSFERS II\
014320 041501 052524 046101 DH31: .ASCII \ACTUAL CORR'T \<15><12>
014342 040504 040524 020040 .ASCIZ \DATA DATA $ERRPC #ERR/TST# $ICNT #\
014413 104 053105 031440 EM32: .ASCIZ \DEV 3 DID DATI BUT HAS INCORRECT VALUES IN DATA REG\
014477 104 053105 032040 EM33: .ASCIZ \DEV 4 DID NOT INTR THE CORRECT # OF TIMES\
014551 114 051501 020124 EM34: .ASCII \LAST DATIP TO DEVICE 1 DB WAS INCORRECT- EITHER DEVICE DID\<15><12>
014645 116 052117 053440 .ASCIZ \NOT WORK OR BUFFER AREA WAS NOT SET UP PROPERLY\
014725 015 041412 052520 EM35: .ASCIZ <15><12>\CPU TRAPPED THRU LOC 0 TO CATCH IMPROPERLY LOADED VECTORS\
015022 015022 .EVEN
015022 001166 001116 001102 DT1: .WORD $REG2,$ERRPC,$STSTM,0
015032 001162 001164 001166 DT2: .WORD $REG0,$REG1,$REG2,$REG3,$REG4,$ERRPC,$STSTM,0
015052 001162 001164 001166 DT3: .WORD $REG0,$REG1,$REG2,$REG3,$ERRPC,$STSTM,0
015070 001164 001166 001170 DT10: .WORD $REG1,$REG2,$REG3,$REG4,$ERRPC,$STSTM,0
015106 001164 001166 001170 DT11: .WORD $REG1,$REG2,$REG3,$ERRPC,$STSTM,0
015122 001162 001164 001116 DT12: .WORD $REG0,$REG1,$ERRPC,$STSTM,0
015134 001116 001102 000000 DT13: .WORD $ERRPC,$STSTM,0
015142 001162 001164 001170 DT15: .WORD $REG0,$REG1,$REG3,$ERRPC,$STSTM,$ICNT,0
015160 001206 001174 001116 DT22: .WORD $TMP4,$REG5,$ERRPC,$STSTM,0
015172 001164 001166 001116 DT31: .WORD $REG1,$REG2,$ERRPC,$STSTM,$ICNT,0

```

(2)

1784
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

```

:*****
:*****
:*****
.SBTTL END OF PASS ROUTINE
:*****
*INCREMENT THE PASS NUMBER ($PASS)
*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO TST1

$EOP:
SCOPE
CLR $STSTM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER

$ENDCT: .WORD 1

;;65$: .ASCIZ <12><15>/END PASS #/
64$: MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE ,67$ ;;TYPE ASCIZ STRING
BR 66$ ;;GET OVER THE ASCIZ

```

```
1816      ;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
1817 015350 66$:
1818 015350 013746 001112      MOV      $ERTTL,-(SP)      ;;SAVE $ERTTL FOR TYPEOUT
1819      TYPDS      ;;TOTAL NUMBER OF ERRORS
1820 015354 104405      TYPE      , $CRLF      ;;GO TYPE--DECIMAL ASCII WITH SIGN
1821 015356 104401 001223      CLR      $ERTTL      ;;TYPE CARRIAGE RETURN. LINE FEED
1822 015362 005037 001112      $GET42: MOV      @#42,R0      ;;CLEAR ERROR TOTAL
1823 015366 013700 000042      BEQ      $DOAGN      ;;GET MONITOR ADDRESS
1824 015372 001405      RESET      ;;BRANCH IF NO MONITOR
1825 015374 000005      $ENDAD: JSR      PC,(R0)      ;;CLEAR THE WORLD
1826 015376 004710      NOP      ;;GO TO MONITOR
1827 015400 000240      NOP      ;;SAVE ROOM
1828 015402 000240      NOP      ;;FOR
1829 015404 000240      NOP      ;;ACT11
1830 015406      $DOAGN:
1831 015406 000137      JMP      @(PC)+      ;;RETURN
1832 015410 003252      $RTNAD: .WORD      TST1
1833 015412 377 377 000 $ENULL: .BYTE      -1,-1,0      ;;NULL CHARACTER STRING
1834      .EVEN
1835      .SBITL SCOPE HANDLER ROUTINE
1836
1837      ;*****
1838      ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1839      ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1840      ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1841      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1842      ;*SW14=1 LOOP ON TEST
1843      ;*SW11=1 INHIBIT ITERATIONS
1844      ;*SW09=1 LOOP ON ERROR
1845      ;*SW08=1 LOOP ON TEST IN SWR<5:0>
1846      ;*CALL
1847      ;* SCOPE      ;;SCOPE=10T
1848
1849 015416      $SCOPE:
1850 015416 032777 040000 163514 1$: BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
1851 015424 001114      BNE      $OVER      ;;YES IF SW14=1
1852      ;*****START OF CODE FOR THE XOR TESTER*****
1853 015426 000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
1854      ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
1855 015430 013746 000004      MOV      @#ERRVEC,-(SP)      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1856 015434 012737 015454 000004      MOV      #5$,@#ERRVEC      ;;SET FOR TIMEOUT
1857 015442 005737 177060      TST      @#177060      ;;TIME OUT ON XOR?
1858 015446 012637 000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
1859 015452 000466      BR      $SVLAD      ;;GO TO THE NEXT TEST
1860 015454 022626      5$: CMP      (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
1861 015456 012637 000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
1862 015462 000426      BR      7$      ;;LOOP ON THE PRESENT TEST
1863 015464      6$:;*****END OF CODE FOR THE XOR TESTER*****
1864 015464 032777 000400 163446      BIT      #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
1865 015472 001407      BEQ      2$      ;;BR IF NO
1866 015474 017746 163440      MOV      @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
1867 015500 042716 000300      BIC      #5$SWRMK,(SP)      ;;STRIP AWAY UNDESIRED BITS
1868 015504 122637 001102      CMPB      (SP)+,$STNM      ;;ON THE RIGHT TEST?
1869 015510 001462      BEQ      $OVER      ;;BR IF YES
1870 015512 105737 001103      2$: TSTB      $ERFLG      ;;HAS AN ERROR OCCURRED?
1871 015516 001421      BEQ      3$      ;;BR IF NO
```



```

1872 015520 123737 001115 001103      CMPB   $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1873 015526 101015                BHI    3$              ;;BR IF NO
1874 015530 032777 001000 163402      BIT    #BIT09,@SWR    ;;LOOP ON ERROR?
1875 015536 001404                BEQ    4$              ;;BR IF NO
1876 015540 013737 001110 001106 7$:   MOV    $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
1877 015546 000443                BR     $OVER
1878 015550 105037 001103      4$:   CLRB   $ERFLG        ;;ZERO THE ERROR FLAG
1879 015554 005037 001212      CLR    $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1880 015560 000415                BR     1$              ;;ESCAPE TO THE NEXT TEST
1881 015562 032777 004000 163350 3$:   BIT    #BIT11,@SWR    ;;INHIBIT ITERATIONS?
1882 015570 001011                BNE    1$              ;;BR IF YES
1883 015572 005737 001100      TST    $PASS         ;;IF FIRST PASS OF PROGRAM
1884 015576 001406                BEQ    1$              ;;      INHIBIT ITERATIONS
1885 015600 005237 001104      INC    $ICNT         ;;INCREMENT ITERATION COUNT
1886 015604 023737 001212 001104      CMP    $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
1887 015612 002021                BGE    $OVER         ;;BR IF MORE ITERATION REQUIRED
1888 015614 012737 000001 001104 1$:   MOV    #1,$ICNT      ;;REINITIALIZE THE ITERATION COUNTER
1889 015622 013737 015672 001212      MOV    $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DC
1890 015630 105237 001102      $SVLAD: INCB   $TSTNM     ;;COUNT TEST NUMBERS
1891 015634 011637 001106      MOV    (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
1892 015640 011637 001110      MOV    (SP),$LPERR   ;;SAVE ERROR LOOP ADDRESS
1893 015644 005037 001214      CLR    $ESCAPE       ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1894 015650 112737 000001 001115      MCVB  #1,$ERMAX     ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1895 015656 013777 001102 163256 $OVER: MOV    $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
1896 015664 013716 001106      MOV    $LPADR,(SP)  ;;FUDGE RETURN ADDRESS
1897 015670 000002                RTI
1898 015672 000040      $MXCNT: 40          ;;FIXES PS
1899                                ;;MAX. NUMBER OF ITERATIONS
1900                                .SBTTL  ERROR HANDLER ROUTINE
1901                                ;:*****
1902                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1903                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1904                                ;*AND GO TO $ERRTYP ON ERROR
1905                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1906                                ;*SW15=1      HALT ON ERROR
1907                                ;*SW13=1      INHIBIT ERROR TYPEOUTS
1908                                ;*SW10=1      BELL ON ERROR
1909                                ;*SW09=1      LOOP ON ERROR
1910                                ;*CALL
1911                                ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1912
1913 015674                $ERROR:
1914 015674 105237 001103      7$:   INCB   $ERFLG        ;;SET THE ERROR FLAG
1915 015700 001775                BEQ    7$              ;;DON'T LET THE FLAG GO TO ZERO
1916 015702 013777 001102 163232      MOV    $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
1917 015710 032777 002000 163222      BIT    #BIT10,@SWR    ;;BELL ON ERROR?
1918 015716 001402                BEQ    1$              ;;NO - SKIP
1919 015720 104401 001216      TYPE  ,SBELL         ;;RING BELL
1920 015724 005237 001112      1$:   INC    $ERTTL        ;;COUNT THE NUMBER OF ERRORS
1921 015730 011637 001116      MOV    (SP),$ERRPC   ;;GET ADDRESS OF ERROR INSTRUCTION
1922 015734 162737 000002 001116      SUB    #2,$ERRPC
1923 015742 117737 163150 001114      MCVB  @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
1924 015750 032777 020000 163162      BIT    #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
1925 015756 001004                BNE    20$            ;;SKIP TYPEOUTS
1926 015760 004737 016042      JSR    PC,$ERRTYP    ;;GO TO USER ERROR ROUTINE
1927 015764 104401 001223      TYPE  ,$CRLF
    
```

```

1928 015770          20$:
1929 015770 005777 163144 2$:  TST      @SWR          ;;HALT ON ERROR
1930 015774 100001          BPL      3$           ;;SKIP IF CONTINUE
1931 015776 000000          HALT                    ;;HALT ON ERROR!
1932 016000 032777 001000 163132 3$:  BIT      #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
1933 016006 001402          BEQ      4$           ;;BR IF NO
1934 016010 013716 001110          MOV      $LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
1935 016014 005737 001214          4$:  TST      $ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
1936 016020 001402          BEQ      5$           ;;BR IF NONE
1937 016022 013716 001214          MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
1938 016026          5$:
1939 016026 022737 015376 000042          CMP      #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
1940 016034 001001          BNE      6$           ;;BRANCH IF NO
1941 016036 000000          HALT                    ;;YES
1942 016040          6$:
1943 016040 000002          RTI                      ;;RETURN
1944          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
1945
1946          ;*****
1947          ;*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
1948          ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
1949          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
1950
1951          $ERRTYP:

```

```

1952 016042 104401 001223          TYPE     , $CRLF      ;;'CARRIAGE RETURN' & 'LINE FEED'
1953 016046 010046          MOV      RO,-(SP)    ;;SAVE RO
1954 016050 005000          CLR      RO          ;;PICKUP THE ITEM INDEX
1955 016052 153700 001114          BISB    @#$ITEMB,RO
1956 016056 001004          BNE      1$           ;;IF ITEM NUMBER IS ZERO, JUST
1957          ;;TYPE THE PC OF THE ERROR
1958 016060 013746 001116          MOV      $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
1959          ;;ERROR ADDRESS
1960 016064 104402          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1961 016066 000426          BR      6$           ;;GET OUT
1962 016070 005300          1$:  DEC      RO          ;;ADJUST THE INDEX SO THAT IT WILL
1963 016072 006300          ASL      RO          ;;WORK FOR THE ERROR TABLE
1964 016074 006300          ASL      RO
1965 016076 006300          ASL      RO
1966 016100 062700 001226          ADD      #ERRTB,RO    ;;FORM TABLE POINTER
1967 016104 012037 016114          MOV      (RO)+,2$    ;;PICKUP 'ERROR MESSAGE' POINTER
1968 016110 001404          BEQ      3$           ;;SKIP TYPEOUT IF NO POINTER
1969 016112 104401          TYPE                    ;;TYPE THE 'ERROR MESSAGE'
1970 016114 000000          2$:  .WORD    0          ;;'ERROR MESSAGE' POINTER GOES HERE
1971 016116 104401 001223          TYPE     , $CRLF      ;;'CARRIAGE RETURN' & 'LINE FEED'
1972 016122 012037 016132          3$:  MOV      (RO)+,4$    ;;PICKUP 'DATA HEADER' POINTER
1973 016126 001404          BEQ      5$           ;;SKIP TYPEOUT IF 0
1974 016130 104401          TYPE                    ;;TYPE THE 'DATA HEADER'
1975 016132 000000          4$:  .WORD    0          ;;'DATA HEADER' POINTER GOES HERE
1976 016134 104401 001223          TYPE     , $CRLF      ;;'CARRIAGE RETURN' & 'LINE FEED'
1977 016140 011000          5$:  MOV      (RO),RO     ;;PICKUP 'DATA TABLE' POINTER
1978 016142 001004          BNE      7$           ;;GO TYPE THE DATA
1979 016144 012600          6$:  MOV      (SP)+,RO    ;;RESTORE RO
1980 016146 104401 001223          TYPE     , $CRLF      ;;'CARRIAGE RETURN' & 'LINE FEED'
1981 016152 000207          RTS      PC          ;;RETURN
1982 016154          7$:
1983 016154 013046          MOV      @ (RO)+,-(SP) ;;SAVE @ (RO)+ FOR TYPEOUT

```

```

1984 016156 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1985 016160 005710          TST      (R0)    ;;IS THERE ANOTHER NUMBER?
1986 016162 001770          BEQ      6$      ;;BR IF NO
1987 016164 104401 016172  TYPE      8$     ;;TYPE TWO(2) SPACES
1988 016170 000771          BR       7$     ;;LOOP
1989 016172 020040 000      8$:  .ASCIZ  / /     ;;TWO(2) SPACES
1990          016176          .EVEN
1991          .SBTTL  TTY INPUT ROUTINE
1992
1993          ;;*****
1994          .ENABL  LSB
1995
1996          .DSABL  LSB
1997
1998
1999          ;;*****
2000          ;;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2001          ;;*CALL:
2002          ;;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
2003          ;;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
2004          ;;*                  ;;WITH PARITY BIT STRIPPED OFF
2005          ;;*
2006
2007 016176 011646          $RDCHR: MOV    (SP),-(SP)    ;;PUSH DOWN THE PC
2008 016200 016666 000004 000002  MOV    4(SP),2(SP)    ;;SAVE THE PS
2009 016206 105777 162732 1$:    TSTB   @STKS          ;;WAIT FOR
2010 016212 100375          BPL     1$         ;;A CHARACTER
2011 016214 117766 162726 000004  MOVB   @STKB,4(SP)    ;;READ THE TTY
2012 016222 042766 177600 000004  BIC    #'C<177>,4(SP) ;;GET RID OF JUNK IF ANY
2013 016230 026627 000004 000023  CMP    4(SP),#23     ;;IS IT A CONTROL-S?
2014 016236 001013          BNE    3$         ;;BRANCH IF NO
2015 016240 105777 162700 2$:    TSTB   @STKS          ;;WAIT FOR A CHARACTER
2016 016244 100375          BPL     2$         ;;LOOP UNTIL ITS THERE
2017 016246 117746 162674          MOVB   @STKB,-(SP)   ;;GET CHARACTER
2018 016252 042716 177600          BIC    #'C177,(SP)  ;;MAKE IT 7-BIT ASCII
2019 016256 022627 000021          CMP    (SP)+,#21    ;;IS IT A CONTROL-Q?
2020 016262 001366          BNE    2$         ;;IF NOT DISCARD IT
2021 016264 000750          BR     1$         ;;YES, RESUME
2022 016266 026627 000004 000140 3$:    CMP    4(SP),#140   ;;IS IT UPPER CASE?
2023 016274 002407          BLT    4$         ;;BRANCH IF YES
2024 016276 026627 000004 000175  CMP    4(SP),#175   ;;IS IT A SPECIAL CHAR?
2025 016304 003003          BGT    4$         ;;BRANCH IF YES
2026 016306 042766 000040 000004  BIC    #40,4(SP)    ;;MAKE IT UPPER CASE
2027 016314 000002          4$:    RTI          ;;GO BACK TO USER
2028          ;;*****
2029          ;;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2030          ;;*CALL:
2031          ;;*      RDLIN          ;;INPUT A STRING FROM THE TTY
2032          ;;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2033          ;;*                  ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2034
2035 016316 010346          $RDLIN: MOV    R3,-(SP)    ;;SAVE R3
2036 016320 012703 016424 1$:    MOV    #$TTYIN,R3    ;;GET ADDRESS
2037 016324 022703 016442 2$:    CMP    #$TTYIN+16,R3 ;;BUFFER FULL?
2038 016330 101405          BLOS   4$         ;;BR IF YES
2039 016332 104406          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY

```

```

2040 016334 112613          MOV      (SP)+,(R3)      ;;GET CHARACTER
2041 016336 122713 000177 10$:  CMPB     #177,(R3)      ;;IS IT A RUBOUT
2042 016342 001003          BNE      3$             ;;SKIP IF NOT
2043 016344 104401 001222 4$:  TYPE     ,SQUES       ;;TYPE A '?'
2044 016350 000763          BR       1$             ;;CLEAR THE BUFFER AND LOOP
2045 016352 111337 016422 3$:  MOV      (R3),9$        ;;ECHO THE CHARACTER
2046 016356 104401 016422     TYPE     ,9$
2047 016362 122723 000015     CMPB     #15,(R3)+     ;;CHECK FOR RETURN
2048 016366 001356          BNE      2$             ;;LOOP IF NOT RETURN
2049 016370 105063 177777     CLRB     -1(R3)        ;;CLEAR RETURN (THE 15)
2050 016374 104401 001224     TYPE     ,S$F         ;;TYPE A LINE FEED
2051 016400 012603          MOV      (SP)+,R3      ;;RESTORE R3
2052 016402 011646          MOV      (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2053 016404 016666 000004 000002 MOV      4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
2054 016412 012766 016424 000004 MOV      #$TTYIN,4(SP)
2055 016420 000002          RTI                    ;;RETURN
2056 016422 000          9$:  .BYTE    0             ;;STORAGE FOR ASCII CHAR. TO TYPE
2057 016423 000          .BYTE    0             ;;TERMINATOR
2058 016424 000016     $TTYIN: .BLKB    16     ;;RESERVE 16 BYTES FOR TTY INPUT
2059 016442 052536 005015 000     $CNTLU: .ASCIZ  /^U/<15><12> ;;CONTROL 'U'
2060 016447 136 006507 000012     $CNTLG: .ASCIZ  /^G/<15><12> ;;CONTROL 'G'
2061 016454 005015 053523 020122     $MSWR:  .ASCIZ  <15><12>/SWR = /
2062 016462 020075 000
2063 016465 040 047040 053505     $MNEW:  .ASCIZ  / NEW = /
2064 016472 036440 000040
2065          .SBTTL  ROUTINE TO SIZE MEMORY
2066
2067          ;;*****
2068          ;;CALL:
2069          ;;* JSR PC,$SIZE
2070          ;;* RETURN
2071          ;;*$LSTAD WILL CONTAIN:
2072          ;;* WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
2073          ;;* WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
2074          ;;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
2075          ;;*$KT11 IS THE MEMORY MANAGEMENT KEY
2076          ;;*$BIT07 = 0 DON'T USE MEMORY MANAGEMENT
2077          ;;* MUST BE SETUP BEFORE THE CALL
2078          ;;*$BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
2079          ;;* DETERMINED BY ROUTINE
2080
2081 016476 010046     $SIZE: MOV      R0,-(SP)    ;;SAVE R0 ON THE STACK
2082 016500 010146     MOV      R1,-(SP)    ;;SAVE R1 ON THE STACK
2083 016502 010246     MOV      R2,-(SP)    ;;SAVE R2 ON THE STACK
2084 016504 010346     MOV      R3,-(SP)    ;;SAVE R3 ON THE STACK
2085 016506 010446     MOV      R4,-(SP)    ;;SAVE R4 ON THE STACK
2086 016510 013746 000114     MOV      @#114,-(SP)  ;;SAVE MEMORY ERROR VECTOR PS & PC
2087 016514 013746 000116     MOV      @#116,-(SP)
2088 016520 012737 000116 000114     MOV      #116,@#114  ;;IGNORE PARITY ERRORS WHILE SIZING
2089 016526 012737 000002 000116     MOV      #RTI,@#116
2090 016534 013746 000004     MOV      @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
2091 016540 013746 000006     MOV      @#ERRVEC+2,-(SP)
2092 016544 010600     MOV      SP,R0        ;;SAVE THE STACK POINTER
2093          ;;SET THE ERRVEC PS TO THE PRESENT PS
2094 016546 104400     TRAP                    ;;PUSH OLD PSW AND PC ON STACK
2095 016550 012637 000006     MOV      (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2

```

BK001

```

2096 016554 012701 003776      MOV      #3776,R1      ;; SETUP ADDRESS
2097 016560 105727      TSTB     (PC)+        ;; USE MEMORY MANAGEMENT?
2098 016562 000200      SKT11:  .WORD      200      ;; SET TO USE MEMORY MANAGEMENT
2099 016564 100145      BPL      SCORE        ;; BR IF NO
2100 016566 012737 017072 000004      MOV      #SKTNEX,@ERRVEC ;; SET FOR TIMEOUT
2101 016574 005737 177572      TST      @SR0         ;; KT11 ARE YOU THERE?
2102 016600 052737 100000 016562      BIS      #100000,SKT11 ;; YES--SET KT11 KEY
2103 016606 012737 016636 000004      MOV      #100$,@ERRVEC ;; SET FOR TIMEOUT BK001
2104 016614 005737 170200      TST      @170200      ;; UNIBUS MAP ARE YOU THERE? BK001
2105 016620 012737 176200 016656      MOV      #176200,@SSTOP ;; YES-SET COMPARISON VALUE FOR 11/70 BK001
2106 016626 012737 000200 016654      MOV      #200,@SMAP    ;; TURN ON MAP INDICATOR BK001
2107 016634 000411      BR       $MAPRG        ;; GO SET UP MAP REGISTERS BK001
2108 016636 012737 006200 016656 100$:  MOV      #6200,@SSTOP  ;; COMPARISON VALUE FOR 18 BIT MAPPING BK001
2109 016644 022626      CMP      (SP)+,(SP)+  ;; CLEAN OFF STACK BK001
2110 016646 005037 016654      CLR      @SMAP        ;; MAKE SURE MAP INDICATOR TURNED OFF BK001
2111 016652 000412      BR       $NOMAP       ;; BK001
2112 016654 000000      $MAP:   .WORD      0      ;; =200 IF MAP PRESENT BK001
2113 016656 000000      $STOP:  .WORD      0      ;; FILLED WITH APPROPRIATE COMPARISON VALUE BK001
2114 016660 012703 000037      $MAPRG: MOV      #37,R3      ;; SET UP COUNTER BK001
2115 016664 012702 170200      MOV      #170200,R2   ;; START WITH MAPLO BK001
2116 016670 005022      100$:  CLR      (R2)+      ;; LOAD ALL MAP REGISTERS BK001
2117 016672 012722 000074      MOV      #74,(R2)+   ;; WITH THE VALUE 17000000 BK001
2118 016676 077304      SOB      R3,100$     ;; DO ALL 31 REGISTERS BK001
2119 016700      $NOMAP:
2120 016700 005046      CLR      -(SP)        ;; INITIALIZE FOR 'PAR' LOADING
2121 016702 012702 172340      MOV      #KIPAR0,R2   ;; ADDRESS OF FIRST 'PAR'
2122 016706 012703 000010      MOV      #^D8,R3     ;; LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
2123 016712 012762 077406 177740 1$:   MOV      #77406,-40(R2) ;; PDR = 4K, UP, READ/WRITE
2124 016720 011622      MOV      (SP),(R2)+   ;; LOAD 'PAR'
2125 016722 062716 000200      ADD      #200,(SP)    ;; UPDATE FOR NEXT 'PAR'
2126 016726 077307      SOB      R3,1$       ;; LOOP UNTIL ALL EIGHT ARE LOADED
2127 016730 012742 177600      MOV      #177600,-(R2) ;; SETUP KIPAR7 FOR I/O
2128 016734 005042      CLR      -(R2)        ;; SETUP KIPAR6 FOR TESTING
2129 016736 012737 016754 000004      MOV      #2$,@ERRVEC  ;; CATCH TIMEOUT IF NO SR3
2130 016744 012737 000060 172516      MOV      #60,@SR3    ;; ENABLE 22 BIT MODE AND UNIBUS MAP BK001
2131 016752 000401      BR       3$          ;; THIS PDP-11 HAS A SR3 REGISTER
2132 016754 022626      2$:   CMP      (SP)+,(SP)+  ;; CLEAN OFF THE STACK--NO SR3
2133 016756 005237 177572      3$:   INC      @SR0         ;; TURN ON MEMORY MANAGEMENT
2134 016762 012737 017030 000004      MOV      #SKTOUT,@ERRVEC ;; SET FOR TIME OUT
2135 016770 105737 016654      TSTB     @SMAP        ;; IS MAP THERE? BK001
2136 016774 100006      BPL      4$          ;; NO-SKIP BK001
2137 016776 012737 017052 000114      MOV      #SMMOUT,@114 ;; SET UP MEMORY ERROR VECTOR BK001
2138 017004 013737 000006 000116      MOV      @ERRVEC+2,@116 ;; LOCK OUT INTERRUPTS BK001
2139 017012 005737 143776      4$:   TST      @143776   ;; TRAP ON NON-EX-MEM
2140 017016 062712 000040      ADD      #40,(R2)    ;; MAKE A 1K STEP
2141 017022 023712 016656      CMP      @SSTOP,(R2) ;; LAST ONE?
2142 017026 101371      BHI      4$          ;; NO--TRY IT
2143 017030 011202      SKTOUT: MOV      (R2),R2   ;; GET LAST BANK+1
2144 017032 005037 177572      CLR      @SR0        ;; TURN OFF MEMORY MANAGEMENT
2145 017036 105737 016654      TSTB     @SMAP        ;; IS MAP THERE? BK001
2146 017042 100034      BPL      $SIZEX      ;; NO-SKIP BK001
2147 017044 005037 172516      CLR      @SR3        ;; TURN OFF MAP BK001
2148 017050 000431      BR       $SIZEX
2149 017052 013704 177744      $MMOUT: MOV      @177744,R4  ;; SAVE MEMORY ERROR REGISTER BK001
2150 017056 010437 177744      MOV      R4,@177744  ;; CLEAR BITS IN REGISTER BK001
2151 017062 032704 000001      BIT      #1,R4       ;; MEMORY TIMEOUT? BK001
    
```

```

2152 017066 001360      BNE      $SKTOUT      ;;YES-EXIT      BK001
2153 017070 000002      RTI              ;;MUST BE PARITY ERROR-IGNORE IT  BK001
2154 017072 042737 100000 016562 $KTNEX: BIC      #100000,$SKT11  ;;KT11 NON-EXISTENT
2155 017100 012737 017130 000004 $SCORE: MOV      #$SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
2156 017106 005002      CLR      R2          ;;SET UP BANK
2157 017110 062701 004000 1$:      ADD      #4000,R1  ;;INCREMENT BY 1K
2158 017114 062702 000040      ADD      #40,R2     ;;1K STEP
2159 017120 005711      TST     (R1)        ;;TRAP ON TIME OUT
2160 017122 022701 177776      CMP     #177776,R1  ;;LAST ONE
2161 017126 001370      BNE     1$         ;;NO--TRY AGAIN
2162 017130 162701 004000 $SCROUT: SUB     #4000,R1
2163 017134 162702 000040 $SIZEX: SUB     #40,R2  ;;DROP BACK
2164 017140 010006      MOV     R0,SP      ;;RESTORE THE STACK
2165 017142 012637 000006      MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
2166 017146 012637 000004      MOV     (SP)+,@#ERRVEC
2167 017152 012637 000116      MOV     (SP)+,@#116  ;;RESTORE MEMORY ERROR VECTOR
2168 017156 012637 000114      MOV     (SP)+,@#114
2169 017162 010137 017206      MOV     R1,$LSTAD  ;;LAST ADDRESS
2170 017166 010237 017210      MOV     R2,$LSTBK  ;;LAST BANK
2171 017172 012604      MOV     (SP)+,R4   ;;RESTORE R4      BK001
2172 017174 012603      MOV     (SP)+,R3   ;;RESTORE R3
2173 017176 012602      MOV     (SP)+,R2   ;;RESTORE R2
2174 017200 012601      MOV     (SP)+,R1   ;;RESTORE R1
2175 017202 012600      MOV     (SP)+,R0   ;;RESTORE R0
2176 017204 000207      RTS     PC
2177 017206 000000 $LSTAD: .WORD 0  ;;CONTAINS THE LAST ADDRESS
2178 017210 000000 $LSTBK: .WORD 0  ;;CONTAINS THE LAST BANK
2179      .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
2180
2181      ;*****
2182      ;*SAVE R0-R5
2183      ;*CALL:
2184      ;* SAVREG
2185      ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
2186      ;*
2187      ;*TOP---(+16)
2188      ;* +2---(+18)
2189      ;* +4---R5
2190      ;* +6---R4
2191      ;* +8---R3
2192      ;*+10---R2
2193      ;*+12---R1
2194      ;*+14---R0
2195
2196      $SAVREG:
2197 017212 010046      MOV     R0,-(SP)  ;;PUSH R0 ON STACK
2198 017214 010146      MOV     R1,-(SP)  ;;PUSH R1 ON STACK
2199 017216 010246      MOV     R2,-(SP)  ;;PUSH R2 ON STACK
2200 017220 010346      MOV     R3,-(SP)  ;;PUSH R3 ON STACK
2201 017222 010446      MOV     R4,-(SP)  ;;PUSH R4 ON STACK
2202 017224 010546      MOV     R5,-(SP)  ;;PUSH R5 ON STACK
2203 017226 016646 000022      MOV     22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
2204 017232 016646 000022      MOV     22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
2205 017236 016646 000022      MOV     22(SP),-(SP) ;;SAVE PS OF CALL
2206 017242 016646 000022      MOV     22(SP),-(SP) ;;SAVE PC OF CALL
2207 017246 000002      RTI
    
```

2208
 2209
 2210
 2211
 2212 017250
 2213 017250 012666 000022
 2214 017254 012666 000022
 2215 017260 012666 000022
 2216 017264 012666 000022
 2217 017270 012605
 2218 017272 012604
 2219 017274 012603
 2220 017276 012602
 2221 017300 012601
 2222 017302 012600
 2223 017304 000002
 2224
 2225
 2226
 2227
 2228
 2229
 2230
 2231
 2232
 2233
 2234
 2235
 2236
 2237
 2238
 2239
 2240
 2241 017306 105737 001157
 2242 017312 100002
 2243 017314 000000
 2244 017316 000407
 2245 017320 010046
 2246 017322 017600 000002
 2247 017326 112046
 2248 017330 001005
 2249 017332 005726
 2250 017334 012600
 2251 017336 062716 000002
 2252 017342 000002
 2253 017344 122716 000011
 2254 017350 001430
 2255 017352 122716 000200
 2256 017356 001006
 2257 017360 005726
 2258 017362 104401
 2259 017364 001223
 2260 017366 105037 017574
 2261 017372 000755
 2262 017374 004737 017456
 2263 017400 123726 001156

;*RESTORE R0-R5

;*CALL:

;* RESREG

\$RESREG:

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

.SBTTL TYPE ROUTINE

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

;*CALL:

*1) USING A TRAP INSTRUCTION

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

;*OR

* TYPE
 * MESADR

\$TYPE: TSTB \$!PFLG ;;IS THERE A TERMINAL?
 BPL 1\$;;BR IF YES
 HALT ;;HALT HERE IF NO TERMINAL
 BR 3\$;;LEAVE
 1\$: MOV R0,-(SP) ;;SAVE R0
 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
 2\$: MOV (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 BNE 4\$;;BR IF IT ISN'T THE TERMINATOR
 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
 60\$: MOV (SP)+,R0 ;;RESTORE R0
 3\$: ADD #2,(SP) ;;ADJUST RETURN PC
 RTI ;;RETURN
 4\$: CMPB #HT,(SP) ;;BRANCH IF <HT>
 BEQ 8\$
 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
 BNE 5\$
 TST (SP)+ ;;POP <CR><LF> EQUIV
 TYPE ;;TYPE A CR AND LF
 \$CRLF
 CLR B \$CHARCNT ;;CLEAR CHARACTER COUNT
 BR 2\$;;GET NEXT CHARACTER
 5\$: JSR PC,\$TYPEC ;;GO TYPE THIS CHARACTER
 6\$: CMPB \$FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?


```

2320      : *      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2321      : *      .BYTE  M      ;;M=1 OR 0
2322      : *      ;;1=TYPE LEADING ZEROS
2323      : *      ;;0=SUPPRESS LEADING ZEROS
2324      : *
2325      : *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2326      : *$TYPOS OR $TYPOC
2327      : *CALL:
2328      : *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2329      : *      TYPON      ;;CALL FOR TYPEOUT
2330      : *
2331      : *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2332      : *CALL:
2333      : *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2334      : *      TYPOC      ;;CALL FOR TYPEOUT
2335
2336 017600 017646 000000      $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
2337 017604 116637 000001 020023      MOV      1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
2338 017612 112637 020025      MOV      (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
2339 017616 062716 000002      ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
2340 017622 000406      BR      $TYPON
2341 017624 112737 000001 020023      $TYPOC: MOV      #1,SOFILL      ;;SET THE ZERO FILL SWITCH
2342 017632 112737 000006 020025      MOV      #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
2343 017640 112737 000005 020022      $TYPON: MOV      #5,SOCNT      ;;SET THE ITERATION COUNT
2344 017646 010346      MOV      R3,-(SP)      ;;SAVE R3
2345 017650 010446      MOV      R4,-(SP)      ;;SAVE R4
2346 017652 010546      MOV      R5,-(SP)      ;;SAVE R5
2347 017654 113704 020025      MOV      SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2348 017660 005404      NEG      R4
2349 017662 062704 000006      ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
2350 017666 110437 020024      MOV      R4,SOMODE      ;;SAVE IT FOR USE
2351 017672 113704 020023      MOV      SOFILL,R4      ;;GET THE ZERO FILL SWITCH
2352 017676 016605 000012      MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
2353 017702 005003      CLR      R3      ;;CLEAR THE OUTPUT WORD
2354 017704 006105      1$: ROL      R5      ;;ROTATE MSB INTO 'C'
2355 017706 000404      BR      3$      ;;GO DO MSB
2356 017710 006105      2$: ROL      R5      ;;FORM THIS DIGIT
2357 017712 006105      ROL      R5
2358 017714 006105      ROL      R5
2359 017716 010503      MOV      R5,R3
2360 017720 006103      3$: ROL      R3      ;;GET LSB OF THIS DIGIT
2361 017722 105337 020024      DECB      SOMODE      ;;TYPE THIS DIGIT?
2362 017726 100016      BPL      7$      ;;BR IF NO
2363 017730 042703 177770      BIC      #177770,R3      ;;GET RID OF JUNK
2364 017734 001002      BNE      4$      ;;TEST FOR 0
2365 017736 005704      TST      R4      ;;SUPPRESS THIS 0?
2366 017740 001403      BEQ      5$      ;;BR IF YES
2367 017742 005204      4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
2368 017744 052703 000060      BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
2369 017750 052703 000040      5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
2370 017754 110337 020020      MOV      R3,8$      ;;SAVE FOR TYPING
2371 017760 104401 020020      TYPE      ,8$      ;;GO TYPE THIS DIGIT
2372 017764 105337 020022      7$: DECB      SOCNT      ;;COUNT BY 1
2373 017770 003347      BGT      2$      ;;BR IF MORE TO DO
2374 017772 002402      BLT      6$      ;;BR IF DONE
2375 017774 005204      INC      R4      ;;INSURE LAST DIGIT ISN'T A BLANK

```

```

2376 017776 000744          BR      2$          ;;GO DO THE LAST DIGIT
2377 020000 012605          6$:  MOV    (SP)+,R5      ;;RESTORE R5
2378 020002 012604          MOV    (SP)+,R4      ;;RESTORE R4
2379 020004 012603          MOV    (SP)+,R3      ;;RESTORE R3
2380 020006 016666 000002 000004  MOV    2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
2381 020014 012616          MOV    (SP)+,(SP)
2382 020016 000002          RTI                    ;;RETURN
2383 020020          000          8$:  .BYTE  0          ;;STORAGE FOR ASCII DIGIT
2384 020021          000          .BYTE  0          ;;TERMINATOR FOR TYPE ROUTINE
2385 020022          000          $OCNT: .BYTE  0      ;;OCTAL DIGIT COUNTER
2386 020023          000          $OFILL: .BYTE  0     ;;ZERO FILL SWITCH
2387 020024 000000          $OMODE: .WORD  0     ;;NUMBER OF DIGITS TO TYPE
2388          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2389
2390          ;;*****
2391          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2392          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2393          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2394          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2395          ;*REPLACED WITH SPACES.
2396          ;*CALL:
2397          ;*      MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
2398          ;*      TYPDS                    ;;GO TO THE ROUTINE
2399
2400          $TYPDS:
2401 020026 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
2402 020030 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
2403 020032 010246          MOV    R2,-(SP)      ;;PUSH R2 ON STACK
2404 020034 010346          MOV    R3,-(SP)      ;;PUSH R3 ON STACK
2405 020036 010546          MOV    R5,-(SP)      ;;PUSH R5 ON STACK
2406 020040 012746 020200          MOV    #2020,-(SP)   ;;SET BLANK SWITCH AND SIGN
2407 020044 016605 000020          MOV    20(SP),R5    ;;GET THE INPUT NUMBER
2408 020050 100004          BPL    1$            ;;BR IF INPUT IS POS.
2409 020052 005405          NEG    R5            ;;MAKE THE BINARY NUMBER POS.
2410 020054 112766 000055 000001          MOVB   #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
2411 020062 005000          1$:  CLR    R0            ;;ZERO THE CONSTANTS INDEX
2412 020064 012703 020242          MOV    #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
2413 020070 112723 000040          MOVB   #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
2414 020074 005002          2$:  CLR    R2            ;;CLEAR THE BCD NUMBER
2415 020076 016001 020232          MOV    $DTBL(R0),R1 ;;GET THE CONSTANT
2416 020102 160105          3$:  SUB    R1,R5        ;;FORM THIS BCD DIGIT
2417 020104 002402          BLT    4$            ;;BR IF DONE
2418 020106 005202          INC    R2            ;;INCREASE THE BCD DIGIT BY 1
2419 020110 000774          BR     3$
2420 020112 060105          4$:  ADD    R1,R5        ;;ADD BACK THE CONSTANT
2421 020114 005702          TST    R2            ;;CHECK IF BCD DIGIT=0
2422 020116 001002          BNE    5$            ;;FALL THROUGH IF 0
2423 020120 105716          TSTB   (SP)          ;;STILL DOING LEADING 0'S?
2424 020122 100407          BMI    7$            ;;BR IF YES
2425 020124 106316          5$:  ASLB   (SP)          ;;MSD?
2426 020126 103003          BCC    6$            ;;BR IF NO
2427 020130 116653 000001 177777          MOVB   1(SP),-1(R3) ;;YES--SET THE SIGN
2428 020136 052702 000060          6$:  BIS    #'0,R2      ;;MAKE THE BCD DIGIT ASCII
2429 020142 052702 000040          7$:  BIS    #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2430 020146 110223          MOVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2431 020150 005720          TST    (R0)+        ;;JUST INCREMENTING

```

```

2432 020152 020027 000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
2433 020156 002746             BLT      2$          ;;GO DO THE NEXT DIGIT
2434 020160 003002             BGT      8$          ;;GO TO EXIT
2435 020162 010502             MOV      R5,R2      ;;GET THE LSD
2436 020164 000764             BR       6$          ;;GO CHANGE TO ASCII
2437 020166 105726             8$: TSTB      (SP)+   ;;WAS THE LSD THE FIRST NON-ZERO?
2438 020170 100003             BPL      9$          ;;BR IF NO
2439 020172 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
2440 020200 105013             9$: CLRB      (R3)   ;;SET THE TERMINATOR
2441 020202 012605             MOV      (SP)+,R5   ;;POP STACK INTO R5
2442 020204 012603             MOV      (SP)+,R3   ;;POP STACK INTO R3
2443 020206 012602             MOV      (SP)+,R2   ;;POP STACK INTO R2
2444 020210 012601             MOV      (SP)+,R1   ;;POP STACK INTO R1
2445 020212 012600             MOV      (SP)+,R0   ;;POP STACK INTO R0
2446 020214 104401 020242      TYPE     $DBLK      ;;NOW TYPE THE NUMBFR
2447 020220 016666 000002 000004  MOV      2(SP),4(SP) ;;ADJUST THE STACK
2448 020226 012616             MOV      (SP)+,(SP)
2449 020230 000002             RTI                          ;;RETURN TO USER
2450 020232 023420             $DTBL: 1000.
2451 020234 001750             1000.
2452 020236 000144             100.
2453 020240 000012             10.
2454 020242 000004             $DBLK: .BLKW 4
2455                                     .SBTTL TRAP DECODER
2456
2457                                     ;*****
2458                                     ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
2459                                     ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2460                                     ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2461                                     ;*GO TO THAT ROUTINE.
2462
2463 020252 010046             $TRAP: MOV      R0,-(SP) ;;SAVE R0
2464 020254 016600 000002      MOV      2(SP),R0    ;;GET TRAP ADDRESS
2465 020260 005740             TST      -(R0)      ;;BACKUP BY 2
2466 020262 111000             MOVB     (R0),R0    ;;GET RIGHT BYTE OF TRAP
2467 020264 006300             ASL      R0         ;;POSITION FOR INDEXING
2468 020266 016000 020306      MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE
2469 020272 000200             RTS      R0         ;;GO TO ROUTINE
2470
2471
2472                                     ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
2473
2474 020274 011646             $TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
2475 020276 016666 000004 000002  MOV      4(SP),2(SP) ;;MOVE THE PSW DOWN
2476 020304 000002             RTI                          ;;RESTORE THE PSW
2477
2478                                     .SBTTL TRAP TABLE
2479
2480                                     ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2481                                     ;*BY THE 'TRAP' INSTRUCTION.
2482
2483                                     : ROUTINE
2484                                     : -----
2485 020306 020274             $TRPAD: .WORD   $TRAP2
2486 020310 017306             $TYPE  ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2487 020312 017624             $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)

```

2488	020314	017600			\$TYPOS	::CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
2489	020316	017640			\$TYPGN	::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
2490	020320	020026			\$TYPDS	::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
2491								
2492								
2493	020322	016176			\$RDCHR	::CALL=RDCHR	TRAP+6(104406)	TTY TYPEIN CHARACTER ROUTINE
2494	020324	016316			\$RDLIN	::CALL=RDLIN	TRAP+7(104407)	TTY TYPEIN STRING ROUTINE
2495	020326	017212			\$SAVREG	::CALL=SAVREG	TRAP+10(104410)	SAVE R0-R5 ROUTINE
2496	020330	017250			\$RESREG	::CALL=RESREG	TRAP+11(104411)	RESTORE R0-R5 ROUTINE
2497					.SBTTL	POWER DOWN AND UP ROUTINES		

```

*****
:POWER DOWN ROUTINE
2501 020332 012737 020472 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;;SET FOR FAST UP
2502 020340 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
2503 020346 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
2504 020350 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
2505 020352 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
2506 020354 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
2507 020356 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
2508 020360 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
2509 020362 017746 160552 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
2510 020366 010637 020476 MOV SP,$SAVR6 ;;SAVE SP
2511 020372 012737 020404 000024 MOV #S:PWUP,@PWRVEC ;;SET UP VECTOR
2512 020400 000000 HALT
2513 020402 000776 BR .-2 ;;HANG UP

```

```

*****
:POWER UP ROUTINE
2517 020404 012737 020472 000024 $PWRUP: MOV #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
2518 020412 013706 020476 MOV $SAVR6,SP ;;GET SP
2519 020416 005037 020476 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
2520 020422 005237 020476 1$: INC $SAVR6 ;;WAIT FOR THE INC
2521 020426 001375 BNE 1$ ;;OF WORD
2522 020430 012677 160504 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
2523 020434 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
2524 020436 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
2525 020440 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
2526 020442 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
2527 020444 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
2528 020446 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
2529 020450 012737 020332 000024 MOV #S:PWDRN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
2530 020456 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
2531 020464 104401 TYPE REPORT THE POWER FAILURE
2532 020466 020500 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
2533 020470 000002 RTI
2534 020472 000000 $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
2535 020474 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
2536 020476 000000 $SAVR6: 0 ;;PUT THE SP HERE
2537 020500 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
2538 020506 000122 .EVEN

```

```

*****
*****
*****
2543 020510 ATEND:

```

UNIBUS EXERCISER
CZKUA.E.P11

27-SEP-79 09:25

MACY11 30A(1052)

04-OCT-79 12:49 PAGE 54
POWER DOWN AND UP ROUTINES

I 5

SEQ 0060

2544

000001

.END

HT	=	000011	35#	2253	2311															
INK		006320	1243	1248	1253	1258#														
INTRN		005250	1048	1053	1058	1063#														
INTR1		005162	1010	1044#																
INTR2		005200	1015	1049#																
INTR3		005216	1020	1054#																
INTR4		005234	1026	1059#																
IOTVEC	=	000020	130#	534*	535*															
KIPAR0	=	172340	185#	2071	2121															
KIPAR1	=	172342	186#																	
KIPAR2	=	172344	187#																	
KIPAR3	=	172346	188#																	
KIPAR4	=	172350	189#																	
KIPAR5	=	172352	190#																	
KIPAR6	=	172354	191#																	
KIPAR7	=	172356	192#																	
KIPDR0	=	172300	174#																	
KIPDR1	=	172302	175#																	
KIPDR2	=	172304	176#																	
KIPDR3	=	172306	177#																	
KIPDR4	=	172310	178#																	
KIPDR5	=	172312	179#																	
KIPDR6	=	172314	180#																	
KIPDR7	=	172316	181#																	
LEEV		010326	1632	1654	1656#															
LF	=	000012	36#	2305	2311															
LOAD1		004734	1008#																	
LOAD2		004772	1015#																	
LOAD3		005024	1020#																	
LOAD4		005056	1025#																	
LODDEV		002314	584#	608	624															
MMVEC	=	000250	141#																	
MOVREG		002350	593#	598																
MOVVEC		002414	603#																	
MULT1		007260	1214	1223	1232	1437#														
NG		003260	719#																	
NODEV		002432	589	611#																
NODO		006050	1165	1200#																
NOG		010416	726	729	732	735	1691#													
NPRTST		003402	752#																	
NXT		005762	1171	1182#	1199															
PBERR		006014	1173	1188#																
PBPSW	=	000116	470#	1175*	1183	1184*														
PBRTN		006036	1174	1196#																
PBVEC	=	000114	469#	1174*	1183*															
PIRQ	=	177772	42#																	
PIRQVE	=	000240	136#																	
PRO	=	000000	59#	775	809	844	879	914	979	1032	1278									
PR1	=	000040	60#																	
PR2	=	000100	61#																	
PR3	=	000140	62#	892	936	1079	1172	1773												
PR4	=	000200	63#	733	857	896	909													
PR5	=	000240	64#	730	822	861	874													
PR6	=	000300	65#	727	787	826	839													
PR7	=	000340	66#	574	579	722	724	753	757	759	770	791	793	804	828					
			863	898	930	933	967	974	1006	1011	1016	1021	1027	1081	1127					

UNIBUS EXERCISER
CZKLA.E.P11 27-SEP-79 09:25

MACY11 30A(1052)

04-OCT-79 12:49 PAGE 63
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0068

\$MNEW	016465	2063#												
\$MSWR	016454	2061#												
\$MXCNT	015672	1889	1898#											
\$NOMAP	016700	2111	2119#											
\$NULL	001154	268#	1137	1147	2265	2311								
\$NWTST=	000001	711#	713	741#	743	778#	780	813#	815	848#	850	883#	885	918#
		920	957#	959	997#	999	1069#	1071	1111#	1113	1152#	1154	1203#	1205
		1265#	1267	1390#										
\$OCNT	020022	2343*	2372*	2385#										
\$OMODE	020024	2338*	2342*	2347	2350*	2361*	2387#							
\$OVER	015656	1851	1869	1877	1887	1895#								
\$PASS	001100	243#	1799*	1800*	1811	1833	1883	1899						
\$POWER	020500	1142	2532	2537#										
\$PWRDN	020332	540	1149	1614	2501#	2529								
\$PWRMG	020466	1137	1142*	1147*	2532#									
\$PWRUP	020404	2511	2517#											
\$QUES	001222	289#	1944	2043	2059	2311								
\$RDCHR	016176	2007#	2493											
\$RDDEC=	***** U	2495												
\$RDLIN	016316	2035#	2494											
\$RDOCT=	***** U	2495												
\$RDSZ =	000016	2028#												
\$REGAD	001160	272#												
\$REGO	001162	274#	518	659*	684*	697	768*	802*	837*	872*	907*	941*	1329*	1348*
		1516*	1539*	1558*	1709*	1783								
\$REG1	001164	275#	769*	803*	838*	873*	908*	942*	987*	1036*	1103*	1328*	1347*	1367*
		1385*	1482*	1483	1515*	1538*	1557*	1710*	1719*	1720	1783			
\$REG2	001166	276#	631*	632*	770*	804*	839*	874*	909*	988*	1037*	1104*	1259*	1320*
		1323	1325*	1330	1368*	1386*	1486	1665*	1666*	1711*	1733*	1774*	1783	
\$REG3	001170	277#	771*	805*	840*	875*	910*	1038*	1105*	1330*	1349*	1517*	1537*	1556*
		1712*	1783											
\$REG4	001172	278#	1039*	1303*	1383	1385	1426*	1713*	1750*	1751*	1752	1783		
\$REG5	001174	279#	594*	950*	1064*	1096*	1189*	1260*	1431*	1783				
\$RESRE	017250	2212#	2496											
\$RTNAD	015410	1832#												
\$R2A =	***** U	2497												
\$SAVRE	017212	2196#	2495											
\$SAVR6	020476	2510*	2518	2519*	2520*	2536#								
\$SCOPE	015416	534	1849#											
\$SETUP=	000037	194#	533	534	536	538	540	542	543	544	546	1797	1850	1914
		1932	1939	1996	2065									
\$SIZE	016476	2081#												
\$SIZEX	017134	2146	2148	2163#										
\$STOP	016656	2105*	2108*	2113#	2141									
\$STUP =	177777	194#												
\$SVLAD	015630	1859	1890#											
\$SVPC =	000232	228#	233											
\$SWR =	167400	2#	14	19	20	21	22	23	24	25	286	287	288	543
		544	546	547	718	751	786	821	856	891	927	966	1006	1077
		1119	1163	1211	1272	1394	1792	1798	1825	1831	1833	1841	1842	1843
		1844	1845	1850	1862	1864	1865	1870	1871	1872	1879	1880	1881	1892
		1895	1898	1905	1906	1907	1908	1909	1917	1924	1929	1932	1944	2533
\$SWRMK=	000300	3#	25	26	1845	1846	1866	1867						
\$TIMES	001212	286#	543*	1119*	1394*	1798*	1879*	1886	1889*	1898				
\$TKB	001146	265#	1396	1994	2011	2017	2284	2291						
\$TKS	001144	264#	1994	2009	2015	2282	2289							

UNIBUS EXERCISER
CZKUAE.P11

MACY11 30A(1052)
27-SEP-79 09:25

04-OCT-79 12:49 PAGE 64
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0069

\$TMP0	001176	204*	221*	280#	568	596*	597	599*	658*	680*	683*	689	702	
\$TMP1	001200	281#	1130*											
\$TMP2	001202	282#												
\$TMP3	001204	283#	1167*	1169*	1170									
\$TMP4	001206	284#	951*	1047*	1052*	1057*	1062*	1097*	1126*	1191*	1242*	1247*	1252*	1257*
		1409*	1416*	1421*	1427*	1586*	1589*	1592*	1595*	1624	1783			
\$TMP5	001210	285#	520											
\$TN	- 000020	14#	711	718#	741	751#	756	778	786#	790	813	821#	825	848
		856#	860	883	891#	895	918	927#	940	944	957	966#	992	994
		997	1006#	1035	1041	1069	1077#	1092	1111	1119#	1121	1143	1152	1163#
		1187	1203	1211#	1216	1221	1230	1235	1265	1272#	1334	1353	1372	1390
		1394#												
\$TPB	001152	267#	2300*	2311										
\$TPFLG	001157	271#	2241	2311										
\$TPS	001150	266#	2298	2311										
\$TRAP	020252	538	2463#											
\$TRAP2	020274	2474#	2485											
\$TRP =	000012	2478#	2487#	2488#	2489#	2490#	2491#	2493	2494#	2495#	2496#	2497#		
\$TRPAD	020306	2468	2485#											
\$TSTNM	001102	244#	1783	1797*	1840	1868	1890*	1895	1899	1916	1944			
\$TTYIN	016424	2036	2037	2054	2058#									
\$TYPBN=	***** U	2491												
\$TYPDS	020026	2400#	2490											
\$TYPE	017306	2241#	2478	2486										
\$TYPEC	017456	2262	2269	2276	2281#									
\$TYPEX	017576	2304	2306	2309#										
\$TYPOC	017624	2341#	2487											
\$TYPON	017640	2340	2343#	2489										
\$TYPOS	017600	2336#	2488											
\$XOFF =	000023	2286	2311											
\$XON =	000021	2293	2311											
\$XTSTR	015426	1853#												
\$SGET4=	000000	1825#												
\$OFILL	020023	2337*	2341*	2351	2386#									
\$QCAT=	***** U	1850	1926											
.	- 020510	196#	200#	203#	220#	228	229#	231#	233#	241#	292	531	546	547
		620#	675#	679#	688#	1783#	1810#	1833	1834#	1898	1899	1944	1990#	1994
		2058#	2059	2065	2311	2454#	2513	2535						

.SACT1	1#	2#	224
.SAPTБ	1#		
.SAPTH	1#		
.SAPTY	1#		
.SASTA	1#		
.SCATC	1#	2#	194
.SCMTA	1#	2#	235
.SDB2D	1#		
.SDB2O	1#		
.SDIV	1#		
.SEOP	1#	2#	1786
.SERRO	1#	2#	1899
.SERRT	1#	2#	1944
.SMULT	1#		
.SPOWE	1#	2#	2497
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#		
.SREAD	1#	2#	1991
.SR2AZ	1#		
.SSAVE	1#	2#	2179
.SSB2D	1#		
.SSB2O	1#		
.SSCOP	1#	2#	1835
.SSIZE	1#	2#	2065
.SSUPR	1#		
.STRAP	1#	2#	2455
.STYPB	1#		
.STYPD	1#	2#	2388
.STYPE	1#	2#	2224
.STYPO	1#	2#	2311
.S4OCA	1#		
.1170	1#		

. ABS. 020510 000

ERRORS DETECTED: 0

CZKUAЕ,CZKUAЕ.LST/CRF/SOL=[400,4531]SYSMAC.C4,[400,2465]CZKUAЕ.P11
RUN-TIME: 42 51 3 SECONDS
RUN-TIME RATIO: 472/98=4.8
CORE USED: 33k (65 PAGES)