

LK11A

PUSH BUTTON MOD DIAG
MD-11-DZLKA-A

EP-DZLKA-A-DL-A

NOV 1976

COPYRIGHT © 1976

digital

FICHE 1 OF 1

MADE IN USA

The microfiche card contains a grid of 48 frames, arranged in 12 rows and 4 columns. Each frame displays technical information, including diagrams, tables, and text. The content is too small to read in detail but appears to be diagnostic or configuration data for the MD-11-DZLKA-A system.

.REM %

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZLKA-A
PRODUCT NAME: LK11A PUSH BUTTON MODULE DIAGNOSTIC
DATE: AUG. 21, 1976
MAINTAINER: DIAGNOSTIC GROUP

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

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

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

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

MAINDEC-11-DZLKA-A
DZLKA.P11
MACY11 27(732) 25-SEP-76 09:47 PAGE 2
IDENTIFICATION
PRODUCT CODE: MAINDEC-11-DZLKA-A
PRODUCT NAME: LK11A PUSH BUTTON MODULE DIAGNOSTIC
DATE: AUG. 21, 1976
MAINTAINER: DIAGNOSTIC GROUP
COPYRIGHT (C) 1976
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130

4.2 STARTING ADDRESS

200 IS STARTING ADDRESS OF THE 25 LOGIC TESTS.
204 IS STARTING ADDRESS OF THE MANUAL INTERVENTION TEST.
210 IS STARTING ADDRESS OF THE ENCODER TEST.

5.0 OPERATING PROCEDURE

THE OPERATOR MUST RUN THE 25 LOGIC TESTS BEFORE RUNNING EITHER OF THE TWO OPERATOR INTERVENTION TESTS.

WHEN STARTING AT 200, THE TEST RUNS WITHOUT OPERATOR INTERVENTION AND PRINTS AN END-OF-PASS MESSAGE WHEN COMPLETED, OR IT WILL HALT IF SW12 IS UP.

WHEN STARTING AT 204, THE OPERATOR IS REQUIRED TO PRESS ONE PUSH BUTTON AT A TIME. THE LIGHT ON THAT BUTTON ONLY SHOULD COMPLEMENT. NO CHANGE SHOULD OCCUR IN ANY OF THE OTHER LIGHTS.

WHEN STARTING AT 210, THE OPERATOR IS REQUIRED TO PRESS ONE PUSH BUTTON AT A TIME. THE LIGHT ON THAT BUTTON ONLY SHOULD TURN ON AND ALL OTHER LIGHTS SHOULD GO OFF.

6.0 ERRORS

THIS PROGRAM USES THE DIAGNOSTIC "SYSMAC" PACKAGE FOR ERROR REPORTING AND TYPEOUT. THE ERROR INFORMATION CONSISTS OF A DESCRIPTIVE MESSAGE ABOUT THE ERROR, THE LOCATION AT WHICH THE ERROR WAS DETECTED, THE ACTUAL DATA VALUE, AND THE EXPECTED DATA VALUE.

7.0 RESTRICTIONS

7.1 THE LOGIC TESTS MUST BE RUN AT LEAST ONE TIME BEFORE RUNNING THE TWO MANUAL INTERVENTION TESTS.

7.2 THIS PROGRAM SUPPORTS ONLY ONE LK11A.

131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186

8.0 MISCELLANEOUS

8.1 EXECUTION TIME

EXECUTION TIME FOR ONE PASS THROUGH THE LOGIC TESTS IS 30 SECONDS.

8.2 DEVICE ADDRESSES LOCATIONS

LOCATION "SBASE" CONTAINS THE LK11A BASE DEVICE ADDRESS <160060>.
LOCATION "SVECT1" CONTAINS THE LK11A INTERRUPT VECTOR <360>.
LOCATION "SVECT1" CONTAINS THE LK11A INTERRUPT LEVEL <200>.

8.3 ACT/XXDP/APT NOTES

THE PROGRAM IS CHAINABLE UNDER "ACT-11" AND "XXDP".

THE SOFTWARE HOOKS FOR "APT" HAVE BEEN PROVIDED BUT HAVE NOT BEEN TESTED.

9.0 PROGRAM DESCRIPTION

9.1 LOGIC TESTS

THERE ARE 25 SEPARATE TESTS WHICH CHECK THE LOGIC OF THE PUSH-BUTTON MODULE. THEY RUN SEQUENTIALLY WITHOUT OPERATOR INTERVENTION AND AN END-OF-PASS MESSAGE IS PRINTED AFTER THE COMPLETION OF EACH PASS, OR A HALT WILL OCCUR IF SW12 IS UP.

9.2 PUSH-BUTTON TEST

THIS TEST REQUIRES OPERATOR INTERVENTION. ITS STARTING ADDRESS IS 204 AND THE TEST MUST BE RUN ONLY AFTER THE LOGIC TESTS HAVE BEEN COMPLETED. THE TEST ITSELF DEPENDS UPON THE OPERATOR TO PRESS A SINGLE BUTTON AND TAKE NOTICE OF THE RESULTS. THE LIGHT WITHIN THE BUTTON THAT WAS PRESSED SHOULD COMPLEMENT; IF IT WAS ON, IT SHOULD GO OFF AND IF IT WAS OFF, IT SHOULD GO ON. NO CHANGE SHOULD OCCUR IN ANY OF THE OTHER 15 LIGHTS.

9.3 ENCODER TEST

THIS TEST REQUIRES OPERATOR INTERVENTION AND ITS STARTING ADDRESS IS 210. THE TEST MUST ONLY BE RUN AFTER THE LOGIC TESTS HAVE BEEN RUN. THE OPERATOR MUST PUSH ONE BUTTON AND THE LIGHT WITHIN THAT BUTTON ONLY SHOULD BE TURNED ON AND WON'T GO OFF UNTIL THE TEST IS RESTARTED.

%
:TITLE MAINDEC-11-DZLKA-A
:#COPYRIGHT (C) 1976
:#DIGITAL EQUIPMENT CORP.
:#MAYNARD, MASS. 01754

```

187      ;*
188      ;*PROGRAM BY VERA BREUER
189      ;*
190      ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
191      ;*PACKAGE (MAINDEC-11-DZQAC-CD),MAR 21, 1976.
192      ;*
193      ;SBTTL  BASIC DEFINITIONS
194
195      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
196      DC1100      STACK= 1100
197      .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
198      .EQUIV  IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
199
200      ;*MISCELLANEOUS DEFINITIONS
201      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
202      000012      LF= 12      ;;CODE FOR LINE FEED
203      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
204      000200      CRLF= 200  ;;CODE FOR CARRIAGE RETURN-LINE FEED
205      177776      PS= 177776 ;;PROCESSOR STATUS WORD
206      .EQUIV  PS,PSW
207      177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
208      177772      PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
209      177570      DSWR= 177570 ;;HARDWARE SWITCH REGISTER
210      177570      DCISP= 177570 ;;HARDWARE DISPLAY REGISTER
211
212      ;*GENERAL PURPOSE REGISTER DEFINITIONS
213      000000      R0= %0      ;;GENERAL REGISTER
214      000001      R1= %1      ;;GENERAL REGISTER
215      000002      R2= %2      ;;GENERAL REGISTER
216      000003      R3= %3      ;;GENERAL REGISTER
217      000004      R4= %4      ;;GENERAL REGISTER
218      000005      R5= %5      ;;GENERAL REGISTER
219      000006      R6= %6      ;;GENERAL REGISTER
220      000007      R7= %7      ;;GENERAL REGISTER
221      .EQUIV  R6,SP      ;;STACK POINTER
222      .EQUIV  R7,PC      ;;PROGRAM COUNTER
223
224      ;*PRIORITY LEVEL DEFINITIONS
225      000000      PR0= 0      ;;PRIORITY LEVEL 0
226      000040      PR1= 40     ;;PRIORITY LEVEL 1
227      000100      PR2= 100    ;;PRIORITY LEVEL 2
228      000140      PR3= 140    ;;PRIORITY LEVEL 3
229      000200      PR4= 200    ;;PRIORITY LEVEL 4
230      000240      PR5= 240    ;;PRIORITY LEVEL 5
231      000300      PR6= 300    ;;PRIORITY LEVEL 6
232      000340      PR7= 340    ;;PRIORITY LEVEL 7
233
234      ;*"SWITCH REGISTER" SWITCH DEFINITIONS
235      100000      SW15= 100000
236      040000      SW14= 40000
237      020000      SW13= 20000
238      010000      SW12= 10000
239      004000      SW11= 4000
240      002000      SW10= 2000
241      001000      SW09= 1000
242      000400      SW08= 400

```

273 000200
274 000100
275 000040
276 000020
277 000010
278 000004
279 000002
280 000001

SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09, SW9
.EQUIV SW08, SW8
.EQUIV SW07, SW7
.EQUIV SW06, SW6
.EQUIV SW05, SW5
.EQUIV SW04, SW4
.EQUIV SW03, SW3
.EQUIV SW02, SW2
.EQUIV SW01, SW1
.EQUIV SW00, SW0

281 100000
282 040000
283 020000
284 010000
285 004000
286 002000
287 001000
288 000400
289 000200
290 000100
291 000040
292 000020
293 000010
294 000004
295 000002
296 000001

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

297 000004
298 000010
299 000014
300 000014
301 000014
302 000020
303 000024
304 000030

.*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
TRITVEC= 14 ;: "I" BIT
TRAVEC= 14 ;: TRACE TRAP
BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 ;: POWER FAIL
EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**

H01

299	000034	TRAPVEC=34	::"TRAP" TRAP
300	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
301	000064	TPVEC= 64	::TTY PRINTER VECTOR
302	000240	PIRQVEC=240	::PROGRAM INTERRUPT REQUEST VECTOR
303	160060	ABASE=160060	
304	100360	AVECT1=100360	
305	000200	APRIOR=200	


```

306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321      000000
322
323
324
325      000174
326 000174 000000
327 000176 000000
328
329 000200 000137 001620
330 000204 000137 004654
331 000210 000137 004710

```

```

.SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH      USE
*      -----
*      15          HALT ON ERROR
*      14          LOOP ON TEST
*      13          INHIBIT ERROR TYPEOUTS
*      12          HALT AT END OF DIAGNOSTIC
*      11          INHIBIT ITERATIONS
*      10          BELL ON ERROR
*      9           LOOP ON ERROR
*      8           LOOP ON TEST IN SWR<7:0>
.SBTTL TRAP CATCHER
      .=0
      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      .=174
DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
      JMP      @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
      JMP      BEGIN1  ;;TEST PBR TURNS LDR LIGHT ON AND OFF
      JMP      BEGIN2  ;;TEST ENCODER WITH PUSH BUTTON

```

333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365

000214
000046
000052
001000

001000
000024
000044
001000

001000
001000
001002
001004
001006
001010
001012

```
.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
    $SVPC=.          ;SAVE PC
    .-46             ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
    $ENDAD           ;;2)SET LOC.52 TO ZERO
    .-52             ;; RESTORE PC
    .WORD 0
    .=$SVPC
.=1000
.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
    .SX=.           ;;SAVE CURRENT LOCATION
    .-24            ;;SET POWER FAIL TO POINT TO START OF PROGRAM
    200             ;;FOR APT START UP
    .-44            ;;POINT TO APT INDIRECT ADDRESS PNTR.
    $APTHDR         ;;POINT TO APT HEADER BLOCK
    .=$X            ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.
$APTHD:
$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBAADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 60 ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 120 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 180 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
$ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```

366
367
368
369
370
371
372      001100
373      001100
374      001100 000000
375      001102      000
376      001103      000
377      001104 000000
378      001106 000000
379      001110 000000
380      001112 000000
381      001114      000
382      001115      001
383      001116 000000
384      001120 000000
385      001122 000000
386      001124 000000
387      001126 000000
388      001130 000000
389      001132 000000
390      001134      000
391      001135      000
392      001136 000000
393      001140 177570
394      001142 177570
395      001144 177560
396      001146 177562
397      001150 177564
398      001152 177566
399      001154      000
400      001155      002
401      001156      012
402      001157      000
403      001160 000000
404
405      001162 000000
406      001164 000000
407      001166 000000
408      001170 000000
409      001172 177607 000377
410      001176      077
411      001177      015
412      001200 000012
413
414
415
416
417
418      001202
419      001202 000000
420      001204 000000
421      001206 000000

```

.SBTTL COMMON TAGS

```

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

```

```

      . =1100
SCMTAG:      . =1100      ;; START OF COMMON TAGS
      .WORD      0
$TSTNM: .BYTE      0      ;; CONTAINS THE TEST NUMBER
$ERFLG: .BYTE      0      ;; CONTAINS ERROR FLAG
$ICNT:  .WORD      0      ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD      0      ;; CONTAINS SCOPE LOOP ADDRESS
$LPER?: .WORD      0      ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD      0      ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE      0      ;; CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE      1      ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD      0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD      0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD      0      ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDADR: .WORD      0      ;; CONTAINS 'GOOD' DATA
$BDADR: .WORD      0      ;; CONTAINS 'BAD' DATA
      .WORD      0      ;; RESERVED--NOT TO BE USED
$AUTOB: .BYTE      0      ;; AUTOMATIC MODE INDICATOR
$INTAG: .BYTE      0      ;; INTERRUPT MODE INDICATOR
      .WORD      0
$SWR:     .WORD      DSWR      ;; ADDRESS OF SWITCH REGISTER
$DISP:    .WORD      DDISP     ;; ADDRESS OF DISPLAY REGISTER
$TKS:     177560      ;; TTY KBD STATUS
$TKB:     177562      ;; TTY KBD BUFFER
$TPS:     177564      ;; TTY PRINTER STATUS REG. ADDRESS
$TPB:     177566      ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL:    .BYTE      0      ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS:   .BYTE      2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC:   .BYTE     12      ;; INSERT FILL CHARS. AFTER A "LINE FEED"
$TPFLG:   .BYTE      0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$REGAD:   .WORD      0      ;; CONTAINS THE ADDRESS FROM WHICH ($REGO) WAS OBTAINED
$REGO:    .WORD      0      ;; CONTAINS (($REGAD)+0)
$REGI:    .WORD      0      ;; CONTAINS (($REGAD)+2)
$TIMES:   0          ;; MAX. NUMBER OF ITERATIONS
$ESCAPE:  0          ;; ESCAPE ON ERROR ADDRESS
$BELL:    .ASCIZ    <207><377><377> ;; CODE FOR BELL
$QUES:    .ASCII    /?/      ;; QUESTION MARK
$CRLF:    .ASCII    <15>     ;; CARRIAGE RETURN
$LF:      .ASCIZ    <12>     ;; LINE FEED
*****
.SBTTL APT MAILBOX-ETABLE
*****
.EVEN
$MAIL:    ;; APT MAILBOX
$MSGTY:   .WORD     AMSGTY    ;; MESSAGE TYPE CODE
$FATAL:   .WORD     AFATAL    ;; FATAL ERROR NUMBER
$TESTN:   .WORD     ATESTN    ;; TEST NUMBER

```

422	001210	000000	\$PASS:	.WORD	APASS	::PASS COUNT
423	001212	000000	\$DEVCT:	.WORD	ADEVCT	:::DEVICE COUNT
424	001214	000000	\$UNIT:	.WORD	AUNIT	:::I/O UNIT NUMBER
425	001216	000000	\$MSGAD:	.WORD	AMSGAD	:::MESSAGE ADDRESS
426	001220	000000	\$MSGLG:	.WORD	AMSGLG	:::MESSAGE LENGTH
427	001222		\$ETABLE:			:::APT ENVIRONMENT TABLE
428	001222	000	\$ENV:	.BYTE	AENV	:::ENVIRONMENT BYTE
429	001223	000	\$ENVM:	.BYTE	AENVM	:::ENVIRONMENT MODE BITS
430	001224	000000	\$SWREG:	.WORD	ASWREG	:::APT SWITCH REGISTER
431	001226	000000	\$USWR:	.WORD	AUSWR	:::USER SWITCHES
432	001230	000000	\$CPUOP:	.WORD	ACPUOP	:::CPU TYPE, OPTIONS
433			*			BITS 15-11=CPU TYPE
434			*			11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
435			*			11/70=06, PDQ=07, Q=10
436			*			BIT 10=REAL TIME CLOCK
437			*			BIT 9=FLOATING POINT PROCESSOR
438			*			BIT 8=MEMORY MANAGEMENT
439	001232	000	\$MAMS1:	.BYTE	AMAMS1	:::HIGH ADDRESS, M.S. BYTE
440	001233	000	\$MTYP1:	.BYTE	AMTYP1	:::MEM. TYPE, BLK#1
441			*			MEM. TYPE BYTE -- (HIGH BYTE)
442			*			900 NSEC CORE=001
443			*			300 NSEC BIPOLAR=002
444			*			500 NSEC MOS=003
445	001234	000000	\$MADR1:	.WORD	AMADR1	:::HIGH ADDRESS, BLK#1
446			*			MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
447	001236	000	\$MAMS2:	.BYTE	AMAMS2	:::HIGH ADDRESS, M.S. BYTE
448	001237	000	\$MTYP2:	.BYTE	AMTYP2	:::MEM. TYPE, BLK#2
449	001240	000000	\$MADR2:	.WORD	AMADR2	:::MEM. LAST ADDRESS, BLK#2
450	001242	000	\$MAMS3:	.BYTE	AMAMS3	:::HIGH ADDRESS, M.S. BYTE
451	001243	000	\$MTYP3:	.BYTE	AMTYP3	:::MEM. TYPE, BLK#3
452	001244	000000	\$MADR3:	.WORD	AMADR3	:::MEM. LAST ADDRESS, BLK#3
453	001246	000	\$MAMS4:	.BYTE	AMAMS4	:::HIGH ADDRESS, M.S. BYTE
454	001247	000	\$MTYP4:	.BYTE	AMTYP4	:::MEM. TYPE, BLK#4
455	001250	000000	\$MADR4:	.WORD	AMADR4	:::MEM. LAST ADDRESS, BLK#4
456	001252	100360	\$VECT1:	.WORD	AVECT1	:::INTERRUPT VECTOR#1, BUS PRIORITY#1
457	001254	000000	\$VECT2:	.WORD	AVECT2	:::INTERRUPT VECTOR#2, BUS PRIORITY#2
458	001256	160060	\$BASE:	.WORD	ABASE	:::BASE ADDRESS OF EQUIPMENT UNDER TEST
459	001260	000000	\$DEVN:	.WORD	ADEVN	:::DEVICE MAP
460	001262	000000	\$CDW1:	.WORD	ACDW1	:::CONTROLLER DESCRIPTION WORD#1
461	001264		\$ETEND:			
462			.MEXIT			

```

463          .SBTTL  ERROR POINTER TABLE
464
465          ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
466          ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
467          ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
468          ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
469          ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
470
471          ;*          EM          ;;POINTS TO THE ERROR MESSAGE
472          ;*          DH          ;;POINTS TO THE DATA HEADER
473          ;*          DT          ;;POINTS TO THE DATA
474          ;*          DF          ;;POINTS TO THE DATA FORMAT
475
476          001264          $ERRTB:
477
478          ;ITEM 1
480          EM1          ;FAILED TO LOAD AND READ LDR
481          001264 005036          ;ERRPC LDR EXPECTED ACTUAL
482          001266 006565          ;$ERRPC, LDR, $GDDAT, $BDDAT, 0
483          001270 006772          ;
484          001272 007040          ;
485          ;ITEM 2
486          EM2          ;FAILED TO SET OR CLEAR PBR
487          001274 005104          ;ERRPC PBR EXPECTED ACTUAL
488          001276 006625          ;$ERRPC, PBR, $GDDAT, $BDDAT, 0
489          001300 007004          ;
490          001302 007040          ;
491          ;ITEM 3
492          EM3          ;FAILED TO CLEAR PB FLAG
493          001304 005137          ;ERRPC PBR EXPECTED ACTUAL
494          001306 006665          ;$ERRPC, PBR, $GDDAT, $BDDAT, 0
495          001310 007016          ;
496          001312 007040          ;
497          ;ITEM 4
498          EM4          ;FAILED TO LOAD AND READ PB FLAG
499          001314 005167          ;ERRPC PBR EXPECTED ACTUAL
500          001316 006665          ;$ERRPC, PBR, $GDDAT, $BDDAT, 0
501          001320 007016          ;
502          001322 007040          ;
503          ;ITEM 5
504          EM5          ;FAILED TO CLEAR INTERRUPT ENABLE
505          001324 005227          ;ERRPC PBR EXPECTED ACTUAL
506          001326 006665          ;$ERRPC, PBR, $GDDAT, $BDDAT
507          001330 007016          ;
508          001332 007040          ;
509          ;ITEM 6
510          EM6          ;FAILED TO SET INTERRUPT ENABLE
511          001334 005270          ;ERRPC PBR EXPECTED ACTUAL
512          001336 006665          ;$ERRPC, PBR, $GDDAT, $BDDAT
513          001340 007016          ;
514          001342 007040          ;
515          ;ITEM 7
516          EM7          ;ERRONEOUSLY CLEARED PB FLAG
517
518          001344 005327

```

519	001346	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
520	001350	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
521	001352	007040	DF1	
522				
523			;ITEM 10	
524	001354	005363	EM10	;FAILED TO GENERATE INTERRUPT WITH PB FLAG
525	001356	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
526	001360	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
527	001362	007040	DF1	
528				
529			;ITEM 11	
530	001364	005435	EM11	;CLR PPSR FAILED TO CLEAR PB FLAG
531	001366	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
532	001370	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
533	001372	007040	DF1	
534				
535			;ITEM 12	
536	001374	005476	EM12	;INTERRUPT OK FAILED TO CLEAR INTR FLOP
537	001376	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
538	001400	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
539	001402	007040	DF1	
540				
541			;ITEM 13	
542	001404	005545	EM13	;CLR PPSR FAILED TO CLEAR INTERRUPT ENABLE
543	001406	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
544	001410	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
545	001412	007040	DF1	
546				
547			;ITEM 14	
548	001414	005617	EM14	;UNEXPECTED INTERRUPT
549	001416	006726	DH4	;ERRPC PBSR LOCATION OF FAILURE
550	001420	007030	DT4	;SERRPC,PBSR,\$GDDAT
551	001422	007040	DF1	
552				
553			;ITEM 15	
554	001424	005644	EM15	;BIC FAILED TO CLEAR LDR
555	001426	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
556	001430	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
557	001432	007040	DF1	
558				
559			;ITEM 16	
560	001434	005675	EM16	;BIS FAILED TO SET LDR
561	001436	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
562	001440	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT
563	001442	007040	DF1	
564				
565			;ITEM 17	
566	001444	005723	EM17	;BISB FAILED TO SET LOWER BYTE
567	001446	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
568	001450	007016	DT3	;SLRPPC,PBSR,\$GDDAT,\$BDDAT
569	001452	007040	DF1	
570				
571			;ITEM 20	
572	001454	006016	EM20	;BISB FAILED TO SET UPPER BYTE
573	001456	006665	DH3	;ERRPC PBSR EXPECTED ACTUAL
574	001460	007016	DT3	;SERRPC,PBSR,\$GDDAT,\$BDDAT

575	001462	007040	DF1	
576				
577			:ITEM	21
578	001464	006111		EM21
579	001466	006665		DH3
580	001470	007016		DT3
581	001472	007040		DF1
582				
583			:ITEM	22
584	001474	006206		EM22
585	001476	006665		DH3
586	001500	007016		DT3
587	001502	007040		DF1
588				
589			:ITEM	23
590	001504	006303		EM23
591	001506	006665		DH3
592	001510	007016		DT3
593	001512	007040		DF1
594				
595			:ITEM	24
596	001514	006341		EM24
597	001516	006565		DH1
598	001520	006772		DT1
599	001522	007040		DF1
600				
601			:ITEM	25
602	001524	006377		EM25
603	001526	006625		DH2
604	001530	006772		DT1
605	001532	007040		DF1
606				
607			:ITEM	26
608	001534	006434		EM26
609	001536	006565		DH1
610	001540	006772		DT1
611	001542	007040		DF1
612				
613			:ITEM	27
614	001544	006471		EM27
615	001546	006625		DH2
616	001550	007004		DT2
617	001552	007040		DF1
618				
619			:ITEM	30
620	001554	006527		EM30
621	001556	006665		DH3
622	001560	007016		DT3
623	001562	007040		DF1

:BICB FAILED TO CLEAR LOWER BYTE
 :ERRPC PBSR EXPECTED ACTUAL
 :SERRPC, PBSR, \$GDDAT, \$BDDAT

:BICB FAILED TO CLEAR UPPER BYTE
 :ERRPC PBSR EXPECTED ACTUAL
 :SERRPC, PBSR, \$GDDAT, \$BDDAT

:LOADING LDR CHANGED PBSR DATA
 :ERRPC PBSR EXPECTED ACTUAL
 :SERRPC, PBSR, \$GDDAT, \$BDDAT

:LOADING PBSR CHANGED LDR DATA
 :ERRPC LDR EXPECTED ACTUAL
 :SERRPC, LDR, \$GDDAT, \$GDDAT

:LOADING LDR CHANGED PBR DATA
 :ERRPC PBR EXPECTED ACTUAL
 :SERRPC, PBSR, \$GDDAT, \$BDDAT

:LOADING PBR CHANGED LDR DATA
 :ERRPC LDR EXPECTED ACTUAL
 :SERRPC, LDR, \$GDDAT, \$BDDAT

:LOADING PBSR CHANGED PBR DATA
 :ERRPC PBR EXPECTED ACTUAL
 :SERRPC, PBR, \$GDDAT, \$BDDAT

:LOADING PBR CHANGED PBSR DATA
 :ERRPC PBSR EXPECTED ACTUAL
 :SERRPC, PBSR, \$GDDAT, \$BDDAT

624	001564	160060			PBSR:	ABASE		;PB STATUS REGISTER
625	001566	160062			PBR:	ABASE+2		;PUSH BUTTON REGISTER
626	001570	160064			LDR:	ABASE+4		;LAMP DATA REGISTER
627	001572	100360			VECT:	AVECT1		;INTR. VECTOR
628	001574	100362			VECTR1:	AVECT1+2		;VECT ADDRESS
629	001576	000000			TEMP:	0		
630	001600	000000			TEMP1:	0		
631	001602	000000			TEMP2:	0		
632	001604	000000			TEMPL:	0		
633	001606	000000			BITSAV:	0		
634								
635	001610	011637	001124		UNEXP:	MOV (SP), \$GDDAT		;POP PC
636	001614	104014			ERROR	14		;UNEXPECTED INTR.
637	001616	000002			RTI			;RETURN FROM INTERRUPT
638								
639	001620	000005			BEGIN:	RESET		
640					.SBTTL	INITIALIZE THE COMMON TAGS		
641					;;CLEAR	THE COMMON TAGS (\$CMTAG) AREA		
642	001622	012706	001100		MOV	;\$CMTAG, R6		;FIRST LOCATION TO BE CLEARED
643	001626	005026			CLR	(R6)+		;CLEAR MEMORY LOCATION
644	001630	022706	001140		CMP	;\$SWR, R6	;; DONE?	
645	001634	001374			BNE	-6		;LOOP BACK IF NO
646	001636	012706	001100		MOV	;\$STACK, SP		;SETUP THE STACK POINTER
647					;;INITIALIZE	A FEW VECTORS		
648	001642	012737	010466	000020	MOV	;\$SCOPE, @IOTVEC		;IOT VECTOR FOR SCOPE ROUTINE
649	001650	012737	000340	000022	MOV	@340, @IOTVEC+2		;LEVEL 7
650	001656	012737	010750	000030	MOV	;\$ERROR, @EMTVEC		;EMT VECTOR FOR ERROR ROUTINE
651	001664	012737	000340	000032	MOV	@340, @EMTVEC+2		;LEVEL 7
652	001672	012737	012262	000034	MOV	;\$TRAP, @TRAPVEC		;TRAP VECTOR FOR TRAP CALLS
653	001700	012737	000340	000036	MOV	@340, @TRAPVEC+2		;LEVEL 7
654	001706	012737	012330	000024	MOV	;\$PWROD, @PWVVEC		;POWER FAILURE VECTOR
655	001714	012737	000340	000026	MOV	@340, @PWVVEC+2		;LEVEL 7
656	001722	013737	004564	004556	MOV	\$ENDCT, \$EOPCT		;SETUP END-OF-PROGRAM COUNTER
657	001730	005037	001166		CLR	\$TIMES		;INITIALIZE NUMBER OF ITERATIONS
658	001734	005037	001170		CLR	\$ESCAPE		;CLEAR THE ESCAPE ON ERROR ADDRESS
659	001740	112737	000001	001115	MOV	@1, \$ERMAX		;ALLOW ONE ERROR PER TEST
660	001746	012737	001746	001106	MOV	@., \$LPAOR		;INITIALIZE THE LOOP ADDRESS FOR SCOPE
661	001754	012737	001754	001110	MOV	@., \$LPER		;SETUP THE ERROR LOOP ADDRESS
662					;;SIZE	FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS		
663					;;EQUAL	TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.		
664	001762	013746	000004		MOV	@ERRVEC, -(SP)		;SAVE ERROR VECTOR
665	001766	012737	002022	000004	MOV	@64\$, @ERRVEC		;SET UP ERROR VECTOR
666	001774	012737	177570	001140	MOV	@DSWR, SWR		;SETUP FOR A HARDWARE SWICH REGISTER
667	002002	012737	177570	001142	MOV	@DISP, DISPLAY		;AND A HARDWARE DISPLAY REGISTER
668	002010	022777	177777	177122	CMP	@-1, @SWR		;TRY TO REFERENCE HARDWARE SWR
669	002016	001012			BNE	665		;BRANCH IF NO TIMEOUT TRAP OCCURRED
670								;AND THE HARDWARE SWR IS NOT = -1
671	002020	000403			BR	655		;BRANCH IF NO TIMEOUT
672	002022	012716	002030		64\$:	MOV @65\$, (SP)		;SET UP FOR TRAP RETURN
673	002026	000002			RTI			
674	002030	012737	000176	001140	65\$:	MOV @SWREG, SWR		;POINT TO SOFTWARE SWR
675	002036	012737	000174	001142	MOV	@DISREG, DISPLAY		
676	002044	012637	000004		66\$:	MOV (SP)+, @ERRVEC		;RESTORE ERROR VECTOR
677								
678	002050	005037	001210		CLR	\$PASS		;CLEAR PASS COUNT
679	002054	132737	000200	001223	BITB	@APTSIZE, \$ENVM		;TEST USER SIZE UNDER APT

680	002062	001403			BEQ	675	::YES,USE NON-APT SWITCH
681	002064	012737	001224	001140	MOV	#SSWREG,SWR	::NO,USE APT SWITCH REGISTER
682	002072						
683	002072	005737	000042		675:	TST	#42
684	002076	001002				BNE	INIT
685	002100	104400				TYPE	
686	002102	007237				HEAD1	
687	002104	013737	001256	001564	INIT:	MOV	\$BASE,PBSR
688	002112	013737	001256	001570		MOV	\$BASE,LDR
689	002120	013737	001256	001566		MOV	\$BASE,PBR
690	002126	062737	000002	001566		ADD	#2,PBR
691	002134	062737	000004	001570		ADD	#4,LDR
692	002142	013737	001252	001572		MOV	\$VECT1,VECT
693	002150	013737	001252	001574		MOV	\$VECT1,VECTR1
694	002156	062737	000002	001574		ADD	#2,VECTR1
695	002164	042737	160000	001572		BIC	#160000,VECT
696	002172	042737	160000	001574		BIC	#160000,VECTR1
697	002200	012777	001610	177364		MOV	#UNEXP,#VECT
698	002206	005077	177362			CLR	#VECTR1
699							

E02

MAINDEC-11-DZLKA-A
DZLKA.P11 T1

MACY11 27(732) 25-SEP-76 09:47 PAGE 18
LOAD ALL 1'S AND READ BACK LDR

```

700      ;:*****
701      ;:TEST 1      LOAD ALL 1'S AND READ BACK LDR
702      ;:*****
703      002212 000004      TST1:  SCOPE
704      002214 012737 177777 001124      MOV      #-1,$GDDAT      ;LOAD EXPECTED DATA
705      002222 013777 001124 177340      MOV      $GDDAT,2LDR      ;LOAD ALL ONES INTO LDR
706      002230 017737 177334 001126      MOV      2LDR,$BDDAT      ;READ ACTUAL DATA
707      002236 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE RESULTS
708      002244 001401      BEQ      TST2      ;;BRANCH IF EQUAL
709      002246 104001      ERROR    1      ;FAILED TO LOAD ALL ONES
710
711      ;:*****
712      ;:TEST 2      LOAD AND READ '25' PATTERN INTO LDR
713      ;:*****
714      002250 000004      TST2:  SCOPE
715      002252 012737 125252 001124      MOV      #125252,$GDDAT      ;LOAD EXPECTED DATA
716      002260 013777 001124 177302      MOV      $GDDAT,2LDR      ;LOAD PATTERN INTO LDR
717      002266 017737 177276 001126      MOV      2LDR,$BDDAT      ;READ ACTUAL DATA
718      002274 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE RESULTS
719      002302 001401      BEQ      TST3      ;;BRANCH IF EQUAL
720      002304 104001      ERROR    1      ;FAILED TO LOAD PATTERN
721
722      ;:*****
723      ;:TEST 3      LOAD AND READ '52' PATTERN INTO LDR
724      ;:*****
725      002306 000004      TST3:  SCOPE
726      002310 012737 052525 001124      MOV      #52525,$GDDAT      ;LOAD EXPECTED DATA
727      002316 013777 001124 177244      MOV      $GDDAT,2LDR      ;LOAD PATTERN INTO LDR
728      002324 017737 177240 001126      MOV      2LDR,$BDDAT      ;READ ACTUAL DATA
729      002332 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE RESULTS
730      002340 001401      BEQ      TST4      ;;BRANCH IF EQUAL
731      002342 104001      ERROR    1      ;FAILED TO LOAD PATTERN
732
733      ;:*****
734      ;:TEST 4      FLOAT A ONE THROUGH LDR
735      ;:*****
736      002344 000004      TST4:  SCOPE
737      002346 012737 000001 001124      MOV      #BIT0,$GDDAT      ;LOAD EXPECTED DATA
738      002354 013777 001124 177206      2S:    MOV      $GDDAT,2LDR      ;LOAD LDR BIT
739      002362 017737 177202 001126      MOV      2LDR,$BDDAT      ;READ ACTUAL DATA
740      002370 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE RESULTS
741      002376 001401      BEQ      1S      ;;BRANCH IF EQUAL
742      002400 104001      ERROR    1      ;FAILED TO SET LDR BIT
743      002402 006137 001124      1S:    ROL      $GDDAT      ;TEST NEXT BIT
744      002406 103362      BCC      2S      ;CONTINUE THROUGH LDR
745

```

```

746 ;:*****
747 ;:TEST 5      FLOAT ZERO THROUGH LDR
748 ;:*****
749 002410 000004
750 002412 012737 177776 001124
751 002420 013777 001124 177142
752 002426 017737 177136 001126
753 002434 023737 001124 001126
754 002442 001401
755 002444 104001
756 002446 000261
757 002450 006137 001124
758 002454 103361
759
760 ;:*****
761 ;:TEST 6      LOAD AND READ PBR USING BIT 0
762 ;:*****
763 002456 000004
764 002460 012737 177777 001124
765 002466 012777 030001 177072
766 002474 017737 177066 001126
767 002502 023737 001124 001126
768 002510 001401
769 002512 104002
770
771 ;:*****
772 ;:TEST 7      CLEAR PBR USING BIT 0
773 ;:*****
774 002514 000004
775 002516 005037 001124
776 002522 005077 177040
777 002526 017737 177034 001126
778 002534 023737 001124 001126
779 002542 001401
780 002544 104002
781
782 ;:*****
783 ;:TEST 10     LOAD AND READ BACK PB FLAG
784 ;:*****
785 002546 000004
786 002550 005077 177010
787 002554 012737 100000 001124
788 002562 013777 001124 176774
789 002570 017737 176770 001126
790 002576 023737 001124 001126
791 002604 001401
792 002606 104004
793

```

```

;:*****
;:TEST 5      FLOAT ZERO THROUGH LDR
;:*****
TST5:  SCOPE
      MOV      #177776,$GDDAT      ;LOAD EXPECTED DATA
25:    MOV      $GDDAT,$LDR        ;LOAD LDR BIT
      MOV      $LDR,$BODAT        ;READ ACTUAL DATA
      CMP      $GDDAT,$BODAT      ;COMPARE RESULTS
      BEQ      IS                  ;;BRANCH IF EQUAL
      ERROR   1                    ;FAILED TO CLEAR LDR BIT
15:    SEC
      ROL      $GDDAT             ;SET CARRY BIT
      BCC     25                  ;TEST NEXT BIT
                                       ;CONTINUE THROUGH LDR
;:*****
;:TEST 6      LOAD AND READ PBR USING BIT 0
;:*****
TST6:  SCOPE
      MOV      #-1,$GDDAT         ;LOAD EXPECTED DATA
      MOV      $BIT0,$PBR        ;SET BIT 0
      MOV      $PBR,$BODAT       ;READ ACTUAL DATA
      CMP      $GDDAT,$BODAT     ;COMPARE RESULTS
      BEQ      TST7              ;;BRANCH IF EQUAL
      ERROR   2                    ;FAILED TO SET PBR
;:*****
;:TEST 7      CLEAR PBR USING BIT 0
;:*****
TST7:  SCOPE
      CLR      $GDDAT            ;EXPECTED DATA
      CLR      $PBR              ;CLEAR BIT 0
      MOV      $PBR,$BODAT       ;READ ACTUAL DATA
      CMP      $GDDAT,$BODAT     ;COMPARE RESULTS
      BEQ      TST10            ;;BRANCH IF EQUAL
      ERROR   2                    ;FAILED TO CLEAR PBR
;:*****
;:TEST 10     LOAD AND READ BACK PB FLAG
;:*****
TST10: SCOPE
      CLR      $PBSR             ;CLEAR STATUS REGISTER
      MOV      $BIT15,$GDDAT     ;LOAD EXPECTED DATA
      MOV      $GDDAT,$PBSR      ;LOAD STATUS REG.
      MOV      $PBSR,$BODAT      ;READ ACTUAL DATA
      CMP      $GDDAT,$BODAT     ;COMPARE RESULTS
      BEQ      TST11            ;;BRANCH IF EQUAL
      ERROR   4                    ;FAILED TO SET BIT 15

```

```

794
795
796
797 002610 000004
798 002612 005077 176746
799 002616 005037 001124
800 002622 052777 100000 176734
801 002630 005077 176730
802 002634 017737 176724 001126
803 002642 023737 001124 001126
804 002650 001401
805 002652 104011
806
807
808
809
810 002654 000004
811 002656 012737 000100 001166
812 002664 005077 176674
813 002670 005037 001124
814 002674 012777 100000 176662
815 002702 000005
816 002704 017737 176654 001126
817 002712 023737 001124 001126
818 002720 001401
819 002722 104003
820
821
822
823
824 002724 000004
825 002726 005077 176632
826 002732 005037 001124
827 002736 012777 100000 176620
828 002744 017737 176616 001576
829 002752 017737 176606 001126
830 002760 023737 001124 001126
831 002766 001401
832 002770 104003
833
834
835
836
837 002772 000004
838 002774 005677 176564
839 003000 005037 001124
840 003004 012777 100000 176552
841 003012 017737 176546 001576
842 003020 017737 176540 001126
843 003026 023737 001124 001126
844 003034 001401
845 003036 104003

```

```

*****
; *TEST 11 TEST PB FLAG CLEARED BY CLR PBSR
*****
TST11: SCOPE
CLR 2PBSR ; CLEAR STATUS REGISTER
CLR $GDDAT ; EXPECTED DATA
BIS #BIT15,2PBSR ; SET PB FLAG
CLR 2PBSR ; CLEAR STATUS REGISTER
MOV 2PBSR,$BDDAT ; READ ACTUAL DATA
CMP $GDDAT,$BDDAT ; COMPARE RESULTS
BEQ TST12 ;; BRANCH IF EQUAL
ERROR 11 ; FAILED TO CLEAR PB FLAG

```

```

*****
; *TEST 12 TEST INIT CLEARS PB FLAG
*****
TST12: SCOPE
MOV #100,$TIMES ;; DO 100 ITERATIONS
CLR 2PBSR ; CLEAR STATUS REG.
CLR $GDDAT ; LOAD EXPECTED DATA
MOV #BIT15,2PBSR ; SET PB FLAG
RESET ; INITIALIZE
MOV 2PBSR,$BDDAT ; READ ACTUAL DATA
CMP $GDDAT,$BDDAT ; COMPARE RESULTS
BEQ TST13 ;; BRANCH IF EQUAL
ERROR 3 ; FAILED TO CLEAR PB FLAG

```

```

*****
; *TEST 13 TEST PB FLAG CLEARED WHEN READ PBR H
*****
TST13: SCOPE
CLR 2PBSR ; CLEAR STATUS REGISTER
CLR $GDDAT ; LOAD EXPECTED DATA
MOV #BIT15,2PBSR ; SET PB FLAG
MOV 2PBR,TEMP ; READ PB REGISTER
MOV 2PBSR,$BDDAT ; READ ACTUAL DATA
CMP $GDDAT,$BDDAT ; COMPARE RESULTS
BEQ TST14 ;; BRANCH IF EQUAL
ERROR 3 ; FAILED TO CLEAR PB FLAG

```

```

*****
; *TEST 14 TEST PB FLAG IS CLEARED BY READ STATUS WHEN INTR. ENABLE IS CLEARED
*****
TST14: SCOPE
CLR 2PBSR ; CLEAR STATUS REG
CLR $GDDAT ; EXPECTED DATA
MOV #BIT15,2PBSR ; SET PB FLAG
MOV 2PBSR,TEMP ; READ SR
MOV 2PBSR,$BDDAT ; READ ACTUAL DATA
CMP $GDDAT,$BDDAT ; COMPARE RESULTS
BEQ TST15 ;; BRANCH IF EQUAL
ERROR 3 ; READ SR FAILED TO CLEAR PB FLAG

```

H02

MAINDEC-11-DZLKA-A
DZLKA.P11 T15

MACY11 27(732) 25-SEP-76 09:47 PAGE 21
LOAD AND READ INTERRUPT ENABLE BIT, ALSO UNEXPECTED INTRPT TEST

```
846                                     ;*****  
847                                     ;*TEST 15      LOAD AND READ INTERRUPT ENABLE BIT, ALSO UNEXPECTED INTRPT TEST  
848                                     ;*****  
849 003040 000004                       †ST15: SCOPE  
850 003042 005077 176516                CLR      @PBSR                ; CLEAR PB STATUS REGISTER  
851 003046 012737 040000 001124        MOV      @BIT14, $GDDAT      ; LOAD EXPECTED DATA  
852 003054 013777 001124 176502        MOV      $GDDAT, @PBSR      ; LOAD STATUS REGISTER  
853 003062 017737 176476 001126        MOV      @PBSR, @BDDAT      ; READ ACTUAL DATA  
854 003070 023737 001124 001126        CMP      $GDDAT, @BDDAT     ; COMPARE RESULTS  
855 003076 001401                       BEQ      TST16                ; ; BRANCH IF EQUAL  
856 003100 104006                       ERROR    6                    ; FAILED TO LOAD INTERRUPT ENABLE  
857                                     ;*****  
858                                     ;*TEST 16      TEST INTERRUPT ENABLE CLEARED BY INIT  
859                                     ;*****  
860 003102 000004                       †ST16: SCOPE  
861 003104 012737 000100 001166        MOV      #100, $TIMES       ; ; DO 100 ITERATIONS  
862 003112 005037 001124                CLR      $GDDAT             ; LOAD EXPECTED DATA  
863 003116 005077 176442                CLR      @PBSR              ; ENSURE CLEAR REG  
864 003122 012777 040000 176434        MOV      @BIT14, @PBSR      ; SET BIT 14  
865 003130 000005                       RESET                               ; INITIALIZE  
866 003132 017737 176426 001126        MOV      @PBSR, @BDDAT      ; READ ACTUAL DATA  
867 003140 023737 001124 001126        CMP      $GDDAT, @BDDAT     ; COMPARE RESULTS  
868 003146 001401                       BEQ      TST17                ; ; BRANCH IF EQUAL  
869 003150 104005                       ERROR    5                    ; FAILED TO CLEAR INTRPT ENABLE  
870                                     ;*****  
871                                     ;*TEST 17      TEST INTERRUPT ENABLE CLEARED BY CLR PBSR  
872                                     ;*****  
873 003152 000004                       †ST17: SCOPE  
874 003154 005077 176404                CLR      @PBSR              ; CLEAR STATUS REGISTER  
875 003160 005037 001124                CLR      $GDDAT             ; EXPECTED DATA  
876 003164 052777 040000 176372        BIS      @BIT14, @PBSR      ; SET INTERRUPT ENABLE  
877 003172 005077 176366                CLR      @PBSR              ; CLEAR STATUS REGISTER  
878 003176 017737 176362 001126        MOV      @PBSR, @BDDAT      ; READ ACTUAL DATA  
879 003204 023737 001124 001126        CMP      $GDDAT, @BDDAT     ; COMPARE RESULTS  
880 003212 001401                       BEQ      TST20                ; ; BRANCH IF EQUAL  
881 003214 104013                       ERROR    13                   ; FAILED TO CLEAR INTERRUPT ENABLE  
882                                     ;*****  
883                                     ;*TEST 20      GENERATE INTERRUPT WITH PB FLAG AND INTRPT EN  
884                                     ;*****  
885                                     ;*****  
886 003216 000004                       †ST20: SCOPE  
887 003220 005077 176340                CLR      @PBSR              ; CLEAR STATUS REGISTER  
888 003224 012777 003274 176340        MOV      @2$, @VECT         ; INTERRUPT ADDRESS  
889 003232 012737 140000 001124        MOV      @BIT15!@BIT14, $GDDAT ; EXPECTED DATA  
890 003240 005037 001126                CLR      @BDDAT            ;  
891 003244 012777 040000 176312        MOV      @BIT14, @PBSR      ; SET INTERRUPT ENABLE  
892 003252 012777 140000 176304        MOV      @BIT15!@BIT14, @PBSR ; SET PB FLAG  
893 003260 005037 177776                CLR      PSH                ; LOWER PSH  
894 003264 000240                       1$: NOP  
895 003266 000240                       NOP  
896 003270 104010                       ERROR    10                   ; PB FLAG FAILED TO GENERATE INTERRUPT  
897 003272 000401                       BR      3$                   ; ; BRANCH AND CLEAR I.E.  
898 003274 022626                       2$: CMP      (SP)+, (SP)+     ; POP STACK  
899 003276 005077 176262                       3$: CLR      @PBSR           ; CLEAR I.E.
```

```

900
901
902
903
904 003302 000004
905 003304 005077 176254
906 003310 113737 001253 001576
907 003316 042737 177437 001576
908 003324 013737 001576 001600
909 003332 162737 000040 001600
910 003340 012777 003404 176224
911 003346 012777 000340 176220
912 003354 013737 001576 177776
913 003362 012777 040000 176174
914 003370 012777 140000 176166
915 003376 000240
916 003400 000240
917 003402 000403
918 003404 022626 15:
919 003406 104010 ERROR 10
920 003410 000425 BR TST22 ;; BRANCH
921 003412 012777 003450 176152 25: MOV #35, @VECT ; INTERRUPT ADDRESS
922 003420 012777 000340 176146 MOV #PR7, @VECTR1 ; SET PRIORITY
923 003426 013737 001600 177776 MOV TEMP1, PSW ; LOAD PSW
924 003434 000240 NOP
925 003436 000240 NOP
926 003440 005037 001126 CLR $BDDAT
927 003444 104010 ERROR 10 ; PB FLAG FAILED TO GENERATE INTERRUPT
928 003446 000406 BR TST22 ;; BRANCH TO NEXT TEST
929 003450 022626 35: CMP (SP)+, (SP)+ ; POP STACK
930 003452 012777 001610 176112 MOV #UNEXP, @VECT ; UNEXPECTED INTERRUPT ADDRESS
931 003460 005077 176100 CLR @PBSR ; CLEAR OUT PBSR
932
933
934
935
936 003464 000004
937 003466 012737 000340 177776
938 003474 005077 176066
939 003500 005077 176060
940 003504 012737 140000 001124
941 003512 013777 001124 176044
942 003520 017737 176040 001124
943 003526 017737 176032 001124
944 003534 023737 001124 001126
945 003542 001401
946 003544 104007

```

```

*****
; *TEST 21 TEST FOR CORRECTNESS OF PRIORITY OF INTERRUPT
*****
TST21: SCOPE
CLR @PBSR ; CLEAR STATUS REGISTER
MOVB $VECT1+1, TEMP ; GET PRIORITY LEVEL
BIC #177437, TEMP ; MASK TO VECTOR BITS
MOV TEMP, TEMP1 ; SAVE IT
SUB #40, TEMP1 ; LOWER PRIORITY
MOV #15, @VECT ; INTERRUPT ADDRESS
MOV #PR7, @VECTR1 ; LOAD VECTOR ADDRESS PSW
MOV TEMP, PSW ; LOWER PSW
MOV #BIT14, @PBSR ; SET INTERRUPT ENABLE
MOV #BIT15!BIT14, @PBSR ; SET PB FLAG
NOP
NOP
BR 25 ; INTERRUPT NOT SERVICED
15: CMP (SP)+, (SP)+ ; POP STACK
ERROR 10 ; INTERRUPT LEVEL NOT CORRECT
BR TST22 ;; BRANCH TO NEXT TEST
25: MOV #35, @VECT ; INTERRUPT ADDRESS
MOV #PR7, @VECTR1 ; SET PRIORITY
MOV TEMP1, PSW ; LOAD PSW
NOP
NOP
CLR $BDDAT
ERROR 10 ; PB FLAG FAILED TO GENERATE INTERRUPT
BR TST22 ;; BRANCH TO NEXT TEST
35: CMP (SP)+, (SP)+ ; POP STACK
MOV #UNEXP, @VECT ; UNEXPECTED INTERRUPT ADDRESS
CLR @PBSR ; CLEAR OUT PBSR
*****
; *TEST 22 TEST PB FLAG IS NOT CLEARED BY READ STATUS WHEN INTR ENABLE IS SET.
*****
TST22: SCOPE
MOV #340, PSW
CLR @PBR ; CLEAR BUTTON REGISTER
CLR @PBSR ; CLEAR STATUS REGISTER
MOV #BIT15!BIT14, $GDDAT ; LOAD EXPECTED DATA
MOV $GDDAT, @PBSR ; LOAD STATUS
MOV @PBSR, $BDDAT ; READ STATUS REG
MOV @PBSR, $BDDAT ; READ STATUS REG AGAIN
CMP $GDDAT, $BDDAT ; COMPARE REGISTERS
BEQ TST23 ;; BR IF PB FLAG CLEARED IN ERROR
ERROR 7 ; ERRONEOUSLY CLEARED PB FLAG

```

```

947
948
949
950 003546 000004
951 003550 005077 176014 001606
952 003554 012737 000001 001606
953 003562 012777 177777 176000
954 003570 012737 177777 001124
955 003576 043737 001606 001124
956 003604 043777 001606 175756
957 003612 017737 175752 001126
958 003620 023737 001124 001126
959 003626 001401
960 003630 104015
961 003632 006137 001606
962 003636 103351
963
964
965
966 003640 000004
967 003642 012737 000001 001606
968 003650 005077 175714
969 003654 013737 001606 001124
970 003662 053777 001606 175700
971 003670 017737 175674 001126
972 003676 023737 001124 001126
973 003704 001401
974 003706 104016
975 003710 006137 001606
976 003714 103355
977
978
979
980
981 003716 000004
982 003720 005077 175644
983 003724 012777 052525 175636
984 003732 012737 052777 001124
985 003740 152777 177777 175622
986 003746 017737 175616 001126
987 003754 023737 001124 001126
988
989 003762 001401
990 003764 104017
991 003766 012777 052525 175574
992 003774 013737 001570 001604
993 004002 005237 001604
994 004006 012737 177525 001124
995 004014 152777 177777 175562
996 004022 017737 175542 001126
997 004030 023737 001124 001126
998 004036 001401
999 004040 104020

;*****
;*TEST 23 TEST BIC INSTRUCTION CLEARS LDR
;*****
TST23: SCOPE
CLR @LDR ;CLEAR LAMP DATA REGISTER
MOV #BIT0,BITSAV ;INITIALIZE BITSAV
25: MOV #-1,@LDR ;SET REGISTER TO ALL 1'S
MOV #-1,$GDDAT ;EXPECTED
BIC BITSAV,$GDDAT ;DATA
BIC BITSAV,@LDR ;CLEAR PDSR BIT
MOV @LDR,$BDDAT ;READ ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;BRANCH IF EQUAL
ERROR 15 ;FAILED TO CLEAR LDR BIT
15: ROL BITSAV ;SHIFT TO TEST NEXT BIT
BCC 25 ;BRANCH IF NOT DONE ALL BITS
;*****
;*TEST 24 TEST BIS INSTRUCTION SETS LDR
;*****
TST24: SCOPE
MOV #BIT0,BITSAV ;INITIALIZE BITSAV
25: CLR @LDR ;CLEAR LAMP DATA REGISTER
MOV BITSAV,$GDDAT ;DATA
BIS BITSAV,@LDR ;SET LDR BIT
MOV @LDR,$BDDAT ;READ ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;BRANCH IF EQUAL
ERROR 16 ;FAILED TO SET LDR BIT
15: ROL BITSAV ;SHIFT TO TEST NEXT BIT
BCC 25 ;BRANCH BACK IF NOT DONE ALL BITS
;*****
;*TEST 25 TEST BISB INSTRUCTION SETS LDR
;*****
TST25: SCOPE
CLR @LDR ;CLEAR LAMP DATA REGISTER
MOV #52525,@LDR ;LOAD ALTERNATING 1'S IN LDR
MOV #52777,$GDDAT ;EXPECTED DATA
BISB #-1,@LDR ;SET LOWER BYTE TO ALL 1'S
MOV @LDR,$BDDAT ;READ ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;BRANCH IF EQUAL
ERROR 17 ;FAILED TO SET LOWER BYTE & LEAVE UPPER BYTE THE
15: MOV #52525,@LDR ;LOAD ALTERNATING 1'S IN LDR
MOV LDR,TEMP1 ;GET ADDRESS OF LDR
INC TEMP1 ;GET ADDRESS OF UPPER BYTE OF LDR
MOV #177525,$GDDAT ;EXPECTED DATA
BISB #-1,@TEMP1 ;SET UPPER BYTE TO ALL 1'S
MOV @LDR,$BDDAT ;READ ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ TST26 ;;BRANCH IF EQUAL
ERROR 20 ;FAILED TO SET UPPER BYTE &
; LEAVE LOWER BYTE THE SAME

```

```

1000
1001
1002
1003 004042 000004
1004 004044 005077 175520
1005 004050 012777 052525 175512
1006 004056 012737 052400 001124
1007 004064 142777 177777 175476
1008 004072 017737 175472 001126
1009 004100 023737 001124 001126
1010 004106 001401
1011 004110 104021
1012 004112 012777 052525 175450 1S:
1013 004120 013737 001570 001604
1014 004126 005237 001604
1015 004132 012737 000125 001124
1016 004140 142777 177777 175436
1017 004146 017737 175416 001126
1018 004154 023737 001124 001126
1019 004162 001401
1020 004164 104022
1021
1022
1023
1024
1025
1026 004166 000004
1027 004170 005077 175370
1028 004174 005037 001124
1029 004200 012777 052525 175362
1030 004206 017737 175352 001126
1031 004214 023737 001124 001126
1032 004222 001401
1033 004224 104023
1034 004226 005077 175336 1S:
1035 004232 012777 100000 175324
1036 004240 017737 175324 001126
1037 004246 023737 001124 001126
1038 004254 001401
1039 004256 104024
1040

```

```

*****
*TEST 26 TEST BICB INSTRUCTION CLEARS LDR
*****
TST26: SCOPE
CLR @LDR ;CLEAR LAMP DATA REGISTER
MOV #52525,@LDR ;LOAD ALTERNATING 1'S IN LSR
MOV #52400,$GDDAT ;EXPECTED DATA
BICB #-1,@LDR ;CLEAR LOWER BYTE OF LDR
MOV @LDR,$BDDAT ;READ ACTUAL DTA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;BRANCH IF EQUAL
ERROR 21 ;FAILED TO CLEAR LOWER BYTE & LEAVE UPPER BYTE T
1S: MOV #52525,@LDR ;LOAD ALTERNATING 1'S IN LDR
MOV LDR,TEMPL ;GET ADDRESS OF LDR
INC TEMPL ;GET ADDRESS OF UPPER BYTE OF LDR
MOV #125,$GDDAT ;EXPECTED DATA
BICB #-1,@TEMPL ;CLEAR UPPER BYTE OF LDR
MOV @LDR,$BDDAT ;READ ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ TST27 ;;BRANCH IF EQUAL
ERROR 22 ;FAILED TO CLEAR UPPER BYTE &
; LEAVE LOWER BYTE THE SAME

*****
*TEST 27 TEST THAT PBSR AND LDR DON'T CHANGE
*****
TST27: SCOPE
CLR @PBSR ;CLEAR STATUS REGISTER
CLR $GDDAT ;LOAD EXPECTED DATA
MOV #52525,@LDR ;LOAD PATTERN INTO LDR
MOV @PBSR,$BDDAT ;READ STATUS REGISTER
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;;GO TO NEXT SECTION
ERROR 23
1S: CLR @LDR ;CLEAR LDR
MOV #BIT15,@PBSR ;SET BIT 15 IN PBSR
MOV @LDR,$BDDAT ;READ LDR DATA
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ TST30 ;;GO TO NEXT TEST
ERROR 24

```



```

1041
1042
1043
1044 004260 000004
1045 004262 005077 175300
1046 004266 012777 000001 175272
1047 004274 012737 177777 001124
1048 004302 012777 052525 175260
1049 004310 017737 175252 001126
1050 004316 023737 001124 001126
1051 004324 001401
1052 004326 104025
1053 004330 005077 175234
1054 004334 005077 175226
1055 004340 005037 001124
1056 004344 012777 000001 175214
1057 004352 017737 175212 001126
1058 004360 023737 001124 001126
1059 004366 023737 001124 001126
1060 004374 001401
1061 004376 104026
1062
1063
1064
1065
1066 004400 000004
1067 004402 005077 175160
1068 004406 005037 001124
1069 004412 012777 052525 175144
1070 004420 017737 175142 001126
1071 004426 023737 001124 001126
1072 004434 001401
1073 004436 104027
1074 004440 005077 175120
1075 004444 012777 000001 175114
1076 004452 017737 175106 001126
1077 004460 042737 000036 001126
1078 004466 023737 001124 001126
1079 004474 001401
1080 004476 104030
1081 004500 005077 175062
1082 004504 005077 175060
1083 004510 005077 175050
1084

;*****
;TEST 30 TEST THAT PBR AND LDR DON'T CHANGE
;*****
TST30: SCOPE
CLR @PBR ;CLEAR PBR
MOV #BIT0,@PBR ;SET ALL 1'S IN PBR
MOV #177777,$GDDAT ;LOAD EXPECTED DATA
MOV #52525,@LDR ;LOAD PATTERN INTO LDR
MOV @PBR,$BDDAT ;READ PBR
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;;GO TO NEXT SECTION
ERROR 25
15: CLR @LDR ;CLEAR LDR
CLR @PBR ;CLEAR PBR
CLR $GDDAT ;LOAD EXPECTED DATA
MOV #BIT0,@PBR ;SET PBR TO ALL ONES
MOV @LDR,$BDDAT ;READ LDR
CMP $GDDAT,$BDDAT ;READ LDR
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ TST31 ;;GO TO NEXT TEST
ERROR 26

;*****
;TEST 31 TEST THAT PBR AND PBR DON'T CHANGE
;*****
TST31: SCOPE
CLR @PBR ;CLEAR PBR
CLR $GDDAT ;LOAD EXPECTED DATA
MOV #52525,@PBR ;LOAD PATTERN INTO PBR
MOV @PBR,$BDDAT ;READ PBR
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 15 ;;GO TO NEXT SECTION
ERROR 27
15: CLR @PBR ;CLEAR PBR
MOV #BIT0,@PBR ;SET PBR TO ALL ONES
MOV @PBR,$BDDAT ;READ PBR
BIC #36,$BDDAT ;MASK TO "ENCODER" BITS
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 25 ;;BR IF CORRECT
ERROR 30 ;PBR CHANGED IN ERROR
25: CLR @PBR ;CLEAR BP REGISTER
CLR @LDR ;CLEAR LAMPS
CLR @PBR ;CLEAR STATUS

```

```

1085
1086
1087
1088
1089 004514 000004
1090 004516 032777 010000 174414
1091 004524 001401
1092 004526 000000
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102 004530
1103 004530 000004
1104 004532 005037 001102
1105 004536 005037 001166
1106 004542 005237 001210
1107 004546 042737 100000 001210
1108 004554 005327
1109 004556 000001
1110 004560 003022
1111 004562 012737
1112 004564 000001
1113 004566 004556
1114 004570 104400 004635
1115 004574 013746 001210
1116 004600 104404
1117 004602 104400 004632
1118 004606 013700 000042
1119 004612 001405
1120 004614 000005
1121 004616 004710
1122 004620 000240
1123 004622 000240
1124 004624 000240
1125 004626
1126 004626 000137
1127 004630 002104
1128 004632 377 377 000
1129 004635 015 042412 042116
1130 004642 050040 051501 020123
1131 004650 000043
1132

```

```

*****
;TEST 32 HALT AT END OF DIAGNOSTIC
*****
†ST32: SCOPE
      BIT #BIT12, QSWR ; IS BIT 12 SET?
      BEQ $EOP ; NO-PRINT END-OF-PASS MESSAGE
      HALT ; HALT AT END OF LOGIC TESTS

.SBTTL END OF PASS ROUTINE

*****
; INCREMENT THE PASS NUMBER ($PASS)
; TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
; IF THERES A MONITOR GO TO IT
; IF THERE ISN'T JUMP TO INIT

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

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

$ENDCT: .WORD 1
      $EOPCT
      TYPE $SENDMG ; TYPE "END PASS #"
      MOV $PASS, -(SP) ; SAVE $PASS FOR TYPEOUT
      TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE $ENULL ; TYPE A NULL CHARACTER
      $GET42: MOV Q#42, R0 ; GET MONITOR ADDRESS
      BEQ $DOAGN ; BRANCH IF NO MONITOR
      RESET ; CLEAR THE WORLD
      $ENDAD: JSR PC, (R0) ; GO TO MONITOR
      NOP ; SAVE ROOM
      NOP ; FOR
      NOP ; ACT11

$DOAGN:
      JMP Q(PC)+ ; RETURN

$RTNAD: .WORD INIT
$ENULL: .BYTE -1, -1, 0 ; NULL CHARACTER STRING
$SENDMG: .ASCIZ <15><12>/END PASS #/

```

```

1133      ;:*****
1134      ;:TEST 33      PUSH BUTTON TURNS LDR LIGHT ON AND OFF LOOP
1135      ;:*****
1136      004652 000004
1137      004654 012706 001100
1138      004660 005077 174700
1139      004664 104400
1140      004666 007042
1141      004670 005777 174670
1142      004674 100375
1143      004676 017777 174664 174664
1144      004704 000771
1145
1146      ;:*****
1147      ;:TEST 34      TEST ENCODER WITH PUSH BUTTON
1148      ;:*****
1149      004706 000004
1150      004710 012706 001100
1151      004714 005077 174644
1152      004720 104400
1153      004722 007140
1154      004724 005077 174636
1155      004730 005777 174630
1156      004734 100375
1157      004736 017737 174622 001600
1158      004744 042737 177741 001600
1159      004752 012737 004776 001602
1160      004760 063737 001600 001602
1161      004766 017777 174610 174574
1162      004774 000753
1163
1164      004776 000001
1165      005000 000002
1166      005002 000004
1167      005004 000010
1168      005006 000020
1169      005010 000040
1170      005012 000100
1171      005014 000200
1172      005016 000400
1173      005020 001000
1174      005022 002000
1175      005024 004000
1176      005026 010000
1177      005030 020000
1178      005032 040000
1179      005034 100000

      ;:*****
      ;:TEST 33      PUSH BUTTON TURNS LDR LIGHT ON AND OFF LOOP
      ;:*****
      †ST33: SCOPE
      BEGIN1: MOV      #STACK, SP      ;LOAD STACK POINTER
              CLR      @PBSR          ;CLEAR STATUS REGISTER
              TYPE     ;TYPE OUT EXPLANATION
              MESS1    ;OF TEST
1S:         TST      @PBSR          ;STALL
              BPL      1$            ;TIME
              MOV      @PBR, @LDR     ;SET OR TURN OFF LIGHT
              BR       1$            ;BRANCH BACK

      ;:*****
      ;:TEST 34      TEST ENCODER WITH PUSH BUTTON
      ;:*****
      †ST34: SCOPE
      BEGIN2: MOV      #STACK, SP      ;LOAD STACK POINTER
              CLR      @PBSR          ;CLEAR STATUS REGISTER
              TYPE     ;TYPE OUT EXPLANATION
              MESS3    ;OF TEST
1S:         CLR      @PBR            ;CLEAR PUSH BUTTON REGISTER
2S:         TST      @PBSR          ;STALL
              BPL      2$            ;TIME
              MOV      @PBSR, TEMP1   ;READ STATUS REGISTER
              BIC      #177741, TEMP1 ;GET ENCODER BITS
              MOV      #TABLE, TEMP2  ;GET STARTING ADDRESS
              ADD      TEMP1, TEMP2   ;ADD OFFSET TO ADDRESS
              MOV      @TEMP2, @LDR   ;SET OR TURN OFF LAMP
              BR       1$            ;BRANCH BACK

      TABLE: BIT0
              BIT1
              BIT2
              BIT3
              BIT4
              BIT5
              BIT6
              BIT7
              BIT8
              BIT9
              BIT10
              BIT11
              BIT12
              BIT13
              BIT14
              BIT15

```

1180						
1181						
1182	005036	040506	046111	042105	EM1:	.SBTTL ASCII MESSAGES
1183	005044	052040	020117	047514		.ASCIZ /FAILED TO LOAD AND READ LDR CORRECTLY/
1184	005052	042101	040440	042116		
1185	005060	051040	040505	020104		
1186	005066	042114	020122	047503		
1187	005074	051122	041505	046124		
1188	005102	000131				
1189	005104	040506	046111	042105	EM2:	.ASCIZ /FAILED TO SET OR CLEAR PBR/
1190	005112	052040	020117	042523		
1191	005120	020124	051117	041440		
1192	005126	042514	051101	050040		
1193	005134	051102	000			
1194	005137	106	044501	042514	EM3:	.ASCIZ /FAILED TO CLEAR PB FLAG/
1195	005144	020104	047524	041440		
1196	005152	042514	051101	050040		
1197	005160	020102	046106	043501		
1198	005166	000				
1199	005167	106	044501	042514	EM4:	.ASCIZ /FAILED TO LOAD AND READ PB FLAG/
1200	005174	020104	047524	046040		
1201	005202	040517	020104	047101		
1202	005210	020104	042522	042101		
1203	005216	050040	020102	046106		
1204	005224	043501	000			
1205	005227	106	044501	042514	EM5:	.ASCIZ /FAILED TO CLEAR INTERRUPT ENABLE/
1206	005234	020104	047524	041440		
1207	005242	042514	051101	044440		
1208	005250	052116	051105	052522		
1209	005256	052120	042440	040516		
1210	005264	046102	000105			
1211	005270	040506	046111	042105	EM6:	.ASCIZ /FAILED TO SET INTERRUPT ENABLE/
1212	005276	052040	020117	042523		
1213	005304	020124	047111	042524		
1214	005312	051122	050125	020124		
1215	005320	047105	041101	042514		
1216	005326	000				
1217	005327	105	051122	047117	EM7:	.ASCIZ /ERRONEOUSLY CLEARED PB FLAG/
1218	005334	047505	051525	054514		
1219	005342	041440	042514	051101		
1220	005350	042105	050040	020102		
1221	005356	046106	043501	000		
1222	005363	106	044501	042514	EM10:	.ASCIZ /FAILED TO GENERATE INTERRUPT WITH PB FLAG/
1223	005370	020104	047524	043440		
1224	005376	047105	051105	052101		
1225	005404	020105	047111	042524		
1226	005412	051122	050125	020124		
1227	005420	044527	044124	050040		
1228	005426	020102	046106	043501		
1229	005434	000				
1230	005435	103	051114	050040	EM11:	.ASCIZ /CLR PBR FAILED TO CLEAR PB FLAG/
1231	005442	051502	020122	040506		
1232	005450	046111	042105	052040		
1233	005456	020117	046103	040505		
1234	005464	020122	041120	043040		
1235	005472	040514	000107			

1235	005476	047111	042524	051122	EM12: .ASCIZ /INTERRUPT OK FAILED TO CLEAR INTR FLOP/
1237	005504	050125	020124	045517	
1238	005512	043040	044501	042514	
1239	005520	020104	047524	041440	
1240	005526	042514	051101	044440	
1241	005534	052116	020122	046106	
1242	005542	050117	000		
1243	005545	103	051114	050040	EM13: .ASCIZ /CLR PBR FAILED TO CLEAR INTERRUPT ENABLE/
1244	005552	051502	020122	040506	
1245	005560	046111	042105	052040	
1246	005566	020117	046103	040505	
1247	005574	020122	047111	042524	
1248	005602	051122	050125	020124	
1249	005610	047105	041101	042514	
1250	005616	000			
1251	005617	125	042516	050133	EM14: .ASCIZ /UNEXPECTED INTERRUPT/
1252	005624	041505	042524	020104	
1253	005632	047111	042524	051122	
1254	005640	050125	000124		
1255	005644	044502	020103	040506	EM15: .ASCIZ /BIC FAILED TO CLEAR LDR /
1256	005652	046111	042105	052040	
1257	005660	020117	046103	040505	
1258	005666	020122	042114	020122	
1259	005674	000			
1260	005675	102	051511	043040	EM16: .ASCIZ /BIS FAILED TO SET LDR/
1261	005702	044501	042514	020104	
1262	005710	047524	051440	052105	
1263	005716	046040	051104	000	
1264	005723	102	051511	020102	EM17: .ASCIZ /BISB FAILED TO SET LOWER BYTE OR LEAVE UPPER BYTE THE SAME/
1265	005730	040506	046111	042105	
1266	005736	052040	020117	042523	
1267	005744	020124	047514	042527	
1268	005752	020122	054502	042524	
1269	005760	047440	020122	042514	
1270	005766	053101	020105	050125	
1271	005774	042520	020122	054502	
1272	006002	042524	052040	042510	
1273	006010	051440	046501	000105	EM20: .ASCIZ /BISB FAILED TO SET UPPER BYTE OR LEAVE LOWER BYTE THE SAME/
1274	006016	044502	041123	043040	
1275	006024	044501	042514	020104	
1276	006032	047524	051440	052105	
1277	006040	052440	050120	051105	
1278	006046	041040	052131	020105	
1279	006054	051117	046040	040505	
1280	006062	042526	046040	053517	
1281	006070	051105	041040	052131	
1282	006076	020105	044124	020105	
1283	006104	040523	042515	000	
1284	006111	102	041511	020102	EM21: .ASCIZ /BICB FAILED TO CLEAR LOWER BYTE OR LEAVE UPPER BYTE THE SAME/
1285	006116	040506	046111	042105	
1286	006124	052040	020117	046103	
1287	006132	040505	020122	047514	
1288	006140	042527	020122	054502	
1289	006146	042524	047440	020122	
1290	006154	042514	053101	020105	
1291	006162	050125	042520	020122	

1292	006170	054502	042524	052040	
1293	006176	042510	051440	046501	
1294	006204	000105			
1295	006206	044502	041103	043040	EM22: .ASCIZ /BICB FAILED TO CLEAR UPPER BYTE OR LEAVE LOWER BYTE THE SAME/
1296	006214	044501	042514	020104	
1297	006222	047524	041440	042514	
1298	006230	051101	052440	050120	
1299	006236	051105	041040	052131	
1300	006244	020105	051117	046040	
1301	006252	040505	042526	046040	
1302	006260	053517	051105	041040	
1303	006266	052131	020105	044124	
1304	006274	020105	040523	042515	
1305	006302	000			
1306	006303	114	040517	044504	EM23: .ASCIZ /LOADING LDR CHANGED PBSR DATA/
1307	006310	043516	046040	051104	
1308	006316	041440	040510	043516	
1309	006324	042105	050040	051502	
1310	006332	020122	040504	040524	
1311	006340	000			
1312	006341	114	040517	044504	EM24: .ASCIZ /LOADING PBSR CHANGED LDR DATA/
1313	006346	043516	050040	051502	
1314	006354	020122	044103	047101	
1315	006362	042507	020104	042114	
1316	006370	020122	040504	040524	
1317	006376	000			
1318	006377	114	040517	044504	EM25: .ASCIZ /LOADING LDR CHANGED PBR DATA/
1319	006404	043516	046040	051104	
1320	006412	041440	040510	043516	
1321	006420	042105	050040	051102	
1322	006426	042040	052101	000101	
1323	006434	047514	042101	047111	EM26: .ASCIZ /LOADING PBR CHANGED LDR DATA/
1324	006442	020107	041120	020122	
1325	006450	044103	047101	042507	
1326	006456	020104	042114	020122	
1327	006464	040504	040524	000	
1328	006471	114	040517	044504	EM27: .ASCIZ /LOADING PBSR CHANGED PBR DATA/
1329	006476	043516	050040	051502	
1330	006504	020122	044103	047101	
1331	006512	042507	020104	041120	
1332	006520	020122	040504	040524	
1333	006526	000			
1334	006527	114	040517	044504	EM30: .ASCIZ /LOADING PBR CHANGED PBSR DATA/
1335	006534	043516	050040	051102	
1336	006542	041440	040510	043516	
1337	006550	042105	050040	051502	
1338	006556	020122	040504	040524	
1339	006564	000			
1340	006565	105	051122	041520	DH1: .ASCIZ /ERRPC LDR EXPECTED ACTUAL/
1341	006572	020040	046040	051104	
1342	006600	020040	042440	050130	
1343	006606	041505	042524	020104	
1344	006614	020040	041501	052524	
1345	006622	046101	000		
1346	006625	105	051122	041520	DH2: .ASCIZ /ERRPC PBR EXPECTED ACTUAL/
1347	006632	020040	050040	051102	

E03

MAINDEC-11-DZLKA-A MACY11 27(732) 25-SEP-76 09:47 PAGE 31
 DZLKAA.P11 ASCII MESSAGES

1348	006640	020040	042440	050130	
1349	006646	041505	042524	020104	
1350	006654	020040	041501	052524	
1351	006662	046101	000		
1352	006665	105	051122	041520	DH3: .ASCIZ /ERRPC PBSR EXPECTED ACTUAL/
1353	006672	020040	050040	051502	
1354	006700	020122	020040	054105	
1355	006706	042520	052103	042105	
1356	006714	020040	040440	052103	
1357	006722	040525	000114		
1358	006726	051105	050122	020103	DH4: .ASCIZ /ERRPC PBSR LOCATION OF FAILURE/
1359	006734	050040	051502	020122	
1360	006742	020040	046040	041517	
1361	006750	020101	047511	020116	
1362	006756	043117	043040	044501	
1363	006764	052514	042522	000	
1364		006772			.EVEN
1365					
1366	006772	001116	001570	001124	DT1: SERRPC, LDR, SGDDAT, SBDDAT, 0
1367	007000	001126	000000		
1368	007004	001116	001566	001124	DT2: SERRPC, PBR, SGDDAT, SBDDAT, 0
1369	007012	001126	000000		
1370	007016	001116	001564	001124	DT3: SERRPC, PBSR, SGDDAT, SBDDAT, 0
1371	007024	001126	000000		
1372	007030	001116	001564	001124	DT4: SERRPC, PBSR, SGDDAT, 0
1373	007036	000000			
1374	007040	000000			DF1: 0
1375					
1376	007042	005015	051120	051505	MESS1: .ASCIZ <15><12>/PRESS PUSH BUTTON. CORRESPONDING LIGHT SHOULD COMPLEMENT./<15><
1377	007050	020123	052520	044123	
1378	007056	041040	052125	047524	
1379	007064	027116	041440	051117	
1380	007072	042522	050123	047117	
1381	007100	044504	043516	046040	
1382	007106	043511	052110	051440	
1383	007114	047510	046125	020104	
1384	007122	047503	050115	042514	
1385	007130	042515	052116	006456	
1386	007136	000012			
1387	007140	005015	051120	051505	MESS3: .ASCIZ <15><12>/PRESS PUSH BUTTON. ENCODER SHOULD SET CORRESPONDING LIGHT./<15>
1388	007146	020123	052520	044123	
1389	007154	041040	052125	047524	
1390	007162	027116	042440	041516	
1391	007170	042117	051105	051440	
1392	007176	047510	046125	020104	
1393	007204	042523	020124	047503	
1394	007212	051122	051505	047520	
1395	007220	042116	047111	020107	
1396	007226	044514	044107	027124	
1397	007234	005015	000		
1398	007237	015	046412	044501	HEAD1: .ASCIZ <15><12>/MAINDEC-11-DZLKA-A: PUSH BUTTON OPTION DIAGNOSTIC/<15><12>
1399	007244	042116	041505	030455	
1400	007252	026461	055104	045514	
1401	007260	026501	035101	050040	
1402	007266	051525	020110	052502	
1403	007274	052124	047117	047440	

```

1404 007302 052120 047511 020116
1405 007310 044504 043501 047516
1406 007316 052123 041511 005015
1407 007324 000
1408 007326
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420 007326 022737 000176 001140
1421 007334 001074
1422 007336 105777 171602
1423 007342 100071
1424 007344 117746 171576
1425 007350 042716 177600
1426 007354 022726 000007
1427 007360 001062
1428 007362 123727 001134 000001
1429 007370 001456
1430
1431 007372 104400 010053
1432 007376 104400 010060
1433 007402 013746 000176
1434 007406 104401
1435 007410 104400 010071
1436 007414 005046
1437 007416 005046
1438 007420 105777 171520
1439 007424 100375
1440
1441 007426 117746 171514
1442 007432 042716 177600
1443
1444
1445
1446 007436 021627 000025
1447 007442 001005
1448 007444 104400 010046
1449 007450 062706 000006
1450 007454 000757
1451
1452
1453 007456 021627 000015
1454 007462 001022
1455 007464 005766 000004
1456 007470 001403
1457 007472 016677 000002 171440
1458 007500 062706 000006
1459 007504 104400 001177
    
```

.EVEN

.SBTTL TTY INPUT ROUTINE

::*****
 .ENABL LSB

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

```

$CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
        BNE      15$            ;; BRANCH IF NO
        TSTB     @STKS          ;; CHAR THERE?
        BPL      15$            ;; IF NO, DON'T WAIT AROUND
        MOV      @STKB,-(SP)    ;; SAVE THE CHAR
        BIC      #1C177,(SP)   ;; STRIP-OFF THE ASCII
        CMP      #7,(SP)+      ;; IS IT A CONTROL G?
        BNE      15$            ;; NO, RETURN TO USER
        CMP      $AUTOB,#1     ;; ARE WE RUNNING IN AUTO-MODE?
        BEQ      15$            ;; BRANCH IF YES
    
```

```

$GTSWR: TYPE     ,SCNTLG        ;; ECHO THE CONTROL-G (+G)
        TYPE     ,SMSWR         ;; TYPE CURRENT CONTENTS
        MOV      SWREG,-(SP)   ;; SAVE SWREG FOR TYPEOUT
        TYPOC    ,SMNEW        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE     ,SMNEW        ;; PROMPT FOR NEW SWR
19$:   CLR      -(SP)          ;; CLEAR COUNTER
        CLR      -(SP)          ;; THE NEW SWR
7$:   TSTB     @STKS          ;; CHAR THERE?
        BPL      7$            ;; IF NOT TRY AGAIN
    
```

```

        MOV      @STKB,-(SP)   ;; PICK UP CHAR
        BIC      #1C177,(SP)  ;; MAKE IT 7-BIT ASCII
    
```

```

9$:   CMP      (SP),#25        ;; IS IT A CONTROL-U?
        BNE      10$          ;; BRANCH IF NOT
        TYPE     ,SCNTLU      ;; YES, ECHO CONTROL-U (+U)
20$:  ADD      #6,SP          ;; IGNORE PREVIOUS INPUT
        BR       19$          ;; LET'S TRY IT AGAIN
    
```

```

10$:  CMP      (SP),#15        ;; IS IT A <CR>?
        BNE      16$          ;; BRANCH IF NO
        TST      4(SP)        ;; YES, IS IT THE FIRST CHAR?
        BEQ      11$          ;; BRANCH IF YES
        MOV      2(SP),@SWR   ;; SAVE NEW SWR
11$:  ADD      #6,SP          ;; CLEAR UP STACK
14$:  TYPE     ,SCRLF         ;; ECHO <CR> AND <LF>
    
```


1460	007510	123727	001135	000001	CMPB	\$INTAG,#1	::RE-ENABLE TTY KBD INTERRUPTS?
1461	007516	001003			BNE	15\$::BRANCH IF NOT
1462	007520	012777	000100	171416	MOV	#100,@\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
1463	007526	000002			15\$: RTI		::RETURN
1464	007530	004737	011516		16\$: JSR	PC,\$TYPEC	::ECHO CHAR
1465	007534	021627	000060		CMP	(SP),#60	::CHAR < 0?
1466	007540	002420			BLT	18\$::BRANCH IF YES
1467	007542	021627	000067		CMP	(SP),#67	::CHAR > 7?
1468	007546	003015			BGT	18\$::BRANCH IF YES
1469	007550	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
1470	007554	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
1471	007560	001403			BEQ	17\$::BRANCH IF YES
1472	007562	006316			ASL	(SP)	::NO, SHIFT PRESENT
1473	007564	006316			ASL	(SP)	::CHAR OVER TO MAKE
1474	007566	006316			ASL	(SP)	::ROOM FOR NEW ONE.
1475	007570	005266	000002		17\$: INC	2(SP)	::KEEP COUNT OF CHAR
1476	007574	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
1477	007600	000707			BR	7\$::GET THE NEXT ONE
1478	007602	104400	001176		18\$: TYPE	,\$QUES	::TYPE ?<CR><LF>
1479	007606	000720			BR	20\$::SIMULATE CONTROL-U
1480					.DSABL	LSB	

::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

::CALL:

::*	RDCHR	::INPUT A SINGLE CHARACTER FROM THE TTY
::*	RETURN HERE	::CHARACTER IS ON THE STACK
::*		::WITH PARITY BIT STRIPPED OFF

1491	007610	011646			\$RDCHR: MOV	(SP),-(SP)	::PUSH DOWN THE PC
1492	007612	016666	000004	000002	MOV	4(SP),2(SP)	::SAVE THE PS
1493	007620	105777	171320		1\$: TSTB	@\$TKS	::WAIT FOR
1494	007624	100375			BPL	1\$::A CHARACTER
1495	007626	117766	171314	000004	MOVB	@\$TKB,4(SP)	::READ THE TTY
1496	007634	042766	177600	000004	BIC	#1C<177>,4(SP)	::GET RID OF JUNK IF ANY
1497	007642	026627	000004	000023	CMP	4(SP),#23	::IS IT A CONTROL-S?
1498	007650	001013			BNE	3\$::BRANCH IF NO
1499	007652	105777	171266		2\$: TSTB	@\$TKS	::WAIT FOR A CHARACTER
1500	007656	100375			BPL	2\$::LOOP UNTIL ITS THERE
1501	007660	117746	171262		MOVB	@\$TKB,-(SP)	::GET CHARACTER
1502	007664	042716	177600		BIC	#1C177,(SP)	::MAKE IT 7-BIT ASCII
1503	007670	022627	000021		CMP	(SP)+,#21	::IS IT A CONTROL-Q?
1504	007674	001366			BNE	2\$::IF NOT DISCARD IT
1505	007676	000750			BR	1\$::YES, RESUME
1506	007700	026627	000004	000140	3\$: CMP	4(SP),#140	::IS IT UPPER CASE?
1507	007706	002407			BLT	4\$::BRANCH IF YES
1508	007710	026627	000004	000175	CMP	4(SP),#175	::IS IT A SPECIAL CHAR?
1509	007716	003003			BGT	4\$::BRANCH IF YES
1510	007720	042766	000040	000004	BIC	#40,4(SP)	::MAKE IT UPPER CASE
1511	007726	000002			4\$: RTI		::GO BACK TO USER

::THIS ROUTINE WILL INPUT A STRING FROM THE TTY

::CALL:

::*	RDLIN	::INPUT A STRING FROM THE TTY
-----	-------	-------------------------------

1512
1513
1514
1515

```

1516          ;*      RETURN HERE          ;: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1517          ;*                               ;: TERMINATOR WILL BE A BYTE OF ALL 0'S
1518
1519 007730 010346          $RDLIN: MOV      R3, -(SP)          ;: SAVE R3
1520 007732 012703 010036 1$:      MOV      #STTYIN, R3          ;: GET ADDRESS
1521 007736 022703 010046 2$:      CMP      #STTYIN+8., R3          ;: BUFFER FULL?
1522 007742 101405          BLOS      4$                      ;: BR IF YES
1523 007744 104407          ROCHR          ;: GO READ ONE CHARACTER FROM THE TTY
1524 007746 112613          MOVB     (SP)+, (R3)          ;: GET CHARACTER
1525 007750 122713 000177 10$:     CMPB     #177, (R3)          ;: IS IT A RUBOUT
1526 007754 001003          BNE      3$                      ;: SKIP IF NOT
1527 007756 104400 001176 4$:      TYPE     $QUES          ;: TYPE A '?'
1528 007762 000763          BR       1$                      ;: CLEAR THE BUFFER AND LOOP
1529 007764 111337 010034 3$:      MOVB     (R3), 9$          ;: ECHO THE CHARACTER
1530 007770 104400 010034          TYPE     9$
1531 007774 122723 000015          CMPB     15, (R3)+          ;: CHECK FOR RETURN
1532 010000 001356          BNE      2$                      ;: LOOP IF NOT RETURN
1533 010002 105063 177777          CLRB    -1(R3)          ;: CLEAR RETURN (THE 15)
1534 010006 104400 001200          TYPE     $LF          ;: TYPE A LINE FEED
1535 010012 012603          MOV      (SP)+, R3          ;: RESTORE R3
1536 010014 011646          MOV      (SP), -(SP)          ;: ADJUST THE STACK AND PUT ADDRESS OF THE
1537 010016 016666 000004 000002  MOV      4(SP), 2(SP)          ;: FIRST ASCII CHARACTER ON IT
1538 010024 012766 010036 000004  MOV      #STTYIN, 4(SP)
1539 010032 000002          RTI
1540 010034          000          9$:      .BYTE     0          ;: RETURN
1541 010035          000          ;: STORAGE FOR ASCII CHAR. TO TYPE
1542 010036 000010          $TTYIN: .BLKB     8.          ;: TERMINATOR
1543 010046 052536 005015          $CNTLU: .ASCIZ   /↑U/<15><12>          ;: RESERVE 8 BYTES FOR TTY INPUT
1544 010053          136 006507 000012  $CNTLG: .ASCIZ   /↑G/<15><12>          ;: CONTROL "U"
1545 010060 005015 053523 020122  $MSWR:  .ASCIZ   <15><12>/SWR = /          ;: CONTROL "G"
1546 010066 020075          000
1547 010071          040 047040 053505  $MNEW:  .ASCIZ   / NEW = /
1548 010076 036440 000040
    
```

```

1549 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1550
1551 ;:*****
1552 ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1553 ;:SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1554 ;:NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1555 ;:BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1556 ;:REPLACED WITH SPACES.
1557 ;:CALL:
1558 ;:*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1559 ;:*      TYPDS                    ;;GO TO THE ROUTINE
1560
1561 $TYPDS:
1562      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1563      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1564      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1565      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
1566      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
1567      MOV      #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
1568      MOV      20(SP),R5     ;;GET THE INPUT NUMBER
1569      BPL      1$           ;;BR IF INPUT IS POS.
1570      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
1571      MOV      #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
1572      CLR      R0           ;;ZERO THE CONSTANTS INDEX
1573      MOV      #50BLK,R3     ;;SETUP THE OUTPUT POINTER
1574      MOV      #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
1575      CLR      R2           ;;CLEAR THE BCD NUMBER
1576      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
1577      SUB      R1,R5        ;;FORM THIS BCD DIGIT
1578      BLT      4$           ;;BR IF DONE
1579      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
1580      BR      3$
1581      ADD      R1,R5        ;;ADD BACK THE CONSTANT
1582      TST      R2           ;;CHECK IF BCD DIGIT=0
1583      BNE      5$           ;;FALL THROUGH IF 0
1584      TSTB    (SP)         ;;STILL DOING LEADING 0'S?
1585      BMI      7$           ;;BR IF YES
1586      BMI      7$           ;;MSD?
1587      BCC      6$           ;;BR IF NO
1588      MOV      1(SP),-1(R3)  ;;YES--SET THE SIGN
1589      BIS      #'0,R2       ;;MAKE THE BCD DIGIT ASCII
1590      BIS      #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1591      MOV      R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1592      TST      (R0)+        ;;JUST INCREMENTING
1593      CMP      R0,#10       ;;CHECK THE TABLE INDEX
1594      BLT      2$           ;;GO DO THE NEXT DIGIT
1595      BGT      8$           ;;GO TO EXIT
1596      MOV      R5,R2       ;;GET THE LSD
1597      BR      6$           ;;GO CHANGE TO ASCII
1598      TSTB    (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
1599      BPL      9$           ;;BR IF NO
1600      MOV      -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
1601      CLRB    (R3)         ;;SET THE TERMINATOR
1602      MOV      (SP)+,R5     ;;POP STACK INTO R5
1603      MOV      (SP)+,R3     ;;POP STACK INTO R3
1604      MOV      (SP)+,R2     ;;POP STACK INTO R2
    
```

1605	010264	012601			MOV	(SP)+,R1	::POP STACK INTO R1
1606	010266	012600			MOV	(SP)+,R0	::POP STACK INTO R0
1607	010270	104400	010316		TYPE	\$DBLK	::NOW TYPE THE NUMBER
1608	010274	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
1609	010302	012616			MOV	(SP)+,'SP)	
1610	010304	000002			RTI		::RETURN TO USER
1611	010306	023420			\$DTBL:	10000.	
1612	010310	001750				1000.	
1613	010312	000144				100.	
1614	010314	000012				10.	
1615	010316	000004			\$DBLK:	.BLKW 4	

K03

1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630 010326 011646
1631 010330 016666 000004 000002
1632 010336 010046
1633 010340 010146
1634 010342 010246
1635 010344 104410
1636 010346 012600
1637 010350 010037 010454
1638 010354 005001
1639 010356 005002
1640 010360 112046
1641 010362 001420
1642 010364 122716 000060
1643 010370 003026
1644 010372 122716 000067
1645 010376 002423
1646 010400 006301
1647 010402 006102
1648 010404 006301
1649 010406 006102
1650 010410 006301
1651 010412 006102
1652 010414 042716 177770
1653 010420 062601
1654 010422 000756
1655 010424 005726
1656 010426 010166 000012
1657 010432 010237 010464
1658 010436 012602
1659 010440 012601
1660 010442 012600
1661 010444 000002
1662 010446 005726
1663 010450 105010
1664 010452 104400
1665 010454 000000
1666 010456 104400 001176
1667 010462 000730
1668 010464 000000

```
.SBTTL READ AN OCTAL NUMBER FROM THE TTY
*****
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY.
*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE 'LEGAL
*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
*CALL:
*      RDOCT          ;; READ AN OCTAL NUMBER
*      RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
*                   ;; HIGH ORDER BITS ARE IN $HIOCT

$RDOCT: MOV      (SP), -(SP)      ;; PROVIDE SPACE FOR THE
        MOV      4(SP), 2(SP)    ;; INPUT NUMBER
        MOV      RO, -(SP)       ;; PUSH RO ON STACK
        MOV      R1, -(SP)       ;; PUSH R1 ON STACK
        MOV      R2, -(SP)       ;; PUSH R2 ON STACK
1$:     RDLIN      ;; READ AN ASCII LINE
        MOV      (SP)+, RO        ;; GET ADDRESS OF 1ST CHARACTER
        MOV      RO, SS          ;; AND SAVE IT
        CLR      R1              ;; CLEAR DATA WORD
        CLR      R2
2$:     MOVB      (RO)+, -(SP)    ;; PICKUP THIS CHARACTER
        BEQ      3$              ;; IF ZERO GET OUT
        CMPB     #'0, (SP)       ;; MAKE SURE THIS CHARACTER
        BGT      4$              ;; IS AN OCTAL DIGIT
        CMPB     #'7, (SP)
        BLT      4$
        ASL      R1              ;; *2
        ROL      R2
        ASL      R1              ;; *4
        ROL      R2
        ASL      R1              ;; *8
        ROL      R2
        BIC      #'C7, (SP)      ;; STRIP THE ASCII JUNK
        ADD      (SP)+, R1       ;; ADD IN THIS DIGIT
        BR       2$              ;; LOOP
3$:     TST      (SP)+           ;; CLEAN TERMINATOR FROM STACK
        MOV      R1, 12(SP)      ;; SAVE THE RESULT
        MOV      R2, $HIOCT
        MOV      (SP)+, R2       ;; POP STACK INTO R2
        MOV      (SP)+, R1       ;; POP STACK INTO R1
        MOV      (SP)+, RO       ;; POP STACK INTO RO
        RTI
4$:     TST      (SP)+           ;; CLEAN PARTIAL FROM STACK
        CLRB     (RO)            ;; SET A TERMINATOR
        TYPE     ;; TYPE UP THRU THE BAD CHAR.
5$:     .WORD    0
        TYPE     $QUES          ;; "?" "CR" & "LF"
        BR       1$            ;; TRY AGAIN
$HIOCT: .WORD    0              ;; HIGH ORDER BITS GO HERE
```

```

1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683 010466
1684 010466 104406
1685 010470 104406
1686 010472 032777 040000 170440
1687 010500 001114
1688
1689 010502 000416
1690
1691 010504 013746 000004
1692 010510 012737 010530 000004
1693 010516 005737 177060
1694 010522 012637 000004
1695 010526 000463
1696 010530 022626
1697 010532 012637 000004
1698 010536 000423
1699 010540
1700 010540 032777 000400 170372
1701 010546 001404
1702 010550 127737 170364 001102
1703 010556 001465
1704 010560 105737 001103
1705 010564 001421
1706 010566 123737 001115 001103
1707 010574 101015
1708 010576 032777 001000 170334
1709 010604 001404
1710 010606 013737 001110 001106
1711 010614 000446
1712 010616 105037 001103
1713 010622 005037 001166
1714 010626 000415
1715 010630 032777 004000 170302
1716 010636 001011
1717 010640 005737 001210
1718 010644 001406
1719 010646 005237 001104
1720 010652 023737 001166 001104
1721 010660 002024
1722 010662 012737 000001 001104
1723 010670 013737 010746 001166
1724 010676 105237 001102

```

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW11=1 INHIBIT ITERATIONS
*SW09=1 LOOP ON ERROR
*SW08=1 LOOP ON TEST IN SWR<7:0>
*CALL
* SCOPE ;;SCOPE=IOT

```

\$SCOPE:

```

CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
CKSWR
15: BIT #BIT14,$SWR ;:LOOP ON PRESENT TEST?
BNE $OVER ;:YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 65 ;:IF RUNNING ON THE "XOR" TESTER CHANGE
;:THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #55,@#ERRVEC ;:SET FOR TIMEOUT
TST @#177060 ;:TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $SVLAD ;:GO TO THE NEXT TEST
55: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR 75 ;:LOOP ON THE PRESENT TEST
65:*****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,$SWR ;:LOOP ON SPEC. TEST?
BEQ 25 ;:BR IF NO
CMPB $SWR,$STNM ;:ON THE RIGHT TEST? SWR<7:0>
BEQ $OVER ;:BR IF YES
25: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
BEQ 35 ;:BR IF NO
CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 35 ;:BR IF NO
BIT #BIT09,$SWR ;:LOOP ON ERROR?
BEQ 45 ;:BR IF NO
75: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
45: CLRB $ERFLG ;:ZERO THE ERROR FLAG
CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 15 ;:ESCAPE TO THE NEXT TEST
35: BIT #BIT11,$SWR ;:INHIBIT ITERATIONS?
BNE 15 ;:BR IF YES
TST $PASS ;:IF FIRST PASS OF PROGRAM
BEQ 15 ;:INHIBIT ITERATIONS
INC $ICNT ;:INCREMENT ITERATION COUNT
CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
BGE $OVER ;:BR IF MORE ITERATION REQUIRED
15: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
$SVLAD: INCB $STNM ;:COUNT TEST NUMBERS

```

MAINDEC-11-DZLKA-A MACY11 27(732) 25-SEP-76 09:47 PAGE 39
 DZLKA.P11 SCOPE HANDLER ROUTINE

1725	010702	113737	001102	001206		MOVB	\$TSTNM,\$TESTN	;;	SET TEST NUMBER IN APT MAILBOX
1726	010710	011637	001106			MOV	(SP),\$LPADR	;;	SAVE SCOPE LOOP ADDRESS
1727	010714	011637	001110			MOV	(SP),\$LPERR	;;	SAVE ERROR LOOP ADDRESS
1728	010720	005037	001170			CLR	\$ESCAPE	;;	CLEAR THE ESCAPE FROM ERROR ADDRESS
1729	010724	112737	000001	001115		MOVB	#1,\$ERMAX	;;	ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1730	010732	013777	001102	170202	\$OVER:	MOV	\$TSTNM,\$DISPLAY	;;	DISPLAY TEST NUMBER
1731	010740	013716	001106			MOV	\$LPADR,(SP)	;;	FUDGE RETURN ADDRESS
1732	010744	000002				RTI		;;	FIXES PS
1733	010746	003720			\$MXCNT:	2000.		;;	MAX. NUMBER OF ITERATIONS

```

1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748 010750
1749 010750 104406
1750 010752 105237 001103
1751 010756 001775
1752 010760 013777 001102 170154
1753 010766 032777 002000 170144
1754 010774 001402
1755 010776 104400 001172
1756 011002 005237 001112
1757 011006 011637 001116
1758 011012 162737 000002 001116
1759 011020 117737 170072 001114
1760 011026 032777 020000 170104
1761 011034 001004
1762 011036 004737 011150
1763 011042 104400 001177
1764 011046
1765 011046 122737 000001 001222
1766 011054 001007
1767 011056 113737 001114 011070
1768 011064 004737 011604
1769 011070 000
1770 011071 000
1771 011072 000777
1772 011074 005777 170040
1773 011100 100002
1774 011102 000000
1775 011104 104406
1776 011106 032777 001000 170024
1777 011114 001402
1778 011116 013716 001110
1779 011122 005737 001170
1780 011126 001402
1781 011130 013716 001170
1782 011134
1783 011134 022737 004616 000042
1784 011142 001001
1785 011144 000000
1786 011146
1787 011146 000002

```

```

.SBTTL ERROR HANDLER ROUTINE
;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;AND GO TO $ERRTYP ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW15=1 HALT ON ERROR
;SW13=1 INHIBIT ERROR TYPEOUTS
;SW10=1 BELL ON ERROR
;SW09=1 LOOP ON ERROR
;CALL
;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
7$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @($ERRPC,$ITEMB) ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
TYPE $CRLF

20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 2$ ;;NO SKIP APT ERROR REPORT
MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT

21$: .BYTE 0
.BYTE 0

22$: BR 22$ ;;APT ERROR LOOP
2$: TST @SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3$: BIT #BIT09,$SWR ;;LOOP ON ERROR SWITCH SET?
BEQ 4$ ;;BR IF NO
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
PEQ 5$ ;;BR IF NONE
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE

5$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
BNE 6$ ;;BRANCH IF NO
HALT ;;YES

6$: RTI ;;RETURN

```



```

1788
1789
1790
1791
1792
1793
1794
1795
1796 011150
1797 011150 104400 001177
1798 011154 010046
1799 011156 005000
1800 011160 153700 001114
1801 011164 001004
1802
1803 011166 013746 001116
1804
1805 011172 104401
1806 011174 000426
1807 011176 005300
1808 011200 006300
1809 011202 006300
1810 011204 006300
1811 011206 062700 001264
1812 011212 012037 011222
1813 011216 001404
1814 011220 104400
1815 011222 000000
1816 011224 104400 001177
1817 011230 012037 011240
1818 011234 001404
1819 011236 104400
1820 011240 000000
1821 011242 104400 001177
1822 011246 011000
1823 011250 001004
1824 011252 012600
1825 011254 104400 001177
1826 011260 000207
1827 011262
1828 011262 013046
1829 011264 104401
1830 011266 005710
1831 011270 001770
1832 011272 104400 011300
1833 011276 000771
1834 011300 020040 000
1835 011304
    
```

.SBTTL ERROR MESSAGE TIMEOUT ROUTINE

```

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

```

$ERRTYP:
    TYPE      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    MOV      RD,-(SP)    ;; SAVE RD
    CLR      RD          ;; PICKUP THE ITEM INDEX
    BISB    2*$ITEMB,RD
    BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                        ;; TYPE THE PC OF THE ERROR
    MOV     $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
                        ;; ERROR ADDRESS
                        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                        ;; GET OUT
    BR      6$          ;; ADJUST THE INDEX SO THAT IT WILL
                        ;; WORK FOR THE ERROR TABLE
1$:
    DEC     RD
    ASL    RD
    ASL    RD
    ASL    RD
    ADD    # $ERRTB,RD  ;; FORM TABLE POINTER
    MOV    (RD)+,2$    ;; PICKUP "ERROR MESSAGE" POINTER
    BEQ    3$          ;; SKIP TYPEOUT IF NO POINTER
    TYPE   "ERROR MESSAGE"
                        ;; "ERROR MESSAGE" POINTER GOES HERE
    TYPE   $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    MOV    (RD)+,4$    ;; PICKUP "DATA HEADER" POINTER
    BEQ    5$          ;; SKIP TYPEOUT IF 0
    TYPE   "DATA HEADER"
                        ;; "DATA HEADER" POINTER GOES HERE
    TYPE   $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    MOV    (RD),RC     ;; PICKUP "DATA TABLE" POINTER
    BNE    7$          ;; GO TYPE THE DATA
    MOV    (SP)+,RD    ;; RESTORE RD
    TYPE   $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    RTS    PC          ;; RETURN
2$:
    .WORD   0
    TYPE   $CRLF
3$:
    MOV    (RD)+,4$
    BEQ    5$
    TYPE
4$:
    .WORD   0
    TYPE   $CRLF
5$:
    MOV    (RD),RC
    BNE    7$
6$:
    MOV    (SP)+,RD
    TYPE   $CRLF
    RTS    PC
7$:
    MOV    2(RD)+,-(SP) ;; SAVE 2(RD)+ FOR TYPEOUT
    TYPOC
    TST    (RD)
    BEQ    6$
    TYPE   2$
    BR     7$
8$:
    .ASCIZ  "/ /"
    .EVEN
    
```

1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891

011304 105737 001157
011310 100002
011312 000000
011314 000430
011316 010046
011320 017600 000002
011324 122737 000001 001222
011332 001011
011334 132737 000100 001223
011342 001405
011344 010037 011354
011350 004737 011574
011354 000000
011356 132737 000040 001223
011364 001003
011366 112046
011370 001005
011372 005726
011374 012600
011376 062716 000002
011402 000002
011404 122716 000011
011410 001430
011412 122716 000200
011416 001106
011420 005726
011422 104400
011424 001177
011426 105037 011562
011432 000755
011434 004737 011516
011440 123726 001156
011444 001350
011446 013746 001154
011452 105366 000001
011456 002770
011460 004737 011516

.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 \$TPFLG ;; IS THERE A TERMINAL?
BPL 1\$;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3\$;; LEAVE
1\$: MOV RO,-(SP) ;; SAVE RO
MOV 22(SP),RO ;; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV,\$ENV ;; RUNNING IN APT MODE
BNE 62\$;; NO, GO CHECK FOR APT CONSOLE
BITB #APTPOOL,\$ENVM ;; SPOOL MESSAGE TO APT
BEQ 62\$;; NO, GO CHECK FOR CONSOLE
MOV RO,61\$;; SETUP MESSAGE ADDRESS FOR APT
JSR PC,\$ATY3 ;; SPOOL MESSAGE TO APT
61\$: .WORD 0 ;; MESSAGE ADDRESS
62\$: BITB #APTCSUP,\$ENVM ;; APT CONSOLE SUPPRESSED
BNE 60\$;; YES, SKIP TYPE OUT
2\$: MOVB (RO)+,-(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)+,RO ;; RESTORE RO
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
CLRB \$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.?
BNE 2\$;; IF NO GO GET NEXT CHAR.
MOV \$NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7\$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6\$;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,\$TYPEC ;; GO TYPE A NULL

```

1892 011464 105337 011562          DECB  $CHARCNT      ;; DO NOT COUNT AS A COUNT
1893 011470 000770          BR      7$          ;; LOOP
1894
1895          ;HORIZONTAL TAB PROCESSOR
1896
1897 011472 112716 000040      8$:  MOVB  #' (SP)      ;; REPLACE TAB WITH SPACE
1898 011476 004737 011516      9$:  JSR   PC,$TYPEC      ;; TYPE A SPACE
1899 011502 132737 000007 011562  BITB  #7,$CHARCNT      ;; BRANCH IF NOT AT
1900 011510 001372          BNE   9$            ;; TAB STOP
1901 011512 005726          TST   (SP)+          ;; POP SPACE OFF STACK
1902 011514 000724          BR      2$            ;; GET NEXT CHARACTER
1903 011516 105777 167426      $TYPEC: TSTB  2$TPS      ;; WAIT UNTIL PRINTER IS READY
1904 011522 100375          BPL   $TYPEC
1905 011524 116677 000002 167420  MOVB  2(SP),2$TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
1906 011532 122766 000015 000002  CMPB  #CR,2(SP)        ;; IS CHARACTER A CARRIAGE RETURN?
1907 011540 001003          BNE   1$            ;; BRANCH IF NO
1908 011542 105037 011562          CLRB  $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
1909 011546 000406          BR      $TYPEX      ;; EXIT
1910 011550 122766 000012 000002  1$:  CMPB  #LF,2(SP)    ;; IS CHARACTER A LINE FEED?
1911 011556 001402          BEQ   $TYPEX      ;; BRANCH IF YES
1912 011560 105227          INCB  (PC)+          ;; COUNT THE CHARACTER
1913 011562 000000          $CHARCNT: .WORD 0    ;; CHARACTER COUNT STORAGE
1914 011564 000207          $TYPEX: RTS         PC
1915

```

```

1916
1917
1918
1919
1920 011566 112737 000001 012032 $ATY1: MOV  #1,$FFLG ;;TO REPORT FATAL ERROR
1921 011574 112737 000001 012030 $ATY3: MOV  #1,$MFLG ;;TO TYPE A MESSAGE
1922 011602 000403
1923 011604 112737 000001 012032 $ATY4: MOV  #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
1924 011612 $ATYC:
1925 011612 010046 MOV  RO,-(SP) ;;PUSH RO ON STACK
1926 011614 010146 MOV  R1,-(SP) ;;PUSH R1 ON STACK
1927 011616 105737 012030 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
1928 011622 001450 BEQ  5$ ;;IF NOT: BR
1929 011624 122737 000001 001222 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
1930 011632 001031 BNE  3$ ;;IF NOT: BR
1931 011634 132737 000100 001223 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1932 011642 001425 BEQ  3$ ;;IF NOT: BR
1933 011644 017600 000004 MOV  #4(SP),RO ;;GET MESSAGE ADDR.
1934 011650 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDR.
1935 011656 005737 001202 1$: TST  $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
1936 011662 001375 BNE  1$ ;;IF NOT: WAIT
1937 011664 010037 001216 MOV  RO,$MSGAD ;;PUT ADDR IN MAILBOX
1938 011670 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
1939 011672 001376 BNE  2$
1940 011674 163700 001216 SUB  $MSGAD,RO ;;SUB START OF MESSAGE
1941 011700 006200 ASR  RO ;;GET MESSAGE LNTH IN WORDS
1942 011702 010037 001220 MOV  RO,$MSG LGT ;;PUT LENGTH IN MAILBOX
1943 011706 012737 000004 001202 MOV  #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
1944 011714 000413 BR   5$
1945 011716 017637 000004 011742 3$: MOV  #4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
1946 011724 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDRESS
1947 011732 013746 177776 MOV  177776,-(SP) ;;PUSH 177776 ON STACK
1948 011736 004737 011304 JSR  PC,$TYPE ;;CALL TYPE MACRO
1949 011742 000000 4$: .WORD 0
1950 011744 5$:
1951 011744 105737 012032 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
1952 011750 001416 BEQ  12$ ;;IF NOT: BR
1953 011752 005737 001222 TST  $ENV ;;RUNNING UNDER APT?
1954 011756 001413 BEQ  12$ ;;IF NOT: BR
1955 011760 005737 001202 11$: TST  $MSGTYPE ;;FINISHED LAST MESSAGE?
1956 011764 001375 BNE  11$ ;;IF NOT: WAIT
1957 011766 017637 000004 001204 MOV  #4(SP),$FATAL ;;GET ERROR #
1958 011774 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDR.
1959 012002 005237 001202 INC  $MSGTYPE ;;TELL APT TO TAKE ERROR
1960 012006 105037 012032 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
1961 012012 105037 012031 CLRB $LFLG ;;CLEAR LOG FLAG
1962 012016 105037 012030 CLRB $MFLG ;;CLEAR MESSAGE FLAG
1963 012022 012601 MOV  (SP)+,R1 ;;POP STACK INTO R1
1964 012024 012600 MOV  (SP)+,RO ;;POP STACK INTO RO
1965 012026 000207 RTS  PC ;;RETURN
1966 012030 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
1967 012031 000 $LFLG: .BYTE 0 ;;LOG FLAG
1968 012032 000 $FFLG: .BYTE 0 ;;FATAL FLAG
1969 012034 .EVEN
1970 000200 APTSIZE=200
1971 000001 APTENV=001

```

```

1972      000100
1973      000040
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999 012034 017646 000000
2000 012040 116637 000001 012257
2001 012046 112637 012261
2002 012052 062716 000002
2003 012056 000406
2004 012060 112737 000001 012257
2005 012066 112737 000006 012261
2006 012074 112737 000005 012256
2007 012102 010346
2008 012104 010446
2009 012106 010546
2010 012110 113704 012261
2011 012114 005404
2012 012116 062704 000006
2013 012122 110437 012260
2014 012126 113704 012257
2015 012132 016605 000012
2016 012136 005003
2017 012140 006105
2018 012142 000404
2019 012144 006105
2020 012146 006105
2021 012150 006105
2022 012152 010503
2023 012154 006103
2024 012156 105337 012260
2025 012162 100016
2026 012164 042703 177770
2027 012170 001002
    
```

```

APTSPool=100
APTCSUP=040
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT
*
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT
*
*$TYPOS: MOV      2(SP),-(SP)    ;;PICKUP THE MODE
        MOV      1(SP),SOFILL   ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)        ;;ADJUST RETURN ADDRESS
        BR      $TYPON
*$TYPOC: MOV      #1,SOFILL     ;;SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1    ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV      #5,SOCNT      ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)       ;;SAVE R3
        MOV      R4,-(SP)       ;;SAVE R4
        MOV      R5,-(SP)       ;;SAVE R5
        MOV      SOMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,SOMODE     ;;SAVE IT FOR USE
        MOV      SOFILL,R4     ;;GET THE ZERO FILL SWITCH
        MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
        CLR      R3           ;;CLEAR THE OUTPUT WORD
1$: ROL      R5               ;;ROTATE MSB INTO "C"
        BR      3$           ;;GO DO MSB
2$: ROL      R5               ;;FORM THIS DIGIT
        ROL      R5
        ROL      R5
        MOV      R5,R3
3$: ROL      R3               ;;GET LSB OF THIS DIGIT
        DECB    SOMODE        ;;TYPE THIS DIGIT?
        BPL     7$           ;;BR IF NO
        BIC     #177770,R3    ;;GET RID OF JUNK
        BNE     4$           ;;TEST FOR 0
    
```

2028	012172	005704		TST	R4	:: SUPPRESS THIS 0?
2029	012174	001403		BEQ	5\$:: BR IF YES
2030	012176	005204	4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
2031	012200	052703	000060	BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
2032	012204	052703	000040	5\$:	BIS	:: MAKE ASCII IF NOT ALREADY
2033	012210	110337	012254	MOVB	R3,8\$:: SAVE FOR TYPING
2034	012214	104400	012254	TYPE	8\$:: GO TYPE THIS DIGIT
2035	012220	105337	012256	7\$:	DECB	\$OCNT
2036	012224	003347		BGT	2\$:: COUNT BY 1
2037	012226	002402		BLT	6\$:: BR IF MORE TO DO
2038	012230	005204		INC	R4	:: BR IF DONE
2039	012232	000744		BR	2\$:: INSURE LAST DIGIT ISN'T A BLANK
2040	012234	012605	6\$:	MOV	(SP)+,R5	:: GO DO THE LAST DIGIT
2041	012236	012604		MOV	(SP)+,R4	:: RESTORE R5
2042	012240	012603		MOV	(SP)+,R3	:: RESTORE R4
2043	012242	016666	000002 000004	MOV	2(SP),4(SP)	:: RESTORE R3
2044	012250	012616		MOV	(SP)+,(SP)	:: SET THE STACK FOR RETURNING
2045	012252	000002		RTI		:: RETURN
2046	012254	000	8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
2047	012255	000		.BYTE	00	:: TERMINATOR FOR TYPE ROUTINE
2048	012256	000	\$OCNT:	.BYTE	00	:: OCTAL DIGIT COUNTER
2049	012257	000	\$OFILL:	.BYTE	00	:: ZERO FILL SWITCH
2050	012260	000000	\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

2051
 2052
 2053
 2054
 2055
 2056
 2057
 2058
 2059
 2060 012262 010046
 2061 012264 016600 000002
 2062 012270 005740
 2063 012272 111000
 2064 012274 006300
 2065 012276 016000 012304
 2066 012302 000200
 2067
 2068
 2069
 2070
 2071
 2072
 2073
 2074
 2075 012304
 2076 012304 011304
 2077 012306 012060
 2078 012310 012034
 2079 012312 012074
 2080 012314 010102
 2081
 2082 012316 007376
 2083
 2084 012320 007326
 2085 012322 007610
 2086 012324 007730
 2087 012326 010326

.SBTTL TRAP DECODER

```

;*****
;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;GO TO THAT ROUTINE.

```

```

$TRAP:  MOV    RO, -(SP)      ;; SAVE RO
        MOV    2(SP), RO    ;; GET TRAP ADDRESS
        TST   -(RO)        ;; BACKUP BY 2
        MOVB  (RO), RO      ;; GET RIGHT BYTE OF TRAP
        ASL   RO           ;; POSITION FOR INDEXING
        MOV   $TRPAD(RO), RO ;; INDEX TO TABLE
        RTS   RO           ;; GO TO ROUTINE

```

.SBTTL TRAP TABLE

```

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

```

```

;          ROUTINE
;          -----
$TRPAD:
$TYPE    ;; CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
$TYPOC   ;; CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS   ;; CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON   ;; CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS   ;; CALL=TYPDS    TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)

$GTSWR   ;; CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

$CKSWR   ;; CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
$RDOCHR  ;; CALL=RDOCHR   TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
$RDLIN   ;; CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
$RDOCT   ;; CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY

```

2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132

012330 012737 012470 000024
012336 012737 000340 000026
012344 010046
012346 010146
012350 010246
012352 010346
012354 010446
012356 010546
012360 017746 166554
012364 010637 012474
012370 012737 012402 000024
012376 000000
012400 000776

012402 012737 012470 000024
012410 013706 012474
012414 005037 012474
012420 005237 012474
012424 001375
012426 012677 166506
012432 012605
012434 012604
012436 012603
012440 012602
012442 012601
012444 012600
012446 012737 012330 000024
012454 012737 000340 000026
012462 104400
012464 012476
012466 000002
012470 000000
012472 000776
012474 000000
012476 005015 047520 042527
012504 000122

000001

```
.SBTTL POWER DOWN AND UP ROUTINES
;*****
:POWER DOWN ROUTINE
$PWRDN: MOV    $SILLUP, @PWRVEC    ;; SET FOR FAST UP
        MOV    #340, @PWRVEC+2    ;; Prio:7
        MOV    RO, -(SP)          ;; PUSH RO ON STACK
        MOV    R1, -(SP)          ;; PUSH R1 ON STACK
        MOV    R2, -(SP)          ;; PUSH R2 ON STACK
        MOV    R3, -(SP)          ;; PUSH R3 ON STACK
        MOV    R4, -(SP)          ;; PUSH R4 ON STACK
        MOV    R5, -(SP)          ;; PUSH R5 ON STACK
        MOV    @SWR, -(SP)        ;; PUSH @SWR ON STACK
        MOV    SP, $SAVR6        ;; SAVE SP
        MOV    @SPWRUP, @PWRVEC    ;; SET UP VECTOR
        HALT
        BR     .-2                ;; HANG UP
;*****
:POWER UP ROUTINE
$PWRUP: MOV    $SILLUP, @PWRVEC    ;; SET FOR FAST DOWN
        MOV    $SAVR6, SP        ;; GET SP
        CLR    $SAVR6           ;; WAIT LOOP FOR THE TTY
        IS:   INC    $SAVR6       ;; WAIT FOR THE INC
        BNE   IS              ;; OF WORD
        MOV   (SP)+, @SWR        ;; POP STACK INTO @SWR
        MOV   (SP)+, R5         ;; POP STACK INTO R5
        MOV   (SP)+, R4         ;; POP STACK INTO R4
        MOV   (SP)+, R3         ;; POP STACK INTO R3
        MOV   (SP)+, R2         ;; POP STACK INTO R2
        MOV   (SP)+, R1         ;; POP STACK INTO R1
        MOV   (SP)+, R0         ;; POP STACK INTO R0
        MOV    @SPWRDN, @PWRVEC    ;; SET UP THE POWER DOWN VECTOR
        MOV    #340, @PWRVEC+2    ;; Prio:7
        TYPE   .WORD    $POWER    ;; REPORT THE POWER FAILURE
        SPWRMG: RTI              ;; POWER FAIL MESSAGE POINTER
        $SILLUP: HALT           ;; THE POWER UP SEQUENCE WAS STARTED
        BR     .-2                ;; BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0                ;; PUT THE SP HERE
        $POWER: .ASCIZ <15><12>"POWER"
        .EVEN
        .END
```


ABASE = 160060	303#	417	458	624	625	626
ACDW1 = 000000	417	460				
ACDW2 = 000000	417					
ACPUOP = 000000	417	432				
ADDW0 = 000000	417					
ADDW1 = 000000	417					
ADDW10 = 000000	417					
ADDW11 = 000000	417					
ADDW12 = 000000	417					
ADDW13 = 000000	417					
ADDW14 = 000000	417					
ADDW15 = 000000	417					
ADDW2 = 000000	417					
ADDW3 = 000000	417					
ADDW4 = 000000	417					
ADDW5 = 000000	417					
ADDW6 = 000000	417					
ADDW7 = 000000	417					
ADDW8 = 000000	417					
ADDW9 = 000000	417					
ADEVCT = 000000	417	423				
ADEVN = 000000	417	459				
RENV = 000000	417	428				
REVM = 000000	417	429				
AFATAL = 000000	417	420				
AMADR1 = 000000	417	445				
AMADR2 = 000000	417	449				
AMADR3 = 000000	417	452				
AMADR4 = 000000	417	455				
AMAMS1 = 000000	417	439				
AMAMS2 = 000000	417	447				
AMAMS3 = 000000	417	450				
AMAMS4 = 000000	417	453				
AMSGAD = 000000	417	425				
AMSGLG = 000000	417	426				
AMSGTY = 000000	417	419				
AMTYP1 = 000000	417	440				
AMTYP2 = 000000	417	448				
AMTYP3 = 000000	417	451				
AMTYP4 = 000000	417	454				
APASS = 000000	417	422				
APRIOR = 000200	305#	417				
APTCSU = 000040	1867	1973#				
APTEMV = 000001	1765	1860	1929	1971#		
APTSIZ = 000200	679	1970#				
APTSPO = 000100	1862	1931	1972#			
ASWREG = 000000	417	430				
ATESTN = 000000	417	421				
AUNIT = 000000	417	424				
AUSWR = 000000	417	431				
AVECT1 = 100360	304#	417	456	627	628	
AVECT2 = 000000	417	457				
BEGIN = 001620	329	639#				
BEGIN1 = 004654	330	1137#				
BEGIN2 = 0047'0	331	1150#				
BITSAV = 001606	633#	952#	955	956	961*	967* 969 970 975*

EM2	005104	487	1189*																
EM20	006016	572	1274*																
EM21	006111	578	1284*																
EM22	006206	584	1295*																
EM23	006303	590	1306*																
EM24	006341	596	1312*																
EM25	006377	602	1318*																
EM26	006434	608	1323*																
EM27	006471	614	1328*																
EM3	005137	493	1194*																
EM30	006527	620	1334*																
EM4	005167	499	1199*																
EM5	005227	505	1205*																
EM6	005270	511	1211*																
EM7	005327	518	1217*																
ERRVEC=	000004	291*	664	665*	676*	1691	1692*	1694*	1697*										
GNS	= ***** U	325	2076	2077	2078	2079	2080	2082	2084	2085	2086	2087							
GTSWR	= 104405	2082*																	
HEAD1	007237	686	1398*																
HT	= 000011	201*	1875	1916															
INIT	002104	684	687*	1127															
IOTVEC=	000020	296*	648*	649*															
LDR	001570	626*	668*	691*	705*	706	716*	717	727*	728	738*	739	751*	752					
		951*	953*	956*	957	968*	970*	971	982*	983*	985*	986	990*	991					
		995	1004*	1005*	1007*	1008	1012*	1013	1017	1029*	1034*	1036	1048*	1053*					
		1057	1082*	1143*	1161*	1366													
LF	= 000012	202*	1910	1916															
MESS1	007042	1140	1376*																
MESS3	007140	1153	1387*																
PBR	001566	625*	689*	690*	765*	766	776*	777	828	938*	1045*	1046*	1049	1054*					
		1056*	1067*	1070	1075*	1081*	1143	1154*	1368										
PBSR	001564	624*	687*	786*	788*	789	798*	800*	801*	802	812*	814*	816	825*					
		827*	829	838*	840*	841	842	850*	852*	853	863*	864*	866	874*					
		876*	877*	878	887*	891*	892*	899*	905*	913*	914*	931*	939*	941*					
		942	943	1027*	1030	1035*	1069*	1074*	1076	1083*	1138*	1141	1151*	1155					
		1157	1370	1372															
PC	=%000007	222*	1108*	1111*	1121*	1126	1464*	1762*	1768*	1826*	1865*	1884*	1891*	1898*					
		1912*	1914*	1948*	1965*														
PIRQ	= 177772	208*																	
PIRQVE=	000240	302*																	
PRO	= 000000	225*																	
PR1	= 000040	226*																	
PR2	= 000100	227*																	
PR3	= 000140	228*																	
PR4	= 000200	229*																	
PR5	= 000240	230*																	
PR6	= 000300	231*																	
PR7	= 000340	232*	911	922															
PS	= 177776	205*	206																
PSW	= 177776	206*	893*	912*	923*	937*													
PWRVEC=	000024	297*	654*	655*	2093*	2094*	2103*	2109*	2121*	2122*									
RDCHR	= 104407	1523	2085*																
RDLIN	= 104410	1635	2086*																
RDOCT	= 104411	2087*																	
RESVEC=	000010	292*																	
RO	=%000000	213*	1118*	1121	1562	1572*	1576	1592	1593	1606*	1632	1636*	1637	1640					

E05

MAINDEC-11-DZLKA-A MACY11 27(732) 25-SEP-76 09:47 PAGE 58
DZLKA.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

.SASTA= ***** U	1733	1734	1788	1835*	1916	1969*	2105	2127
.SX = 001000	1921	1924						
	349*	354						

.SPTY	18	183#	1917
.SASTA	18		
.SCATC	18	183#	319
.SCMTA	18	183#	366
.SDB2D	18		
.SDB2O	18		
.SDIV	18		
.SEOP	18	183#	1094
.SERRO	18	183#	1734
.SERRT	18	183#	1789
.SMULT	18		
.SPARM	183#		
.SPOHE	18	183#	2089
.SRAND	18		
.SRODE	18		
.SROOC	18	183#	1616
.SREAD	18	183#	1410
.SR2AZ	18		
.SSAVE	18	183#	
.SSB2D	18		
.SSB2O	18		
.SSCOP	18	183#	1669
.SSIZE	18		
.SSPAC	183#		
.SSUPR	18		
.SSMDO	183#		
.STRAP	18	183#	2052
.STYPB	18		
.STYPD	18	183#	1549
.STYPE	18	183#	1837
.STYPO	18	183#	1974
.S4OCA	18		
.1170	18		

ADD	690	691	694	1160	1449	1458	1581	1653	1811	1873	1934	1946	1958	2002	2012
ASL	1472	1473	1474	1646	1648	1650	1808	1809	1810	2064					
ASLB	1586														
ASR	1941														
BCC	744	758	962	976	1587										
BEQ	680	708	719	730	741	754	768	779	791	804	818	631	844	855	868
	880	945	959	973	988	997	1010	1019	1032	1038	1051	1060	1072	1079	1091
	1119	1429	1456	1471	1641	1701	1703	1705	1709	1718	1751	1754	1777	1780	1813
	1818	1831	1863	1876	1911	1928	1932	1952	1954	2029					
BGE	1721														
BGT	1110	1468	1509	1595	1643	2036									
BHI	1707														
BIC	695	696	907	955	956	1077	1107	1158	1425	1442	1469	1496	1502	1510	1652
	2026														
BICB	1007	1016													
BIS	800	876	970	1476	1589	1590	2031	2032							
BISB	985	994	1800												
BIT	1090	1686	1700	1708	1715	1753	1760	1776							
BITB	679	1862	1867	1899	1931										
BLOS	1522														
BLT	1466	1507	1578	1594	1645	1890	2037								
BMI	1585														
BNE	645	669	684	1421	1427	1447	1454	1461	1498	1504	1526	1532	1583	1687	1716
	1761	1766	1784	1801	1823	1861	1868	1870	1878	1886	1900	1907	1930	1936	1939
	1956	2027	2113												
BPL	1142	1156	1423	1439	1494	1500	1569	1599	1773	1855	1904	2025			
BR	671	897	917	920	928	1144	1162	1450	1477	1479	1505	1528	1580	1597	1654
	1667	1689	1695	1698	1711	1714	1771	1806	1833	1857	1883	1893	1902	1909	1922
	1944	2003	2018	2039	2105	2127									
CLR	643	657	658	678	698	775	776	786	798	799	801	812	813	825	826
	838	839	850	862	863	874	875	877	887	890	893	899	905	926	931
	938	939	951	968	982	1004	1027	1028	1034	1045	1053	1054	1055	1067	1068
	1074	1081	1082	1083	1104	1105	1138	1151	1154	1436	1437	1572	1575	1638	1639
	1713	1728	1799	2016	2111										
CLR8	1533	1601	1663	1712	188	1908	1960	1961	1962						
CMP	644	668	707	718	729	740	753	767	778	790	803	817	830	843	854
	867	879	898	918	929	944	958	972	987	996	1009	1018	1031	1037	1050
	1058	1059	1071	1078	1420	1426	1446	1453	1465	1467	1497	1503	1506	1508	1521
	1593	1696	1720	1783											
CMP8	1428	1460	1525	1531	1642	1644	1702	1706	1765	1860	1875	1877	1885	1906	1910
	1929														
DEC	1108	1807													
DECB	1889	1892	2024	2035											
EMT	197														
HALT	325	1092	1774	1785	1856	2104	2126								
INC	992	1014	1106	1475	1579	1719	1756	1959	2030	2038	2112				
INCB	1724	1750	1912												
IOT	198														
JMP	329	330	331	1126											
JSR	1121	1464	1762	1768	1865	1884	1891	1898	1948						
MOV	635	642	646	648	649	650	651	652	653	654	655	656	660	661	664
	665	666	667	672	674	675	676	681	687	688	689	692	693	697	704
	705	706	715	716	717	726	727	728	737	738	739	750	751	752	764
	765	766	777	787	788	789	802	811	814	816	827	828	829	840	841
	842	851	852	853	861	864	866	878	888	889	891	892	908	910	911
	912	913	914	921	922	923	930	937	940	941	942	943	952	953	954

	957	967	969	971	983	984	986	990	991	993	995	1005	1006	1008	1012
	1013	1015	1017	1029	1030	1035	1036	1046	1047	1048	1049	1056	1057	1069	1070
	1075	1076	1111	1115	1118	1137	1143	1150	1157	1159	1161	1433	1457	1462	1491
	1492	1519	1520	1535	1536	1537	1538	1562	1563	1564	1565	1566	1567	1568	1573
	1576	1596	1602	1603	1604	1605	1606	1608	1609	1630	1631	1632	1633	1634	1636
	1637	1656	1657	1658	1659	1660	1691	1692	1694	1697	1710	1722	1723	1726	1727
	1730	1731	1752	1757	1778	1781	1798	1803	1812	1817	1822	1824	1828	1858	1859
	1864	1872	1887	1925	1926	1933	1937	1942	1943	1945	1947	1957	1963	1964	1999
	2007	2008	2009	2015	2022	2040	2041	2042	2043	2044	2060	2061	2065	2093	2094
	2095	2096	2097	2098	2099	2100	2101	2102	2103	2109	2110	2114	2115	2116	2117
	2118	2119	2120	2121	2122										
MOV8	659	906	1424	1441	1495	1501	1524	1529	1571	1574	1588	1591	1600	1640	1725
	1729	1759	1767	1869	1897	1905	1920	1921	1923	2000	2001	2004	2005	2006	2010
	2013	2014	2033	2063											
NEG	1570	2011													
NOP	894	895	915	916	924	925	1122	1123	1124						
RESET	639	815	865	1120											
ROL	743	757	961	975	1647	1649	1651	2017	2019	2020	2021	2023			
RTI	637	673	1463	1511	1539	1610	1661	1732	1787	1974	2045	2023			
RTS	1826	1914	1965	2066											
SEC	756														
SUB	909	1577	1758	1940											
TRAP	2068	2077	2078	2079	2080	2082	2084	2085	2086	2087					
TST	683	1141	1155	1455	1470	1582	1592	1655	1662	1693	1717	1772	1779	1830	1871
	1879	1901	1935	1953	1955	2028	2062								
TSTB	1422	1438	1493	1499	1584	1598	1704	1854	1903	1927	1938	1951			
.ASCII	410	411													
.ASCIZ	409	412	1129	1182	1189	1194	1199	1205	1211	1217	1222	1230	1236	1243	1251
	1255	1260	1264	1274	1284	1295	1306	1312	1318	1323	1328	1334	1340	1346	1352
	1358	1376	1387	1398	1543	1544	1545	1547	1834	2129					
.BLKB	1542														
.BLKW	1615														
.BYTE	375	376	381	382	390	391	399	400	401	402	428	429	439	440	447
	448	450	451	453	454	1128	1540	1541	1769	1770	1966	1967	1968	2046	2047
	2048	2049													
.DSABL	1480														
.ENABL	1	183	1413												
.END	2132														
.ENDC	188	197	289	303	315	317	318	319	330	336	340	342	347	349	356
	369	373	375	403	407	408	409	410	414	417	439	447	450	453	456
	457	458	459	460	463	479	646	647	650	652	654	656	657	658	660
	662	683	701	702	703	704	709	712	713	714	715	720	723	724	725
	726	731	734	735	736	737	742	747	748	749	750	755	761	762	763
	764	769	772	773	774	775	780	783	784	785	786	792	795	796	797
	798	805	808	809	810	811	812	819	822	823	824	825	832	835	836
	837	838	845	847	848	849	850	856	858	859	860	861	862	869	871
	872	873	874	881	884	885	886	887	898	902	903	904	905	921	929
	934	935	936	937	946	948	949	950	951	964	965	966	967	979	980
	981	982	998	1001	1002	1003	1004	1020	1024	1025	1026	1027	1033	1039	1042
	1043	1044	1045	1052	1061	1064	1065	1066	1067	1073	1080	1087	1088	1089	1090
	1092	1097	1098	1099	1101	1104	1110	1113	1114	1118	1120	1126	1128	1129	1132
	1134	1135	1136	1137	1147	1148	1149	1150	1413	1414	1416	1444	1480	1484	1512
	1513	1520	1522	1525	1527	1543	1549	1552	1619	1625	1669	1672	1675	1680	1686
	1688	1699	1702	1703	1704	1706	1708	1715	1719	1724	1726	1730	1733	1734	1737
	1740	1750	1757	1762	1763	1764	1772	1783	1787	1788	1792	1807	1836	1840	1869
	1920	1921	1924	1951	1966	1977	2055	2061	2064	2076	2077	2078	2079	2080	2081

	2082	2083	2084	2085	2086	2087	2088	2092	2101	2102	2108	2114	2115	2125	2132
.EQUIV	197	198	206	221	222	251	252	253	254	255	256	257	258	259	260
	279	280	281	282	283	284	285	286	287	288					
.EVEN	417	1364	1408	1835	1969	2131									
.IF	184	195	261	289	314	316	317	318	319	328	335	338	340	346	348
	355	368	372	374	403	407	408	409	413	414	416	439	447	450	453
	456	457	458	459	460	461	463	479	641	646	648	650	652	654	656
	657	658	660	678	700	702	704	708	711	713	715	719	722	724	726
	730	733	735	737	741	746	748	750	754	760	762	764	768	771	773
	775	779	782	784	786	791	794	796	798	804	807	809	811	812	818
	821	823	825	831	834	836	838	844	846	848	850	855	857	859	861
	862	868	870	872	874	880	883	885	887	897	901	903	905	920	928
	933	935	937	945	947	949	951	963	965	967	978	980	982	997	1000
	1002	1004	1019	1023	1025	1027	1032	1038	1041	1043	1045	1051	1060	1063	1065
	1067	1072	1079	1086	1088	1090	1091	1096	1097	1098	1099	1100	1101	1103	1109
	1112	1114	1118	1120	1126	1128	1129	1133	1135	1137	1146	1148	1150	1412	1414
	1415	1416	1444	1483	1484	1512	1520	1521	1525	1526	1542	1543	1549	1551	1618
	1621	1637	1671	1674	1679	1685	1686	1698	1700	1701	1702	1704	1705	1706	1715
	1717	1725	1727	1732	1733	1734	1736	1739	1750	1753	1760	1762	1763	1765	1772
	1776	1783	1787	1788	1791	1806	1822	1839	1860	1919	1921	1924	1951	1966	1976
	2054	2060	2064	2068	2077	2078	2079	2080	2081	2082	2084	2085	2086	2087	2088
.IFF	2091	2101	2102	2107	2114	2115	2123	2125	2129						
	195	314	317	318	319	336	340	342	347	349	356	369	372	375	403
	414	417	646	701	702	703	704	709	712	713	714	715	720	723	724
	725	726	731	734	735	736	737	741	747	748	749	750	754	761	762
	763	764	769	772	773	774	775	780	783	784	785	786	792	795	796
	797	798	805	808	809	810	811	819	822	823	824	825	832	835	836
	837	838	845	847	848	849	850	856	858	859	860	861	869	871	872
	873	874	881	884	885	886	887	897	902	903	904	905	921	929	934
	935	936	937	946	948	949	950	951	964	965	966	967	979	980	981
	982	998	1001	1002	1003	1004	1020	1024	1025	1026	1027	1032	1039	1042	1043
	1044	1045	1051	1061	1064	1065	1066	1067	1072	1079	1087	1088	1089	1090	1091
	1097	1100	1104	1110	1113	1128	1134	1135	1136	1137	1147	1148	1149	1150	1413
	1416	1484	1486	1491	1512	1513	1522	1526	1543	1552	1619	1672	1699	1702	1703
	1706	1733	1734	1737	1739	1753	1783	1788	1792	1807	1836	1840	1920	1977	2055
.IFT	2061	2092	2108	2125											
.IFTF	1486	1491	1642	1662	1669	1714	1763								
.IIF	1431	1484	1487	1638	1646	1668	1712	1762							
	183	188	193	311	312	313	315	318	319	325	413	417	647	650	656
	657	658	660	661	1098	1104	1105	1116	1128	1132	1413	1434	1535	1543	1549
	1669	1675	1676	1677	1678	1679	1680	1684	1713	1714	1730	1733	1734	1740	1741
	1742	1743	1744	1749	1775	1783	1788	1804	1829	1916	2076	2077	2078	2079	2080
.IRP	2082	2084	2085	2086	2087										
	479	700	711	722	733	746	760	771	782	794	807	821	834	846	857
	870	883	901	933	947	963	978	1000	1023	1041	1063	1086	1133	1146	1562
.LIST	1602	1632	1658	1685	1925	1926	1947	1963	1964	2095	2101	2114	2115		
	1	183	303	318	325	403	405	406	407	414	417	479	662	700	704
	711	715	722	726	733	737	746	750	760	764	771	775	782	786	794
	798	807	811	821	825	834	838	846	850	857	861	870	874	883	887
	901	905	933	937	947	951	963	967	978	982	1000	1004	1023	1027	1041
	1045	1063	1067	1086	1090	1104	1120	1133	1137	1146	1150	1512	1679	1783	2068
	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088		
.MACRO	1	319	366	678	2068										
.MCALL	183	303	414	662											
.MEXIT	462														
.NLIST	1	183	303	318	325	403	405	406	407	414	417	479	662	700	704

	711	715	722	726	733	737	746	750	760	764	771	775	782	786	794
	798	807	811	821	825	834	838	846	850	857	861	870	874	883	887
	901	905	933	937	947	951	963	967	978	982	1000	1004	1023	1027	1041
	1045	1063	1067	1086	1090	1104	1120	1133	1137	1146	1150	1512	1679	1783	2068
.PAGE	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088		
.REM	366	463													
.REPT	1														
.SBTTL	325	405													
	193	307	319	328	333	344	366	414	463	640	700	711	722	733	746
	760	771	782	794	807	821	834	846	857	870	883	901	933	947	963
	978	1000	1023	1041	1063	1086	1094	1133	1146	1181	1410	1549	1616	1669	1734
.TITLE	1789	1837	1917	1974	2052	2068	2089								
.WORD	183														
	325	326	327	341	360	361	362	363	364	365	374	377	378	379	380
	383	384	385	386	387	388	389	392	393	394	403	405	406	419	420
	421	422	423	424	425	426	430	431	432	445	449	452	455	456	457
	458	459	460	1109	1112	1127	1665	1668	1815	1820	1866	1913	1949	2050	2124

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

* DZLKAA.SEG/SOL/CRF/PAGNUM/ML:TOC=DZLKAA.SML,DZLKAA.P11
 RUN-TIME: 36 44 5 SECONDS
 RUN-TIME RATIO: 156/86=1.8
 CORE USED: 33K (65 PAGES)

