

DR11-K

DIGITAL I/O TEST
MD-11-DZDRG-C

EP DZDRG-C-DE A
COPYRIGHT 1976
FICHE 1 OF 1

NOV 1976
digital
MADE IN USA

This microfiche card contains a grid of frames, each displaying digital test data. The data is organized into columns and rows, with some frames containing headers or specific test results. The frames are arranged in a regular grid pattern, typical of microfiche storage.



.HEM.

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZDRG-C-D
PRODUCT NAME:	DR11-K DIGITAL I/O TEST
DATE:	MAY 21, 1976
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	RAYMOND SHOOP

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1976 BY DIGITAL EQUIPMENT CORPORATION.

MAINTENANCE TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 1

DR11 LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 2
DZDRG-C.P11

1. ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE DR11-K DIGITAL INPUT OUTPUT CONTROL OPTION. ALL FUNCTIONS OF THE OPTION WILL BE TESTED. DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR MAY BE REQUIRED TO SUPPLY OPTION CHARACTERISTICS. THE PROGRAM WILL HANDLE ALL CONFIGURATIONS OF INPUT INTERRUPT SWITCHES AND INPUT DATA LATCHING JUMPERS. FOR SYSTEMS WITH CONSECUTIVE MULTIPLE DR11-K'S, THESE CONFIGERATIONS MUST BE THE SAME. THE FOLLOWING JUMPERS MUST BE INSERTED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A. THIS PROGRAM WILL TEST SEQUENTIAL DR11-K'S STARTING AT THE BUS ADDRESS AND VECTOR IN LOCATIONS "BASEBA" AND "BASEI". FOR NORMAL FACTORY CONFIG., ALL SWITCH REGISTER BITS SHOULD BE RESET.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH TELETYPE
DR11-K OPTION INSTALLED
BCOBR-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <↑G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3. AFTER THE "NEW=" HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <↑U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	HALT ON ERROR
SW 14 = 1	LOOP ON CURRENT SUB-TEST
SW 13 = 1	INHIBIT ERROR TYPINGS
SW 12 = 1	LOOP ON CURRENTLY SELECTED DR11-K
SW 11 = 1	INHIBIT SUB-TEST INTERACTIONS
SW 10 = 1	OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
SW 09 = 1	LOOP ON ERROR
SW 08 = 1	LOOP ON TEST IN SWR 17:0

1-DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 3
DZDRG-C.P11

E01

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

4.2 STARTING ADDRESS OR ADDRESSES

200 IS THE STARTING ADDRESS OF THE LOGIC TEST.
204 IS THE RESTART ADDRESS OF THE LOGIC TEST.
210 IS THE STARTING ADDRESS OF THE CONTROL LINE LOOP.
214 IS THE STARTING ADDRESS OF THE COULTER INTERFACE LCCP

5. OPERATING PROCEDURE

THE FOLLOWING JUMPERS MUST BE INSTALLED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A
IF THE CUSTOMER HAS SELECTED THE "B" SECTION OF THESE JUMPERS IT MUST BE RETURNED TO THE "FACTORY" POSITION BEFORE RUNNING THE LOGIC TEST. ** WORST CASE WILL ONLY BE CHANGING THREE JUMPERS. **
THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR. WITH THE INPLT INTERRUPT SWITCHES AND DATA LATCHING JUMPERS IN THE FACTORY POSITION, ALL SWITCH REGISTER BITS SHOULD BE RESET. ONCE STARTED THE TEST WILL RUN IN IT'S NORMAL MANNER WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.

*** NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY PROGRAM. THE MACHINE WILL TYPE: SWR=XXXXXXNEW= AFTER NEW= THE INFORMATION IS INSERTED. REFER TO SECTION 4.1 2) ***

6. ERRORS

THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR ERROR REPORTING AND TYPEOUT. REFER TO THE "ERROR POINTER TABLE" FOR TYPE AND DESCRIPTION OF ERRORS.

7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH Jumper CONFIGURATION.
2. THE FOLLOWING JUMPERS MUST BE IN THE "FACTORY" POSITION:
W21A, W22A AND W23A
3. THE OPERATOR MUST SUPPLY THE CORRECT INTERRUPT SWITCHES AND DATA PATH JUMPER JUMPER AND SWITCH CONFIGURATION INFORMATION TO THE INITIALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:
THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 60 SECONDS FOR COMPLETION ON A PD11-05 TYPE AND WILL TYPE 'END PASS'.
THE CONTROL LINE LOOP WILL NEVER EXIT.
THE COULTER INTERFACE LOOP WILL NEVER EXIT.

F01

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 5
DZDRGC.P11

:S2

183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231

8.2 DEVICE ADDRESS PROGRAM LOCATIONS (AT APPROX. 1300)

LOCATION BASEAD CONTAINS THE DR11-K BASE DEVICE ADDRESS (767770)
LOCATION BASEIV CONTAINS THE DR11-K BASE INTERRUPT VECTOR (300)
LOCATION BASEBR CONTAINS THE DR11-K BR LEVEL (200)(4)

*NOTE: IF THESE LOCATIONS ARE CHANGED, THE OPERATOR MUST START THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE ADDRESS AND VECTOR AND UPDATE THE ACTUAL PROGRAM VALUES.

9. PROGRAM DESCRIPTION

9.1 LOGIC TEST

THE LOGIC TEST IS A TEST OF THE CONTROL AND INPUT/OUTPUT REGISTERS. ALL JUMPERS AND SWITCHES COMBINATIONS EXCEPT:

W21A, W22A AND W23A CAN BE DIAGNOSED.

THE PROGRAM CHECKS THAT THE DR11-K CAN INTERRUPT AND THAT "RESET" WILL WORK CORRECTLY.

9.2 CONTROL LINE LOOP

THIS TEST LOOP PROVIDES THE OPERATOR WITH A SCOPE LOOP FOR CHECKING W21, W22 AND W23 IN THE "B" POSITION..

9.3 COULTER FUNCTION LOOP

THE COULTER INTERFACE LOOP PROVIDES THE OPERATOR WITH SOFTWARE INTERFACE WITH THE COULTER HARDWARE. THE OPERATOR IS INSTRUCTED TO RUN A SAMPLE ON THE COULTER. THE PROGRAM WILL WAIT FOR AN INTERRUPT FROM THE DR11K. UPON AN INTERRUPT, THE PROGRAM DELAYS 70 MSEC. TO ALLOW THE DATA LINES TO SETTLE. THE INPUT DATA IS SAVED IN A CORE BUFFER. THE HIGHER FOUR BITS ARE THE TEST NUMBERS WITH ZERO BEING THE RUN TERMINATOR. UPON COMPLETION OF A RUN (SAMPLE), THE PROGRAM WILL TYPE THE RESULTS IN THE FOLLOWING FORMAT AND THEN RESTART THE RUN SEQUENCE. THE TYPED RESULTS CAN BE COMPARED TO THE COULTER TYPED OUTPUT TO VERIFY A SUCCESSFUL RUN.

"N-XXX" WHERE N = TEST NUMBER AND XXX = TEST RESULTS

10. TABLE OF CONTENTS

ATTACHED.:

232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287

.TITLE DR11-K LOGIC TEST MAINDEC-11-DZDRG-C
:*COPYRIGHT (C) 1975
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
*
:*PROGRAM BY RAYMOND SHOOP
*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
*
:REVISED 21 MAY 76 BY F. ROEMER
:TO SUPPORT DYNAMIC LOADING OF SOFTWARE SWITCH REGISTER

.SBTTL BASIC DEFINITIONS

001100
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT.ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT.SCOPE ;;BASIC DEFINITION OF SCOPE CALL

.*MISCELLANEOUS DEFINITIONS

000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
000012 LF= 12 ;;CODE FOR LINE FEED
000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

.*GENE. PURPOSE REGISTER DEFINITIONS

000000 R0= %0 ;;GENERAL REGISTER
000001 R1= %1 ;;GENERAL REGISTER
000002 R2= %2 ;;GENERAL REGISTER
000003 R3= %3 ;;GENERAL REGISTER
000004 R4= %4 ;;GENERAL REGISTER
000005 R5= %5 ;;GENERAL REGISTER
000006 R6= %6 ;;GENERAL REGISTER
000007 R7= %7 ;;GENERAL REGISTER
.EQUIV R6,SP ;;STACK POINTER
.EQUIV R7,PC ;;PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

000000 PR0= 0 ;;PRIORITY LEVEL 0
000040 PR1= 40 ;;PRIORITY LEVEL 1
000100 PR2= 100 ;;PRIORITY LEVEL 2
000140 PR3= 140 ;;PRIORITY LEVEL 3
000200 PR4= 200 ;;PRIORITY LEVEL 4
000240 PR5= 240 ;;PRIORITY LEVEL 5
000300 PR6= 300 ;;PRIORITY LEVEL 6
000340 PR7= 340 ;;PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

100000 SW15= 100000


```

288      040000
289      020000
290      010000
291      004000
292      002000
293      001000
294      000400
295      000200
296      000100
297      000040
298      000020
299      000010
300      000004
301      000002
302      000001
303
304
305
306
307
308
309
310
311
312
313
314
315      100000
316      040000
317      020000
318      010000
319      004000
320      002000
321      001000
322      000400
323      000200
324      000100
325      000040
326      000020
327      000010
328      000004
329      000002
330      000001
331
332
333
334
335
336
337
338
339
340
341
342
343      000004

```

```

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

```

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

```

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

```

```

.*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4 ;; TIME OUT AND OTHER ERRORS

```

344	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
345	000014	TBITVEC=14	::"T" BIT
346	000014	TRIVEC= 14	::TRACE TRAP
347	000014	BPTVEC= 14	::BREAKPOINT TRAP (BPT)
348	000020	IOTVEC= 20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
349	000024	PWRVEC= 24	::POWER FAIL
350	000030	EMTVEC= 30	::EMULATOR TRAP (EMT) **ERROR**
351	000034	TRAPVEC=34	::"TRAP" TRAP
352	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
353	000064	TPVEC= 64	::TTY PRINTER VECTOR
354	000240	PIRQVEC=240	::PROGRAM INTERRUPT REQUEST VECTOR

```

355 .SBTTL OPERATIONAL SWITCH SETTINGS
356 : *
357 : * SWITCH USE
358 : * -----
359 : *
360 : * 15 HALT ON ERROR
361 : * 14 LOOP ON TEST
362 : * 13 INHIBIT ERROR TYPEOUTS
363 : * 12 LOOP ON CURRENTLY SELECTED DR11-K
364 : * 11 INHIBIT ITERATIONS
365 : * 10 OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
366 : * 9 LOOP ON ERROR
367 : * 8 LOOP ON TEST IN SWR<7:0>
368
369 .SBTTL TRAP CATCHER
370 : *
371 : * .=0
372 : * : *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ". +2,HALT"
373 : * : *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
374 : * : *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
375 : *
376 : * .=174
377 : *
378 : * DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
379 : * SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
380 : * .SBTTL STARTING ADDRESS(ES)
381 : * JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
382 : * JMP @#BEGIN1 ;JUMP TO THE RESTART ADDRESS
383 : * JMP @#EXTTST ;JUMP TO THE CONTROL LINES LOOP
384 : * JMP @#COULTR ;JUMP TO THE COULTER FUNCTION LOOP

```


.SBTTL COMMON TAGS

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

382
383
384
395
396
397
398
399
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428

.=1100
SCMTAG:
SPASS: .WORD 0
STSTNM: .BYTE 00
SERFLG: .BYTE 00
SICNT: .WORD 00
SLPADR: .WORD 00
SLPERR: .WORD 00
SERTTL: .WORD 00
SITEMB: .BYTE 0
SERMAX: .BYTE 1
SERRPC: .WORD 0
SGDADR: .WORD 00
SBDADR: .WORD 00
SGDDAT: .WORD 00
SBDDAT: .WORD 00
SAUTOB: .BYTE 0
SINTAG: .BYTE 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$REGAD: .WORD 0
\$REGO: .WORD 0
\$REG1: .WORD 0
\$TIMES: 0
\$ESCAPE: 0
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

;; START OF COMMON TAGS
;; CONTAINS PASS COUNT
;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED
;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR
;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
;; CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
;; CONTAINS ((\$REGAD)+0)
;; CONTAINS ((\$REGAD)+2)
;; MAX. NUMBER OF ITERATIONS
;; ESCAPE ON ERROR ADDRESS
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

;;*****

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;ITEM 1
EM1 ;STATUS REGISTER IN ERROR
DH1 ;ERRPC DRADD STATUS EXPECTED
DT1 ;\$ERRPC DRADD \$BDDAT \$GDDAT
0

;ITEM 2
EM2 ;INPUT REGISTER IN ERROR
DH2 ;ERRPC DRADD INPUT EXPECTED
DT1 ;\$ERRPC DRADD \$BDDAT \$GDDAT
0

;ITEM 3
EM3 ;OUTPUT REGISTER IN ERROR
DH3 ;ERRPC DRADD OUTPUT EXPECTED
DT1 ;\$ERRPC DRADD \$BDDAT \$GDDAT
0

;ITEM 4
EM4 ;INPUT FAILED TO INTERRUPT
DH4 ;ERRPC DRADD
DT4 ;\$ERRPC DRADD
0

;ITEM 5
EM5 ;OUTPUT FAILED TO INTERRUPT
DH4 ;ERRPC DRADD
DT4 ;\$ERRPC DRADD
0

;ITEM 6
EM6 ;UNEXPECTED INTERRUPT
DH4 ;ERRPC DRADD
DT4 ;\$ERRPC DRADD
0

429
430
431
432
433
434
435
436
437
438
439
440
441
442
443 001176
444
445
446 001176 012322
447 001200 012742
448 001202 013176
449 001204 000000
450
451
452 001206 012353
453 001210 013003
454 001212 013176
455 001214 000000
456
457
458 001216 012403
459 001220 013044
460 001222 013176
461 001224 000000
462
463
464 001226 012434
465 001230 013105
466 001232 013210
467 001234 000000
468
469
470 001236 012466
471 001240 013105
472 001242 013210
473 001244 000000
474
475
476 001246 012521
477 001250 013105
478 001252 013210
479 001254 000000
480

481			:ITEM	7	
482	001256	012546		EM7	:OPERATOR INTERVENTION ERROR
483	001260	013105		DH4	:ERRPC DRADD
484	001262	013210		DT4	:SERRPC DRADD
485	001264	000000		0	
486					
487			:ITEM	10	
488	001266	012602		EM10	:INTERRUPT INPUT BIT FAILED TO SET INPUT READY
489	001270	013123		CH10	:ERRPC DRADD STATUS EXPECTED INPLT BIT
490	001272	013216		DT10	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3
491	001274	000000		0	
492					
493			:ITEM	11	
494	001276	012665		EM11	:NON-INTERRUPTING INPUT BIT SET INPUT READY
495	001300	013123		DH10	:ERRPC DRADD STATUS EXPECTED INPUT BIT
496	001302	013216		DT10	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3
497	001304	000000		0	
498					
499	001306	167770	BASEBA:	167770	:STARTING BASE BUS ADDRESS
500	001310	000300	BASEIV:	300	:STARTING BASE INTERRUPT ADDRESS
501	001312	000000	NMBEXT:	0	:ADDITIONAL DR-11-K
502	001314	000000	NBEXT:	0	
503					
504	001316	167770	DRADD:	167770	:CURRENT DR11-K STARTING ADDRESS
505	001320	000300	DRIV:	300	:CURRENT DR11-K STARTING INTERRUPT VECTOR
506	001322	000200	DRBRL:	200	:DRA BR LEVEL
507					
508					
509	001324	167770	GRSTAT:	167770	:DR STATUS
510	001326	167772	GRDAI:	167772	:INPUT REG.
511	001330	167774	GRDIO:	167774	:OUTPUT REG.
512	001332	167775	GRGHIO:	167775	:OUTPUT REG HIGH BYTE
513	001334	000000	NOTLCH:	0	
514	001336	000000	INTBIT:	0	
515	001340	000300	GRIVA:	300	
516	001342	000302	GRIVSA:	302	
517	001344	000304	GRIVB:	304	
518	001346	000306	GRIVSB:	306	
519	001350	000200	DIOBRL:	200	
520	001352	000000	BRLEV1:	0	
521	001354	000000	BRLEV2:	0	
522	001356	000000	BRLEV3:	0	
523	001360	000000	ODDJMP:	0	
524	001362	000000	MINSIN:	0	
525	001364	000000	TRANST:	0	
526					

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082

001366 005000
001370 000402
001372 012700 177777
001376 000005
001400 012706 001100
001404 012737 001432 000004
001412 013702 001306
001416 005003
001420 005712
001422 162702 000010
001426 005203
001430 000773
001432 022626
001434 005703
001436 001001
001440 000000

001442 005303
001444 010337 001312
001450 012737 000006 000004
001456 005037 000005
001462 000005

001464 012706 001100
001470 005026
001472 022706 001140
001476 001374
001500 012706 001100

001504 012737 013232 000020
001512 012737 000340 000022
001520 012737 013504 000030
001526 012737 000340 000032
001534 012737 015566 000034
001542 012737 000340 000036
001550 012737 014224 000024
001556 012737 000340 000026
001564 013737 011412 011404
001572 005037 001166
001576 005037 001170
001602 012737 000001 001115
001610 012737 001610 001106
001616 012737 001616 001110

001624 013746 000004
001630 012737 001664 000004
001636 012737 177570 001140
001644 012737 177570 001142
001652 022777 177777 177260
001660 001012

```
::*****  
: DIGITAL I-O LOGIC TEST  
:*****  
BEGIN: CLR R0 ;CLEAR R0  
BR RBEG  
BEGIN1: MOV #-1,R0 ;LOAD R0  
RBEG: RESET  
MOV #STACK,SP ;LOAD STACK  
MOV #16,2#4 ;LOAD BUS ERROR  
MOV #BASEBA,R2 ;LOAD STARTING ADDRESS  
CLR R3 ;CLEAR COUNT  
25: TST (R2) ;TEST IF EXISTENT  
SUB #10,R2 ;EXIST, UPDATE TEST ADDRESS  
INC R3 ;UPDATE # OF DR11'S  
BR 25  
15: CMP (SP)+,(SP)+ ;POP STACK  
TST R3 ;TEST IF FIRST DOES EXIST  
BNE 35 ;BR  
HALT ;FIRST DR11-K DOES NOT EXIST  
;CHECK THE PROGRAM DEVICE ADDRESS  
35: DEC R3 ;ADJUST R3  
MOV R3,NMBEXT ;SAVE THE NUMBER OF ADDITIONAL DR11-K  
MOV #6,2#4 ;RESET BUS ERROR  
CLR 2#6  
RBEG1: RESET  
:SBTTL INITIALIZE THE COMMON TAGS  
: :CLEAR THE COMMON TAGS ($CMTAG) AREA  
MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED  
CLR (R6)+ ;:CLEAR MEMORY LOCATION  
CMP #SWR,R6 ;:DONE?  
BNE -6 ;:LOOP BACK IF NO  
MOV #STACK,SP ;:SETUP THE STACK POINTER  
: :INITIALIZE A FEW VECTORS  
MOV #SCOPE,2#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE  
MOV #340,2#IOTVEC+2 ;:LEVEL 7  
MOV #ERRR,2#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE  
MOV #340,2#EMTVEC+2 ;:LEVEL 7  
MOV #STRAP,2#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS  
MOV #340,2#TRAPVEC+2 ;:LEVEL 7  
MOV #SPWRDN,2#PWAVEC ;:POWER FAILURE VECTOR  
MOV #340,2#PWAVEC+2 ;:LEVEL 7  
MOV #SENDCT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER  
CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS  
CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS  
MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST  
MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE  
MOV #,$SLPERR ;:SETUP THE ERROR LOOP ADDRESS  
: :SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS  
: :EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.  
MOV 2#ERRVEC,-(SP) ;:SAVE ERROR VECTOR  
MOV #64,$ERRVEC ;:SET UP ERROR VECTOR  
MOV #DSWA,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER  
MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER  
CMP #-1,$SWR ;:TRY TO REFERENCE HARDWARE SWR  
BNE 663 ;:BRANCH IF NO TIMEOUT TRAP OCCURRED  
: :AND THE HARDWARE SWR IS NOT = -1
```

```

00000000 001662 000403      BR      655      ::BRANCH IF NO TIMEOUT
00000001 001664 012716 001672    645:    MOV      #655,(SP)  ::SET UP FOR TRAP RETURN
00000002 001665 000000      RTI
00000003 001667 012737 000176 001140  655:    MOV      #SWREG,SWR  ::POINT TO SOFTWARE SWR
00000004 001700 012737 000174 001142      MOV      #DISPREG,DISPLAY
00000005 001706 012637 000004 665:    MOV      (SP)+,2#ERRVEC  ::RESTORE ERROR VECTOR

```

```

590 001712 005737 000042      TST      0042      ;TEST IF UNDER A MONITOR
591 001716 001002           BNE      45       ;BR IF
592 001720 005700           TST      R0       ;TEST R0
593 001722 001402           BEQ      25       ;BR IF CLEARED
594 001724 000137 003316      JMP      IOTEST   ;JUMP IF SET
595 001730           45:
596 001730 104401 001736           25:
597 001734 000426           TYPE     68$     ;;TYPE ASCIZ STRING
598 001734 000426           BR       67$     ;;GET OVER THE ASCIZ
599 002012           68$: .ASCIZ <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
600 002012 104401 002020           67$:
601 002016 000414           TYPE     70$     ;;TYPE ASCIZ STRING
602 002016 000414           BR       69$     ;;GET OVER THE ASCIZ
603 002050           70$: .ASCIZ <15><12>/MAINDEC-11-DZDRG-C/<15><12>
604 002050 013746 001312           69$:
605 002054 104403           MOV      NMBEXT,-(SP)
606 002056 000002           TYPOS
607 002060 104401 002066           .WORD   2
608 002064 000421           TYPE     72$     ;;TYPE ASCIZ STRING
609 002064 000421           BR       71$     ;;GET OVER THE ASCIZ
610 002130           72$: .ASCIZ /(8) ADDITIONAL DR11-K'S CONNECTED/
611 002130 104401 002136           71$:
612 002134 000435           TYPE     74$     ;;TYPE ASCIZ STRING
613 002134 000435           BR       73$     ;;GET OVER THE ASCIZ
614 002230           74$: .ASCIZ <15><12>/SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING BITS/
615 002230 104401 002236           73$:
616 002234 000410           TYPE     76$     ;;TYPE ASCIZ STRING
617 002234 000410           BR       75$     ;;GET OVER THE ASCIZ
618 002256           76$: .ASCIZ <15><12>/DEPRESS CONT./
619 002256 000000           75$:
620 002260 000042           .SBTTL  HALT      ;WAIT FOR OPERATOR
621 002260 005737 000042           GET VALUE FOR SOFTWARE SWITCH REGISTER
622 002264 001006           TST      0042     ;;ARE WE RUNNING UNDER XXDP/ACT?
623 002266 023727 001140 000176      BNE      77$     ;;BRANCH IF YES
624 002274 001005           CMP      SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
625 002276 104406           BNE      78$     ;;BRANCH IF NO
626 002300 000403           GTSWR
627 002302 112737 000001 001134      BR       78$     ;;GET SOFT-SWR SETTINGS
628 002310           77$: MOV      #1,$AUTOB ;;SET AUTO-MODE INDICATOR
629 002310 017737 176624 001334      78$:
630 002316 104401 002324           MOV      $SWR,NOTLCH ;SAVE SWITCHES
631 002322 000435           TYPE     80$     ;;TYPE ASCIZ STRING
632 002322 000435           BR       79$     ;;GET OVER THE ASCIZ
633 002416           80$: .ASCIZ <15><12>/SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING BITS/
634 002416 104401 002424           79$:
635 002422 000410           TYPE     82$     ;;TYPE ASCIZ STRING
636 002422 000410           BR       81$     ;;GET OVER THE ASCIZ
637 002444           82$: .ASCIZ <15><12>/DEPRESS CONT./
638 002444 000000           81$:
639 002446 000042           .SBTTL  HALT      ;WAIT FOR OPERATOR
640 002446 005737 000042           GET VALUE FOR SOFTWARE SWITCH REGISTER
641 002452 001006           TST      0042     ;;ARE WE RUNNING UNDER XXDP/ACT?
642 002454 023727 001140 000176      BNE      83$     ;;BRANCH IF YES
643 002462 001005           CMP      SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
644 002464 104406           BNE      84$     ;;BRANCH IF NO
645 002466 000403           GTSWR
645 002466 000403           BR       84$     ;;GET SOFT-SWR SETTINGS

```



```

646 002470 112737 000001 001134 93$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
647 002476 84$:
648 002476 017737 176436 001336 MOV @SWR,INTBIT ;SAVE SWITCHES
649 002504 104401 002512 TYPE #86$ ;;TYPE ASCIZ STRING
650 002510 000437 BR #85$ ;;GET OVER THE ASCIZ
651 ::86$: .ASCIZ <15><12>/SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
652 002610 85$:
653 002610 104401 002616 TYPE #88$ ;;TYPE ASCIZ STRING
654 002614 000410 BR #87$ ;;GET OVER THE ASCIZ
655 ::88$: .ASCIZ <15><12>/DEPRESS CONT./
656 002636 87$:
657 002636 000000 HALT
658 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
659 002640 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
660 002644 001006 BNE #89$ ;;BRANCH IF YES
661 002646 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
662 002654 001005 BNE #90$ ;;BRANCH IF NO
663 002656 104406 GTSWR ;;GET SOFT-SWR SETTINGS
664 002660 000403 BR #90$
665 002662 112737 000001 001134 89$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
666 002670 90$:
667 002670 017737 176244 001362 MOV @SWR,MINSIN ;SAVE MINUS INPUT
668 002676 104401 002704 TYPE #92$ ;;TYPE ASCIZ STRING
669 002702 000440 BR #91$ ;;GET OVER THE ASCIZ
670 ::92$: .ASCIZ <15><12>/SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
671 003004 91$:
672 003004 104401 003012 TYPE #94$ ;;TYPE ASCIZ STRING
673 003010 000410 BR #93$ ;;GET OVER THE ASCIZ
674 ::94$: .ASCIZ <15><12>/DEPRESS CONT./
675 003032 93$:
676 003032 000000 HALT
677 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
678 003034 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
679 003040 001006 BNE #95$ ;;BRANCH IF YES
680 003042 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
681 003050 001005 BNE #96$ ;;BRANCH IF NO
682 003052 104406 GTSWR ;;GET SOFT-SWR SETTINGS
683 003054 000403 BR #96$
684 003056 112737 000001 001134 95$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
685 003064 96$:
686 003064 017737 176650 001364 MOV @SWR,TRANST ;SAVE SWITCHES
687
688 .SBTTL DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
689 003072 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD TEST BIT
690 003100 033737 001124 001362 10$: BIT $GDDAT,MINSIN ;TEST IF NEG INPUT BIT
691 003106 001407 BEQ #11$ ;BR IF NOT
692 003110 033737 001124 001364 BIT $GDDAT,TRANST ;TEST IF TRANS. INPUT
693 003116 001403 BEQ #11$ ;BR IF NOT
694 003120 104007 ERROR 7 ;LOGICAL OPERATOR ERROR
695 ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
696 003122 000137 001366 JMP BEGIN ;
697
698 003126 006137 001124 .1$: ROL $GDDAT ;MOVE LEFT
699 003132 103362 BCC #10$ ;BR UNTIL DONE
700
701 003134 104401 003142 TYPE #98$ ;;TYPE ASCIZ STRING

```

F02

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 18
 DZDRGC.P11 DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR

```

702 003140 000434          BR      97$          ;;GET OVER THE ASCIZ
703          ;;98$: .ASCIZ <15><12>/SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
704          97$:
705 003232          TYPE      100$          ;;TYPE ASCIZ STRING
706 003232 104401 003240    BR      99$          ;;GET OVER THE ASCIZ
707 003236 000412          ;;100$: .ASCIZ <15><12>/DEPRESS CONT./<15><12><12>
708          99$:
709 003264          HALT          ;WAIT FOR OPERATOR
710 003264 000000          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
711 003266 005737 000042    TST      0#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
712 003272 001076          BNE      101$          ;;BRANCH IF YES
713 003274 02. 07 001140 000176  CMP      SWR,#SWREG          ;;SOFTWARE SWITCH REG SELECTED?
714 003302 001005          SNE      102$          ;;BRANCH IF NO
715 003304 104406          GTSWR          ;;GET SOFT-SWR SETTINGS
716 003306 000403          BR      102$
717 003310 112737 000001 001134 101$: MOVB     #1,$AUTOB          ;;SET AUTO-MODE INDICATOR
718 003316          102$:
719 003316 000005          IOTEST: RESET
720 003320 013737 001306 001316  MOV     BASEBA,DRADD          ;LOAD INITIAL STARTING ADDRESS
721 003326 013737 001310 001320  MOV     BASEIV,DRIV          ;LOAD INITIAL STARTING VECTOR
722 003334 013737 001312 001314  MOV     NMBEXT,NBEXT          ;RELOAD * AVAILABLE
723 003342 000005          RSEG2: RESET
724 003344 012702 000232          MOV     #232,R2          ;LOAD R2
725 003350 012701 000230          MOV     #230,R1          ;LOAD R1
726 003354 010221          5$: MOV     R2,(R1)+          ;LOAD .+2
727 003356 005021          CLR     (R1)+          ;LOAD HALT
728 003360 010102          MOV     R1,R2          ;LOAD R2
729 003362 005722          TST     (R2)+          ;BUMP R2
730 003364 020227 001076          CMP     R2,#$CMTAG-2          ;TEST FOR LAST
731 003370 001371          BNE     5$          ;BR UNTIL DONE
  
```

```

732 003372 000005          IOTST1: RESET
733 003374 004737 003402      JSR      PC,SETADD          ;SET UP BUS ADDRESS AND VECTOR
734 003400 000453          BR       IOTTS1
735 003402 013737 001316 001324  SETADD: MOV    DRADD,GRSTAT      ;LOAD 1ST ADDRESS
736 003410 013737 001316 001326      MOV    DRADD,GRDAI          ;LOAD 2ND ADDRESS
737 003416 062737 000002 001326      ADD    #2,GRDAI
738 003424 013737 001316 001330      MOV    DRADD,GRDIO         ;LOAD 3RD ADDRESS
739 003432 062737 000004 001330      ADD    #4,GRDIO
740 003440 013737 001316 001332      MOV    DRADD,GRBHIO        ;LOAD 4TH ADDRESS
741 003446 062737 000005 001332      ADD    #5,GRBHIO
742 003454 013737 001320 001340      MOV    DRIV,GRIVA          ;LOAD FIRST VECTOR
743 003462 013737 001320 001342      MOV    DRIV,GRIVSA
744 003470 062737 000002 001342      ADD    #2,GRIVSA
745 003476 013737 001320 001344      MOV    DRIV,GRIVB          ;LOAD 2ND VECTOR
746 003504 062737 000004 001344      ADD    #4,GRIVB
747 003512 013737 001320 001346      MOV    DRIV,GRIVSB
748 003520 062737 000006 001346      ADD    #6,GRIVSB
749 003526 000207          RTS      PC
750 003530 013737 001322 001350  IOTTS1: MOV    DRBRL,DIOBRL        ;LOAD BR LEVEL
751 003536 005037 001360          CLR     ODDJMP
752 003542 053737 001362 001360      BIS    MINSIN,ODDJMP        ;SET MINUS INPUT BITS
753 003550 053737 001364 001360      BIS    TRANST,ODDJMP       ;SET TRANSITION INPUT BITS
754 003556 012706 001100          MOV    #STACK,SP           ;LOAD STACK
755 003562 013777 001342 175550      MOV    GRIVSA,@GRIVA        ;RESET INPUT VECTOR
756 003570 005077 175546          CLR    @GRIVSA
757 003574 013777 001346 175542      MOV    GRIVSB,@GRIVB        ;RESET OUTPUT VECTOR
758 003602 005077 175540          CLR    @GRIVB
759                                     ;*****
760                                     ;*TEST 1          TEST FOR NO BUS ERRORS
761                                     ;*****
762 003606 000004          †ST1:  SCOPE
763 003610 005077 175510          CLR     @GRSTAT
764 003614 005077 175506          CLR     @GRDAI
765 003620 005077 175504          CLR     @GRDIO
766
767                                     ;*****
768                                     ;*TEST 2          TEST THAT OUTPUT REG. CAN HOLD #-1
769                                     ;*****
770 003624 000004          †ST2:  SCOPE
771 003626 012737 177777 001124      MOV    #-1,$GDDAT          ;LOAD EXPECTED
772 003634 012777 177777 175466      MOV    #-1,@GRDIO          ;ALL ONES TO REGISTER
773 003642 017737 175462 001126      MOV    @GRDIO,$BDDAT       ;READ OUTPUT REG.
774 003650 023737 001124 001126      CMP    $GDDAT,$BDDAT       ;COMPARE
775 003656 001401          BEQ    TST3                 ;;BR IF EQUAL
776 003660 104003          ERROR 3                     ;REG WILL NOT HOLD ONES

```

```

777
778
779
780 003662 000004
781 003664 012737 000040 001166
782 003672 005037 001124
783 003676 012777 177777 175424
784 003704 000005
785 003706 017737 175416 001126
786 003714 001401
787 003716 104003
788
789
790
791
792 003720 000004
793 003722 012737 052525 001124
794 003730 012777 052525 175372
795 003736 017737 175366 001126
796 003744 023737 001124 001126
797 003752 001401
798 003754 104003
799
800
801
802
803 003756 000004
804 003760 012737 125252 001124
805 003766 012777 125252 175334
806 003774 017737 175330 001126
807 004002 023737 001124 001126
808 004010 001401
809 004012 104003
810
811
812
813
814 004014 000004
815 004016 012737 000004 001166
816 004024 012737 004036 001110
817 004032 005037 001124
818 004036 013777 001124 175264
819 004044 017737 175260 001126
820 004052 023737 001124 001126
821 004060 001401
822 004062 104003
823 004064 005237 001124
824 004070 001362

*****
*TEST 3 TEST THAT RESET CLEARS OUTPUT REG.
*****
TST3: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR $GDDAT
MOV #-1,$GRDIO
RESET ;SET DATA TO ALL ONES
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
BEQ TST4 ;;BR IF EQUAL
ERROR 3 ;REG FAILED TO CLEAR

*****
*TEST 4 TEST THAT OUTPUT REG. CAN HOLD #52525
*****
TST4: SCOPE
MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
MOV #52525,$GRDIO
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST5 ;;BR IF EQUAL
ERROR 3 ;DATA NOT=52525

*****
*TEST 5 TEST THAT OUTPUT REG. CAN HOLD #125252
*****
TST5: SCOPE
MOV #125252,$GDDAT ;LOAD EXPECTED VALUE
MOV #125252,$GRDIO
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST6 ;;BR IF EQUAL
ERROR 3 ;DATA NOT=125252

*****
*TEST 6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
*****
TST6: SCOPE
MOV #4,$TIMES ;;DO 4 ITERATIONS
MOV #2,$SLPERR ;LOAD SCOPE ERROR RETURN
CLR $GDDAT ;CLEAR PATTERN
MOV $GDDAT,$GRDIO ;LOAD THE OUTPUT REG.
MOV $GRDIO,$BDDAT ;READ THE REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF EQUAL
ERROR 3 ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
1$: INC $GDDAT ;UPDATE THE PATTERN
BNE 2$ ;TRY AGAIN
2$: MOV $GDDAT,$GRDIO
MOV $GRDIO,$BDDAT
CMP $GDDAT,$BDDAT
BEQ 1$

```

```

825      ;:*****
826      ;*TEST 7      FLOAT A 1 ACROSS THE OUTPUT REGISTER
827      ;:*****
828      004072 000004      †ST7:  SCOPE
829      004074 012737 004110 001110      MOV      #1$, $LPERR      ;LOAD SCOPE ERROR RETURN
830      004102 012737 000001 001124      MOV      #BIT0, $GDDAT    ;LOAD EXPECTED VALUE
831
832      004110 005077 175214      1$:  CLR      @GRDIO      ;CLEAR OUTPUT
833      004114 053777 001124 175206      BIS      $GDDAT, @GRDIO  ;SET THAT BIT
834      004122 017737 175202 001126      MOV      @GRDIO, $BDDAT  ;READ OUTPUT REG.
835      004130 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;TEST RESULTS
836      004136 001401      BEQ      2$              ;BR IF EQUAL ?
837      004140 104003      ERROR    3
838
839      004142 006337 001124      2$:  ASL      $GDDAT      ;SHIFT EXPECTED DATA
840      004146 001360      BNE      1$              ;BR UNTIL DONE
841
842      ;:*****
843      ;*TEST 10     FLOAT A 0 ACROSS THE OUTPUT REGISTER
844      ;:*****
845      004150 000004      †ST10: SCOPE
846      004152 012737 004166 001110      MOV      #1$, $LPERR     ;LOAD SCOPE ERROR RETURN
847      004160 012737 000001 001124      MOV      #BIT0, $GDDAT   ;LOAD EXPECTED VALUE
848
849      004166 012777 177777 175134      1$:  MOV      #-1, @GRDIO   ;LOAD OUTPUT TO A ONE
850      004174 043777 001124 175126      BIC      $GDDAT, @GRDIO  ;CLEAR A BIT
851      004202 017737 175122 001126      MOV      @GRDIO, $BDDAT  ;READ OUTPUT REG.
852      004210 005137 001126      COM      $BDDAT          ;COMPLEMENT IT
853      004214 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;EQUAL ?
854      004222 001401      BEQ      2$              ;BR IF EQUAL
855      004224 104003      ERROR    3
856
857      004226 006337 001124      2$:  ASL      $GDDAT      ;SHIFT LEFT
858      004232 001355      BNE      1$              ;BRANCH UNTIL DONE
859
860      ;:*****
861      ;*TEST 11     TEST FOR SLOW OUTPUT GATES WITH #125252
862      ;:*****
863      004234 000004      †ST11: SCOPE
864      004236 012737 125252 001124      MOV      #125252, $GDDAT ;LOAD EXPECTED VALUE
865      004244 013777 001124 175056      MOV      $GDDAT, @GRDIO  ;LOAD OUTPUT
866      004252 005177 175052      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
867      004256 005177 175046      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
868      004262 005177 175042      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
869      004266 005177 175036      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
870      004272 005177 175032      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
871      004276 005177 175026      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
872      004302 005177 175022      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
873      004306 005177 175016      COM      @GRDIO          ;COMPLEMENT OUTPUT REG.
874      004312 017737 175012 001126      MOV      @GRDIO, $BDDAT  ;READ OUTPUT REG.
875      004320 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;TEST REGISTER
876      004326 001401      BEQ      TST12           ;;BR IF CORRECT
877      004330 104003      ERROR    3

```



```

877
878
879
890 004332 000004
891 004334 012737 052525 001124
892 004342 013777 001124 174760
893 004350 005177 174754
984 004354 005177 174750
895 004360 005177 174744
896 004364 005177 174740
887 004370 005177 174734
898 004374 005177 174730
889 004400 005177 174724
890 004404 005177 174720
891 004410 017737 174714 001126
892 004416 023737 001124 001126
893 004424 001401
894 004426 104003
895
896
897
898
899 004430 000004
900 004432 012737 000400 001166
901 004440 012737 004462 001110
902 004446 012737 052400 001124
903 004454 013777 001124 174646
904 004462 113777 001124 174640 2$:
905 004470 017737 174634 001126
906 004476 023737 001124 001126
907 004504 001401
908 004506 104003
909 004510 105237 001124 1$:
910 004514 001362
911
912
913
914
915 004516 000004
916 004520 012737 000400 001166
917 004526 012737 004550 001110
918 004534 012737 000252 001124
919 004542 013777 001124 174560
920 004550 113777 001125 174554 2$:
921 004556 017737 174546 001126
922 004564 023737 001124 001126
923 004572 001401
924 004574 104003
925 004576 105237 001125 1$:
926 004602 001362

```

```

*****
; *TEST 12 TEST FOR SLOW OUTPUT GATES WITH #52525
*****
TST12: SCOPE
MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
MOV $GDDAT,@GRDIO ;LOAD OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;TEST PATTERN
BEQ TST13 ;:BR IF EQUAL
ERROR 3 ;OUTPUT REGISTER IN ERROR

*****
; *TEST 13 TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
*****
TST13: SCOPE
MOV #400,$TIMES ;:DO 400 ITERATIONS
MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #52400,$GDDAT ;LOAD EXPECTED DATA
MOV $GDDAT,@GRDIO ;LOAD OUTPUT REG.
2$: MOVB $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT
BEQ 1$ ;BRANCH IF EQUAL
ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
1$: INCB $GDDAT ;UPDATE PATTERN
BNE 2$

*****
; *TEST 14 TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
*****
TST14: SCOPE
MOV #400,$TIMES ;:DO 400 ITERATIONS
MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #252,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@GRDIO ;LOAD OUTPUT REG.
2$: MOVB $GDDAT+1,@GRBHIO ;LOAD THE HIGH BYTE
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT
BEQ 1$ ;BRANCH IF EQUAL
ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
1$: INCB $GDDAT+1
BNE 2$

```

```

927
928
929
930 004604 000004
931 004606 005077 174516
932 004612 012777 177777 174506
933 004620 012737 100000 001124
934 004626 012777 100000 174470
935 004634 017737 174464 001126
936 004642 033737 001124 001126
937 004650 001001
938 004652 104001
939
940
941
942
943 004654 000004
944 004656 005077 174442
945 004662 012737 040000 001124
946 004670 012777 040000 174426
947 004676 017737 174422 001126
948 004704 033737 001124 001126
949 004712 001001
950 004714 104001
951
952
953
954
955 004716 000004
956 004720 012737 000200 001124
957 004726 012777 000200 174370
958 004734 017737 174364 001126
959 004742 023737 001124 001126
960 004750 001401
961 004752 104001
962
963
964
965
966 004754 000004
967 004756 005077 174342
968 004762 012737 000100 001124
969 004770 012777 000100 174326
970 004776 017737 174322 001126
971 005004 023737 001124 001126
972 005012 001401
973 005014 104001
974

```

```

*****
;*TEST 15 TEST OUTPUT DATA ACCEPT FLAG
*****
TST15: SCOPE
CLR QGRDIO
MOV #-1,QGRDAI
MOV #BIT15,$GDDAT ;LOAD EXPECTED
MOV #BIT15,QGRSTAT ;SET BIT 15
MOV QGRSTAT,$BDDAT ;READ STATUS
BIT $GDDAT,$BDDAT
BNE TST16 ;;BR IF SET
ERROR 1 ;ERROR, BIT 15 FAILED TO SET

```

```

*****
;*TEST 16 TEST OUTPUT INTERRUPT ENABLE
*****
TST16: SCOPE
CLR QGRSTAT ;CLEAR STATUS
MOV #BIT14,$GDDAT ;LOAD EXPECTED
MOV #BIT14,QGRSTAT ;LOAD BIT 14
MOV QGRSTAT,$BDDAT ;READ STATUS
BIT $GDDAT,$BDDAT
BNE TST17 ;;BR IF SET
ERROR 1 ;ERROR BIT 14 FAILED TO SET

```

```

*****
;*TEST 17 TEST INPUT DATA READY FLAG
*****
TST17: SCOPE
MOV #BIT7,$GDDAT ;LOAD EXPECTED
MOV #BIT7,QGRSTAT ;SET BIT 7
MOV QGRSTAT,$BDDAT ;READ RESULT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST20 ;;BR IF EQUAL
ERROR 1 ;ERROR, BIT 7 FAILED TO SET

```

```

*****
;*TEST 20 TEST INPUT INTERRUPT ENABLE
*****
TST20: SCOPE
CLR QGRSTAT ;CLEAR STATUS
MOV #BIT6,$GDDAT ;LOAD EXPECTED
MOV #BIT6,QGRSTAT ;SET BIT 6
MOV QGRSTAT,$BDDAT ;READ RESULT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST21 ;;BR IF EQUAL
ERROR 1 ;ERROR, BIT 6 FAILED TO SET

```

```

975 ;:*****
976 ;*TEST 21 TEST THAT RESET CLEARS THE DIGITAL STATUS REGISTER
977 ;:*****
978 005016 000004 †ST21: SCOPE
979 005020 012737 000040 001166 MOV #40,$TIMES ;;DO 40 ITERATIONS
980 005026 005077 174272 CLR @GRSTAT ;CLEAR STATUS
981 005032 005037 001124 CLR $GDDAT ;LOAD EXPECTED
982 005036 012777 140300 174260 MOV #BIT15!BIT14!BIT7!BIT6,@GRSTAT
983 005044 000005 RESET
984 005046 017737 174252 001126 MOV @GRSTAT,$BDDAT ;READ RESULT
985 005054 001401 BEQ TST22 ;;BR IF EQUAL
986 005056 104001 ERROR 1 ;ERROR, RESET FAILED TO CLEAR DIGITAL
987 ; STATUS REGISTER
988
989
990 ;:*****
991 ;*TEST 22 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
992 ;:*****
993 005060 000004 †ST22: SCOPE
994 005062 032777 002000 174050 BIT #BIT10,@SWR ;TEST SWITCH BIT
995 005070 001402 BEQ 1$ ;BRANCH IF DOWN
996 005072 000137 007406 JMP DRT21 ;BYPASS SOME TEST USING THE EXTERNAL CABLE
997 005076
998 005076 005077 174226 1$: CLR @GRDIO ;CLEAR OUTPUT REGISTER
999 005102 012777 177777 174216 MOV #-1,@GRDAI ;CLEAR INPUT
1000 005110 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1001 005114 012777 000000 174206 MOV #0,@GRDIO ;LOAD THE OUTPUT
1002 005122 017737 174200 001126 MOV @GRDAI,$BDDAT ;READ THE INPUT
1003 005130 043737 001360 001126 BIC ODDJMP,$BDDAT ;MASK
1004 005136 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1005 005144 001401 BEQ TST23 ;;BR IF EQUAL
1006 005146 104002 ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.
1007
1008 ;:*****
1009 ;*TEST 23 TEST INPUT WITH #-1
1010 ;:*****
1011 005150 000004 †ST23: SCOPE
1012 005152 005077 174152 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1013 005156 012777 177777 174142 MOV #-1,@GRDAI ;CLEAR INPUT
1014 005164 012737 177777 001124 MOV #-1,$GDDAT ;LOAD EXPECTED
1015 005172 043737 001360 001124 BIC ODDJMP,$GDDAT ;MASK
1016 005200 013777 001124 174122 MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
1017 005206 017737 174114 001126 MOV @GRDAI,$BDDAT ;READ INPUT
1018 005214 043737 001360 001126 BIC ODDJMP,$BDDAT ;MASK
1019 005222 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1020 005230 001401 BEQ TST24 ;;BR IF EQUAL
1021 005232 104002 ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
1022

```

```

1023
1024
1025
1026 005234 000004
1027 005236 005077 174066
1028 005242 012777 177777 174056
1029 005250 012737 052525 001124
1030 005256 043737 001360 001124
1031 005264 013777 001124 174036
1032 005272 017737 174030 001126
1033 005300 043737 001360 001126
1034 005306 023737 001124 001126
1035 005314 001401
1036 005316 104002
1037
1038
1039
1040
1041 005320 000004
1042 005322 005077 174002
1043 005326 012777 177777 173772
1044 005334 012737 125252 001124
1045 005342 043737 001360 001124
1046 005350 013777 001124 173752
1047 005356 017737 173744 001126
1048 005364 043737 001360 001126
1049 005372 023737 001124 001126
1050 005400 001401
1051 005402 104002

```

```

*****
;*TEST 24 TEST INPUT WITH #52525
*****
†ST24: SCOPE
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #52525,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST25 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT

```

```

*****
;*TEST 25 TEST INPUT WITH #125252
*****
†ST25: SCOPE
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #125252,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST26 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT

```

N02

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 26
 DZDRGC.P11 T26 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS

```

1052 ;:*****
1053 ;*TEST 26 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
1054 ;:*****
1055 005404 000004 TST26: SCOPE
1056 005406 012737 000001 001166 MOV #1,$TIMES ;;DO 1 ITERATION
1057 005414 005737 001360 TST ODDJMP ;TEST IF ANY ODD JUMPERS
1058 005420 001461 BEQ TST27 ;;BR IF NONE
1059 005422 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD TEST BIT
1060 005430 033737 001360 001124 1$: BIT ODDJMP,$GDDAT ;TEST IF ODD JUMPER BIT
1061 005436 001447 BEQ 2$ ;BR IF NOT
1062 005440 033737 001334 001124 BIT NOTLCH,$GDDAT ;TEST IF LATCHING INPUT BIT
1063 005446 001043 BNE 2$ ;BR IF NOT
1064 005450 013777 001124 173652 MOV $GDDAT,$GRDIO ;LOAD OUTPUT
1065 005456 012777 177777 173642 MOV #-1,$GRDAI ;CLEAR INPUT
1066 005464 043777 001124 173636 BIC $GDDAT,$GRDIO ;CLEAR OUTPUT BIT
1067 005472 017737 173630 001126 MOV $GRDAI,$BDDAT ;READ INPUT REG
1068 005500 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1069 005506 001401 BEQ 3$ ;BR IF EQUAL
1070 005510 104002 ERROR 2 ;ERROR, NEG. INPUT OR NEG. TRANSITION
1071 ; INPUT BIT FAILED TO SET INPUT REGISTER
1072
1073 005512 033737 001124 001362 3$: BIT $GDDAT,$MINSIN ;TEST IF NEG. INPUT BIT
1074 005520 001016 BNE 2$ ;BR IF
1075 005522 012777 177777 173576 MOV #-1,$GRDAI ;CLEAR INPUT
1076 005530 053777 001124 173572 BIS $GDDAT,$GRDIO ;LOAD OUTPUT
1077 005536 017737 173564 001126 MOV $GRDAI,$BDDAT ;READ INPUT
1078 005544 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1079 005552 001401 BEQ 2$ ;BR IF EQUAL
1080 005554 104002 ERROR 2 ;ERROR, POSITIVE INPUT TRANSITION
1081 ; LOGIC FAILED TO SET INPUT REGISTER BIT
1082
1083 005556 006137 001124 2$: ROL $GDDAT ;CHANGE DATA PATTERN
1084 005562 103322 BCC 1$ ;BR IF MORE DATA
  
```



```

*****
*TEST 27      FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
*****
↑ST27:  SCOPE
1095          005564 000004          MOV      #25,$LPERR      ;LOAD ERROR SCOPE RETURN
1096          005566 012737 005614 001110          ;
1097          005574 012737 000001 001354          MOV      #BIT0,BRLEV2   ;LOAD EXPECTED
1098          005602 013737 001334 001356          MOV      NOTLCH,BRLEV3  ;GET NON-LATCH
1099          005610 005137 001356          COM      BRLEV3         ;COMPLEMENT
1100          005614 013737 001354 001124 2$:      MOV      BRLEV2,$GDDAT  ;LOAD GOOD
1101          005622 033737 001124 001350          BIT      $GDDAT,ODDJMP  ;TEST IF ODD JUMPER
1102          005630 001042          BNE     1$             ;BYPASS IF ODD JUMPER
1103          005632 033737 001124 001334          BIT      $GDDAT,NOTLCH ;TEST FOR NON-LATCH
1104          005640 001436          BEQ     1$             ;BR IF LATCHING
1105          005642 013777 001124 173460          MOV      $GDDAT,$GRDIO  ;LOAD OUTPUT
1106          005650 017737 173452 001126          MOV      $GRDAI,$BDDAT ;READ INPUT
1107          005656 043737 001356 001126          BIC     BRLEV3,$BDDAT  ;MASK TO LATCH BITS
1108          005664 023737 001124 001126          CMP     $GDDAT,$BDDAT  ;COMPARE
1109          005672 001401          BEQ     3$             ;:BR IF EQUAL
1110          005674 104002          ERROR   2             ;INPUT REGISTER IN ERROR
1111                                     ;SJB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
1112                                     ;:BR IF EQUAL
1113                                     ;INPUT REGISTER IN ERROR
1114                                     ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1115          005676 043777 001124 173424 3$:      BIC     $GDDAT,$GRDIO  ;CLEAR OUTPUT BIT
1116          005704 017737 173416 001126          MOV      $GRDAI,$BDDAT ;READ INPUT
1117          005712 043737 001356 001126          BIC     BRLEV3,$BDDAT  ;MASK TO LATCH BITS
1118          005720 005037 001124          CLR     $GDDAT         ;CLEAR EXPECTED
1119          005724 033737 001354 001126          BIT      BRLEV2,$BDDAT ;TEST ,OR BIT
1120          005722 001401          BEQ     1$             ;:BR IF CLEARED
1121          005734 104002          ERROR   2             ;INPUT BIT LATCHED IN ERROR
1122                                     ;WAS CORRECT LATCH·NON-LATCH SUPPLIED ?
1123          005736 006337 001354          1$:      ASL     BRLEV2         ;CHANGE PATTERN
1124          005742 001324          SNE     2$             ;BR UNTIL DONE

```



```

1175 ::*****
1176 :*TEST 32 TEST FOR SLOW INPUT GATES WITH #125252
1177 :*****
1178 †ST32: SCOPE
1179
1180 006222 000004
1180 006224 012737 125252 00:124 MOV #125252,$GDDAT ;LOAD EXPECTED
1181 006232 043737 001334 001124 BIC NOTLCH,$GDDAT ;CONVERT
1182 006240 043737 001360 001124 BIC ODDJMP,$GDDAT ;MASK
1183 006246 013700 001124 MOV $GDDAT,R0 ;LOAD PATTERN
1184 006252 005077 173052 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1185 006256 012777 177777 173042 MOV #-1,JGRDAI ;CLEAR INPUT
1186
1187 006264 010077 173040 MOV R0,JGRDIO ;LOAD OUTPUT
1188 006270 005077 173034 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1189 006274 017701 173026 MOV JGRDAI,R1 ;READ INPUT
1190 006300 050177 173022 BIS R1,JGRDAI ;CLEAR INPUT
1191 006304 005100 COM R0
1192 006306 010077 173016 MOV R0,JGRDIO ;LOAD OUTPUT
1193 006312 005077 173012 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1194 006316 017701 173004 MOV JGRDAI,R1 ;READ INPUT
1195 006322 050177 173000 BIS R1,JGRDAI ;CLEAR INPUT
1196 006326 005100 COM R0
1197 006330 010077 172774 MOV R0,JGRDIO ;LOAD OUTPUT
1199 006334 005077 172770 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1199 006340 017701 172762 MOV JGRDAI,R1 ;READ INPUT
1200 006344 050177 172756 BIS R1,JGRDAI ;CLEAR INPUT
1201 006350 005100 COM R0
1202 006352 010077 172752 MOV R0,JGRDIO ;LOAD OUTPUT
1203 006356 005077 172746 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1204 006362 017701 172740 MOV JGRDAI,R1 ;READ INPUT
1205 006366 050177 172734 BIS R1,JGRDAI ;CLEAR INPUT
1206 006372 005100 COM R0
1207 006374 010077 172730 MOV R0,JGRDIO ;LOAD OUTPUT
1208 006400 005077 172724 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1209 006404 017701 172716 MOV JGRDAI,R1 ;READ INPUT
1210 006410 050177 172712 BIS R1,JGRDAI ;CLEAR INPUT
1211 006414 005100 COM R0
1212 006416 010077 172706 MOV R0,JGRDIO ;LOAD OUTPUT
1213 006422 005077 172702 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1214 006426 017701 172674 MOV JGRDAI,R1 ;READ INPUT
1215 006432 050177 172670 BIS R1,JGRDAI ;CLEAR INPUT
1216 006436 005100 COM R0
1217 006440 010077 172664 MOV R0,JGRDIO ;LOAD OUTPUT
1218 006444 005077 172660 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1219 006450 017701 172652 MOV JGRDAI,R1 ;READ INPUT
1220 006454 050177 172646 BIS R1,JGRDAI ;CLEAR INPUT
1221 006460 005100 COM R0
1222 006462 010077 172642 MOV R0,JGRDIO ;LOAD OUTPUT
1223 006466 005077 172636 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1224 006472 017701 172630 MOV JGRDAI,R1 ;READ INPUT
1225 006476 050177 172624 BIS R1,JGRDAI ;CLEAR INPUT
1226 006502 005100 COM R0
1227 006504 010077 172620 MOV R0,JGRDIO ;LOAD OUTPUT
1228 006510 005077 172614 CLR JGRDIO ;CLEAR OUTPUT REGISTER
1229 006514 017701 172606 MOV JGRDAI,R1 ;READ INPUT
1230 006520 050177 172602 BIS R1,JGRDAI ;CLEAR INPUT

```

```

1231 006524 005100          CUM      RO
1232 006526 010077 172576  MOV      RO, @GRDIO      ;LOAD OUTPUT
1233 006532 005077 172572  CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1234 006536 017701 172564  MOV      @GRDAI, R1      ;READ INPUT
1235 006542 050177 172560  BIS      R1, @GRDAI      ;CLEAR INPUT
1236 006546 005100          COM      RO
1237 006550 010077 172554  MOV      RO, @GRDIO      ;LOAD OUTPUT
1238 006554 005077 172550  CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1239 006560 017701 172542  MOV      @GRDAI, R1      ;READ INPUT
1240 006564 050177 172536  BIS      R1, @GRDAI      ;CLEAR INPUT
1241 006570 005100          COM      RO
1242
1243 006572 010137 001126  MOV      R1, $BDDAT      ;LOAD READ
1244 006576 000240          NOP
1245 006600 000240          NOP
1246 006602 000240          NOP
1247 006604 023737 001124 001126  CMP      $GDDAT, $BDDAT  ;COMPARE
1248 006612 001401          BEQ      TST33           ;;BR IF EQUAL
1249 006614 104002          ERROR      2           ;INPUT GATE SLOW
1250

```

```

*****
;:TEST 33      TEST FOR SLOW INPUT GATES WITH #52525
*****
TST33: SCOPE

```

```

1256 006620 012737 052525 001124  MOV      #52525, $GDDAT  ;SETUP EXPECTED
1257 006626 043737 001360 001124  BIC      ODDJMP, $GDDAT  ;MASK ODD JUMPER BITS
1258 006634 043737 001334 001124  BIC      NOTLCH, $GDDAT  ;CONVERT
1259 006642 013700 001124          MOV      $GDDAT, RO
1260 006646 005077 172456          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1261 006652 012777 177777 172446  MOV      #-1, @GRDAI     ;CLEAR INPUT
1262
1263 006660 010077 172444          MOV      RO, @GRDIO      ;LOAD OUTPUT
1264 006664 005077 172440          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1265 006670 017701 172432          MOV      @GRDAI, R1      ;READ INPUT
1266 006674 050177 172426          BIS      R1, @GRDAI      ;CLEAR INPUT
1267 006700 005100          COM      RO
1268 006702 010077 172422          MOV      RO, @GRDIO      ;LOAD OUTPUT
1269 006706 005077 172415          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1270 006712 017701 172410          MOV      @GRDAI, R1      ;READ INPUT
1271 006716 050177 172404          BIS      R1, @GRDAI      ;CLEAR INPUT
1272 006722 005100          COM      RO
1273 006724 010077 172400          MOV      RO, @GRDIO      ;LOAD OUTPUT
1274 006730 005077 172374          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1275 006734 017701 172366          MOV      @GRDAI, R1      ;READ INPUT
1276 006740 050177 172362          BIS      R1, @GRDAI      ;CLEAR INPUT
1277 006744 005100          COM      RO
1278 006746 010077 172356          MOV      RO, @GRDIO      ;LOAD OUTPUT
1279 006752 005077 172352          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1280 006756 017701 172344          MOV      @GRDAI, R1      ;READ INPUT
1281 006762 050177 172340          BIS      R1, @GRDAI      ;CLEAR INPUT
1282 006766 005100          COM      RO
1283 006770 010077 172334          MOV      RO, @GRDIO      ;LOAD OUTPUT
1284 006774 005077 172330          CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1285 007000 017701 172322          MOV      @GRDAI, R1      ;READ INPUT
1286 007004 050177 172316          BIS      R1, @GRDAI      ;CLEAR INPUT

```

1297	007010	005100		CUM	RO	
1298	007012	010077	172312	MOV	RO, @GRDIO	; LOAD OUTPUT
1299	007016	005077	172306	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1290	007022	017701	172300	MOV	@GRDAI, R1	; READ INPUT
1291	007026	050177	172274	BIS	R1, @GRDAI	; CLEAR INPUT
1292	007032	005100		COM	RO	
1293	007034	010077	172270	MOV	RO, @GRDIO	; LOAD OUTPUT
1294	007040	005077	172264	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1295	007044	017701	172256	MOV	@GRDAI, R1	; READ INPUT
1296	007050	050177	172252	BIS	R1, @GRDAI	; CLEAR INPUT
1297	007054	005100		COM	RO	
1298	007056	010077	172246	MOV	RO, @GRDIO	; LOAD OUTPUT
1299	007062	005077	172242	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1300	007066	017701	172234	MOV	@GRDAI, R1	; READ INPUT
1301	007072	050177	172230	BIS	R1, @GRDAI	; CLEAR INPUT
1302	007076	005100		COM	RO	
1303	007100	010077	172224	MOV	RO, @GRDIO	; LOAD OUTPUT
1304	007104	005077	172220	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1305	007110	017701	172212	MOV	@GRDAI, R1	; READ INPUT
1306	007114	050177	172206	BIS	R1, @GRDAI	; CLEAR INPUT
1307	007120	005100		COM	RO	
1308	007122	010077	172202	MOV	RO, @GRDIO	; LOAD OUTPUT
1309	007126	005077	172176	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1310	007132	017701	172170	MOV	@GRDAI, R1	; READ INPUT
1311	007136	050177	172164	BIS	R1, @GRDAI	; CLEAR INPUT
1312	007142	005100		COM	RO	
1313	007144	010077	172160	MOV	RO, @GRDIO	; LOAD OUTPUT
1314	007150	005077	172154	CLR	@GRDIO	; CLEAR OUTPUT REGISTER
1315	007154	017701	172146	MOV	@GRDAI, R1	; READ INPUT
1316	007160	050177	172142	BIS	R1, @GRDAI	; CLEAR INPUT
1317	007164	005100		COM	RO	
1318						
1319	007166	010137	001126	MOV	R1, \$BDDAT	; LOAD VALUE READ
1320	007172	000240		NOP		
1321	007174	000240		NOP		
1322	007176	000240		NOP		
1323	007200	023737	001124 001126	CMP	\$GDDAT, \$BDDAT	; COMPARE
1324	007206	001401		BEQ	TST34	::BR IF EQUAL
1325	007210	104002		ERROR	2	


```

1326
1327
1328
1329 007212 000004
1330 007214 012737 000040 001166
1331 007222 005077 172102
1332 007226 012777 177777 172072
1333 007234 012777 177777 172066
1334 007242 005037 001124
1335 007246 000005
1336 007250 017737 172052 001126
1337 007256 043737 001360 001126
1338 007264 001401
1339 007266 104002
1340
1341
1342
1343
1344 007270 000004
1345 007272 012737 000200 001124
1346 007300 005077 172020
1347 007304 012777 000000 172016
1348 007312 022727 000000 000000
1349 007320 017737 172000 001126
1350 007326 023737 001124 001126
1351 007334 001401
1352 007336 104001
1353
1354
1355
1356
1357
1358
1359 007340 000004
1360 007342 012737 100000 001124
1361 007350 005077 171750
1362 007354 012777 000000 171746
1363 007362 022727 000000 000000
1364 007370 017700 171732
1365 007374 017737 171724 001126
1366 007402 100401
1367 007404 104001
1368

```

```

*****
*TEST 34 TEST THAT RESET CLEARS INPUT REGISTER BITS
*****
†ST34: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #-1,@GRDIO ;LOAD OUTPUT
CLR $GDDAT ;LOAD EXPECTED
RESET
MOV @GRDAI,$BDDAT ;READ INPUT REG.
BIC ODDJMP,$BDDAT ;MASK ODD JUMPERS
BEQ TST35 ;;BR IF ALL BITS CLEARED
ERROR 2 ;INPUT REG. FAILED TO CLEAR UPON RESET INST.

```

```

*****
*TEST 35 TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
*****
†ST35: SCOPE
MOV #BIT7,$GDDAT ;LOAD EXPECTED
CLR @GRSTAT
MOV #0,@GRDIO ;OUTPUT 0
CMP #0,#0 ;DELAY
MOV @GRSTAT,$BDDAT ;READ RESULTS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST36 ;;BR IF EQUAL
ERROR 1 ;INPUT DATA READY FLAG FAILED TO SET

```

;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET

```

*****
*TEST 36 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
*****
†ST36: SCOPE
MOV #BIT15,$GDDAT ;LOAD EXPECTED
CLR @GRSTAT
MOV #0,@GRDIO
CMP #0,#0
MOV @GRDAI,R0 ;READ INPUT
MOV @GRSTAT,$BDDAT ;READ RESULTS
BMI TST37 ;;BR IF SET
ERROR 1 ;INPUT DATA READY FLAG FAILED TO SET

```

H03

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 33
DZDRGC.P11 T36 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET

```
1369                                     :TEST THAT THE DIGITAL I/O DOES NOT INTERRUPT
1370                                     :SET INPUT-OUTPUT FLAGS BUT DO NOT ENABLE INTERRUPTS
1371
1372 007406
1373
1374
1375
1376 007406 000004
1377 007410 012737 000040 001166
1378 007416 000005
1379 007420 005077 171700
1380 007424 005037 177776
1381 007430 012777 007510 171702
1382 007436 012777 000340 171676
1383 007444 012777 007514 171672
1384 007452 012777 000340 171666
1385 007460 012777 000000 171642
1386 007466 017700 171634
1387
1388 007474 000240
1389 007476 000240
1390 007500 000240
1391 007502 005077 171616
1392 007506 000404
1393
1394 007510 104006 15: ERROR 6 ;ERROR, DIGITAL INPUT INTERRUPTED
1395 007512 000002 RTI
1396
1397 007514 104006 25: ERROR 6 ;ERROR, DIGITAL OUTPUT INTERRUPTED
1398 007516 000002 RTI

DRT21:
:*****
;*TEST 37 TEST FOR UNEXPECTED INTERRUPT
:*****
TST37: SCOPE
MOV #40,STIMES ;;DO 40 ITERATIONS
RESET
CLR @GRSTAT
CLR PSW
MOV #15,@GRIVA ;SET UP INTERRUPT INPUT VECTOR
MOV #340,@GRIVSA
MOV #25,@GRIVB ;SET UP INTERRUPT OUTPUT VECTOR
MOV #340,@GRIVSB
MOV #0,@GRDIO ;OUTPUT
MOV @GRDAI,RO ;INPUT
NOP
NOP
NOP
NOP
CLR @GRSTAT ;CLEAR STATUS
BR TST40 ::

15: ERROR 6 ;ERROR, DIGITAL INPUT INTERRUPTED
RTI

25: ERROR 6 ;ERROR, DIGITAL OUTPUT INTERRUPTED
RTI
```

```

1399          ;:*****
1400          ;*TEST 40      TEST THAT THE INPUT CAN INTR. USING THE MAINT. BIT
1401          ;:*****
1402 007520 000004          TST40: SCOPE
1403 007522 000240          NOP
1404 007524 000240          NOP
1405 007526 005077 171572  CLR      @GRSTAT          ;CLEAR STATUS
1406 007532 012777 007574 171600  MOV      #15,@GRIVA        ;LOAD RETURN VECTOR
1407 007540 012777 000340 171574  MOV      #340,@GRIVSA     ;
1408 007546 013777 001346 171570  MOV      GRIVSB,@GRIVB    ;LOAD OUTPUT VECTOR
1409 007554 012777 000000 171564  MOV      #0,@GRIVSB
1410 007562 052777 000040 171534  BIS      #BITS,@GRSTAT   ;MAINT. INT C
1411 007570 000240          NOP
1412 007572 104004          ERROR      4          ;ERROR, INPUT FAILED TO INTERRUPT
1413
1414          ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN NO INTERRUPT WILL OCCUR
1415          ; IF INTERRUPT OCCURS 'INT DONE C' FAILED TO CLEAR INT C FLOP
1416
1417 007574 012777 007616 171536 1$:  MOV      #25,@GRIVA        ;LOAD INPUT VECTOR
1418 007602 022626          CMP      (SP)+,(SP)+      ;POP THE STACK #4
1419 007604 005037 177776          CLR      PSW             ;LOWER PRIOR.
1420 007610 000240          NOP
1421 007612 000240          NOP
1422 007614 000404          BR       TST41          ;;<NEXT TEST>
1423
1424 007616 022626          2$:  CMP      (SP)+,(SP)+      ;POP THE STACK
1425 007620 104006          ERROR      6          ;UNEXPECTED INTERRUPT
1426 007622 005037 177776          CLR      PSW

```

J03

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 35
 DZDRGC.P11 T41 TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT

```

1427 ;*****
1428 ;*TEST 41 TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT
1429 ;*****
1430 007626 000004 TST41: SCOPE
1431 007630 000240 NOP
1432 007632 000240 NOP
1433 007634 005077 171464 CLR QGRSTAT ;CLEAR STATUS
1434 007640 012777 007710 171472 MOV #15,QGRIVA ;LOAD RETURN VECTOR
1435 007646 012777 000340 171466 MOV #340,QGRIVSA ;
1436 007654 013777 001346 171462 MOV GRIVSB,QGRIVB ;LOAD OUTPUT VECTOR
1437 007662 012777 000000 171456 MOV #0,QGRIVSB ;
1438 007670 012777 000100 171426 MOV #BIT6,QGRSTAT ;LOAD INPUT INTR. ENABLE
1439 007676 052777 000040 171420 SIS #BITS,QGRSTAT ;MAINT. INT C
1440 007704 000240 NOP
1441 007706 104004 ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
1442
1443 007710 012777 007752 171422 1$: MOV #25,QGRIVA ;LOAD INPUT VECTOR
1444 007716 022626 CMP (SP)+,(SP)+ ;POP THE STACK *4
1445 007720 005037 177776 CLR PSW ;LOWER PRIOR.
1446 007724 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1447 007730 017737 171370 001126 MOV QGRSTAT,$BDDAT ;READ STATUS
1448 007736 032737 000100 001126 BIT #BIT6,$BDDAT ;TEST BIT 6
1449 007744 001406 BEQ TST42 ;;BR IF CLEARED
1450 007746 104001 ERROR 1 ;INPUT INTR. FAILED TO CLEAR
1451 007750 000404 BR TST42 ;;<NEXT TEST>
1452
1453 007752 022626 2$: CMP (SP)+,(SP)+ ;POP THE STACK
1454 007754 104006 ERROR 6 ;UNEXPECTED INTERRUPT
1455 007756 005037 177776 CLR PSW
  
```

K03

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 36
DZDRGC.P11 T42 TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT

```

1456 ;:*****
1457 ;*TEST 42 TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT
1458 ;:*****
1459 007762 000004 TST42: SCOPE
1460 007764 000240 NOP
1461 007766 005077 171332 CLR QGRSTAT ;CLEAR STATUS
1462 007772 013777 001342 171340 MOV GRIVSA,QGRIVA ;LOAD INPUT VECTOR
1463 010000 012777 000000 171334 MOV #0,QGRIVSA
1464 010006 012777 010034 171330 MOV #1,QGRIVB ;LOAD OUTPUT VECTOR
1465 010014 012777 000340 171324 MOV #340,QGRIVSB
1466 010022 052777 G20000 171274 BIS #BIT13,QGRSTAT ;MAINT. INTERRUPT
1467 010030 000240 NOP
1468 010032 104005 ERROR 5 ;OUTPUT FAILED TO INTERRUPT
1469
1470 ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN, NO INTERRUPT WILL OCCUR
1471 ; IF IT DOES, 'INT DONE D HIGH' FAILED TO CLEAR 'INT D' FLOP
1472
1473 010034 012777 010056 171302 1$: MOV #2$,QGRIVB ;LOAD OUTPUT VECTOR
1474 010042 022626 CMP (SP)+,(SP)+ ;POP THE STACK *4
1475 010044 005037 177776 CLR PSW
1476 010050 000240 NOP
1477 010052 000240 NOP
1478 010054 000404 BR TST43 ;;<NEXT TEST>
1479 010056 022626 2$: CMP (SP)+,(SP)+ ;POP THE STACK
1480 010060 104006 ERROR 6 ;UNEXPECTED INTERRUPT
1481 010062 005037 177776 CLR PSW
1482

```



```

1483
1484
1485
1486 010066 000004
1487 010070 000240
1488 010072 000240
1489 010074 032777 002000 171036
1490 010102 001401
1491 010104 000455
1492 010106 005077 171212
1493 010112 012777 010202 171220
1494 010120 012777 000340 171214
1495 010126 013777 001346 171210
1496 010134 005077 171206
1497 010140 012777 000100 171156
1498 010146 012777 000000 171154
1499 010154 017700 171146
1500 010160 000240
1501 010162 000240
1502 010164 000240
1503 010166 042777 000100 171130
1504 010174 000240
1505 010176 104004
1506 010200 000417
1507 010202 022626
1508 010204 005037 177776
1509 010210 005077 171110
1510 010214 013777 001342 171116
1511 010222 005077 171114
1512 010226 013777 001346 171110
1513 010234 005077 171106
1514

```

```

*****
*TEST 43 TEST FOR INTR. FROM DRA INPUT
*****
TST43: SCOPE
NOP
NOP
BIT #BIT10,@SWR ;TEST SWITCH
BEQ 1$
BR TST44 ;;
1$: CLR @GRSTAT ;
MOV #25,@GRIVA ;IN CASE OF INTERRUPT
MOV #340,@GRIVSA
MOV GRIVSB,@GRIVB
CLR @GRIVSB
MOV #BIT6,@GRSTAT ;ENABLE INPUT INTR.
MOV #0,@GRDIO ;OUTPUT
MOV @GRDAI,R0 ;INPUT
NOP ;LET INTERRUPT OCCUR
NOP
NOP
BIC #BIT6,@GRSTAT
NOP
ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
BR TST44 ;;
2$: CMP (SP)+,(SP)+
CLR PSW
CLR @GRSTAT ;CLEAR STATUS
MOV GRIVSA,@GRIVA ;RESET INPUT VECTOR
CLR @GRIVSA
MOV GRIVSB,@GRIVB ;RESET OUTPUT VECTOR
CLR @GRIVSB

```

M03

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 38
 DZDRGC.P11 T44 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG

```

1515 ::*****
1516 ;*TEST 44 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
1517 ::*****
1518 010240 000004 TST44: SCOPE
1519 010242 032777 002000 170670 BIT #BIT10,@SWR ;TEST CABLE SWITCH
1520 010250 001067 BNE TST45 ;:BYPASS IF NO I/O CABLE
1521
1522 010252 012737 010266 001110 MOV #15,$LPERR ;LOAD ERROR SCOPE RETURN
1523 010260 012737 000001 001356 MOV #BIT0,BRLEV3 ;LOAD INTERRUPT BIT
1524 010266 005037 001126 1$: CLR $BDDAT ;CLEAR BAD DATA
1525 010272 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD GOOD DATA
1526
1527 010300 033737 001356 001336 BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT
1528 010306 001445 BEQ 3$ ;;NO TRY NEXT BIT
1529 010310 005077 171014 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1530 010314 012777 177777 171004 MOV #-1,@GRDAI ;CLEAR INPUT
1531 010322 005077 170776 CLR @GRSTAT ;CLEAR STATUS
1532 010326 053777 001356 170774 BIS BRLEV3,@GRDIO ;LOAD OUTPUT
1533 010334 043777 001356 170766 BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
1534 010342 053777 001356 170760 BIS BRLEV3,@GRDIO ;LOAD OUTPUT
1535 010350 005077 170750 CLR @GRSTAT ;CLEAR FLAG FROM DATA READY
1536 ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1537 ;IF INTERRUPT INPUT SWITCH IS ON
1538 010354 105777 170744 TSTB @GRSTAT ;TEST READY BIT
1539 010360 100401 BMI 2$ ;;BR IF SET
1540 010362 104010 ERROR 10 ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
1541 ;?? DID OPERATOR GIVE CORRECT
1542 ;INPUT INTERRUPT BITS ??
1543
1544 010364 043777 001356 170736 2$: BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
1545 010372 053777 001356 170726 BIS BRLEV3,@GRDAI ;CLEAR INPUT
1546 010400 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1547 010404 005077 170714 CLR @GRSTAT ;CLEAR STATUS
1548 010410 117737 170710 001126 MOVB @GRSTAT,$BDDAT ;READ STATUS
1549 010416 100001 BPL 3$ ;;BR IF CLEARED
1550 010420 104001 ERROR 1 ;INPUT READY FAILED TO CLEAR
1551
1552 010422 006337 001356 3$: ASL BRLEV3 ;CHANGE BIT
1553 010426 001317 BNE 1$ ;BR IF NOT DONE

```

N03

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 39
DZDRGC.P11 T45 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG

```

1554 ::*****
1555 :*TEST 45 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
1556 :*****
1557 010430 000004 TST45: SCOPE
1558 010432 032777 002000 170500 BIT #BIT10, QSWR ;TEST CABLE SWITCH
1559 010440 001050 BNE TST46 ;;BYPASS IF NO I/C CABLE
1560
1561 010442 012737 010456 001110 MOV #15, $LPERR ;LOAD ERROR SCOPE RETURN
1562 010450 012737 000001 001356 MOV #BIT0, BRLEV3 ;LOAD NON-INTERRUPT BIT
1563 010456 012737 000200 001126 1$: MOV #200, $EJDAT ;LOAD BAD DATA
1564 010464 005037 001124 CLR $GDDAT ;CLEAR GOOD DATA
1565
1566 010470 033737 001356 001336 BIT BRLEV3, INTBIT ;TEST IF THIS BIT WILL INTERRUPT
1567 010476 001026 BNE 3$ ;;YES SKIP AND TRY NEXT BIT
1568 010500 005077 170624 CLR QGRDIO ;CLEAR OUTPUT REGISTER
1569 010504 012777 177777 170614 MOV #-1, QGRDAI ;CLEAR INPUT
1570 010512 005077 170606 CLR QGRSTAT ;CLEAR STATUS
1571 010516 053777 001356 170604 BIS BRLEV3, QGRDIO ;LOAD OUTPUT
1572 010524 043777 001356 170576 BIC BRLEV3, QGRDIO ;CLEAR OUTPUT
1573 010532 053777 001356 170570 BIS BRLEV3, QGRDIO ;SET OUTPUT
1574 010540 005077 170560 CLR QGRSTAT ;CLEAR FLAG FROM DATA READY
1575 ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1576 010544 105777 170554 TSTB QGRSTAT ;TEST READY BIT
1577 010550 100001 BPL 3$ ;;BR IF CLEAR
1578 010552 104011 ERROR 11 ;INPUT NON-INTERRUPT BIT SET INPUT READY
1579 ;?? DID OPERATOR GIVE CORRECT
1580 ;INPUT INTERRUPT BITS ??
1581
1582
1583 010554 006337 001356 3$: ASL BRLEV3 ;CHANGE BIT
1584 010560 001336 BNE 1$ ;BR IF NOT DONE

```

```

1505
1506
1507
1508 010552 000004
1509 010554 000240
1510 010556 000240
1511 010570 032777 002000 170342
1512 010576 001401
1513 010600 000455
1514 010602 005077 170516
1515 010606 012777 010676 170530
1516 010614 012777 000340 170524
1517 010622 013777 001342 170510
1518 010630 005077 170506
1519 010634 012777 040000 170462
1520 010642 012777 000000 170450
1521 010650 017700 170452
1522 010654 000240
1523 010656 000240
1524 010660 000240
1525 010662 042777 040000 170434
1526 010670 000240
1527 010672 104007
1528 010674 000417
1529 010676 022626
1610 010700 013777 001342 170432
1611 010706 005077 170430
1612 010712 013777 001346 170424
1613 010720 005077 170422
1614 010724 005037 177776
1615 010732 005077 170370
1616
1617
1618
1619
1620 010734 000004
1621 010736 012737 000001 001166
1622 010744 000005
1623 010746 042737 177437 001350
1624 010754 001001
1625 010756 104007
1626 010760 022737 000340 001350
1627 010766 001001
1628 010770 104007
1629 010772 013737 001350 001352
1630 011000 162737 000040 001352
1631 011006 013737 001350 001354
1632 011014 013737 001350 001356
1633 011022 062737 000040 001356

```

```

*****
*TEST 46 TEST FOR INTR. FROM DRA OUTPUT
*****
TST46: SCOPE
NOP
NOP
9IT #BIT10,DSWR ;TEST SWITCH
BEQ 15
BR TST47 ;;
15: CLR JGRSTAT
MOV #25,JGRIVB ;IN CASE OF INTERRUPT
MOV #340,JGRIVSB
MOV GRIVSA,JGRIVA
CLR JGRIVSA
MOV #BIT14,JGRSTAT ;ENABLE OUTPUT INTR.
MOV #0,JGRDIO ;OUTPUT
MOV JGRDAI,RO ;INPUT
NOP ;LET INTERRUPT OCCUR
NOP
NOP
BIC #BIT14,JGRSTAT
NOP
ERROR 5 ;ERROR, OUTPUT FAILED TO INTERRUPT
BR TST47 ;;
25: CMP (SP)+,(SP)+
MOV GRIVSA,JGRIVA ;RESET INPUT VECTOR
CLR JGRIVSA
MOV GRIVSB,JGRIVB ;RESET OUTPUT VECTOR
CLR JGRIVSB
CLR PSW
CLR JGRSTAT ;CLEAR STATUS
*****
*TEST 47 PRE-INTERRUPT SETUP
*****
TST47: SCOPE
MOV #1,STIMES ;;DO 1 ITERATION
RESET
BIC #177437,DIOBRL
BE 15
ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 0
15: CMP #340,DIOBRL
BNE 25
ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
25: MOV DIOBRL,BRLEV1
SUB #40,BRLEV1
MOV DIOBRL,BRLEV2
MOV DIOBRL,BRLEV3
ADD #40,BRLEV3

```

```

1634
1635
1636
1637 011030 000004
1638 011032 005077 170266
1639 011036 012777 011072 170274
1640 011044 013737 001352 177776
1641 011052 052777 000040 170244
1642 011060 000240
1643 011062 000240
1644 011064 000240
1645 011066 000240
1646 011070 104004
1647 011072 022626
1648 011074 013777 001342 170236
1649 011102 005077 170234
1650 011106 013777 001346 170230
1651 011114 005077 170226
1652 011120 005037 177776
1653
1654
1655
1656 011124 000004
1657 011126 005077 170172
1658 011132 012777 011204 170200
1659 011140 013737 001354 177776
1660 011146 052777 000040 170150
1661 011154 000240
1662 011156 000240
1663 011160 000240
1664
1665 011162 012777 011216 170150
1666 011170 005037 177776
1667 011174 000240
1668 011176 000240
1669 011200 000240
1670 011202 000403
1671 011204 022626
1672 011206 005037 177776
1673 011212 104006
1674 011214 000415
1675 011216 022626
1676 011220 013777 001342 170112
1677 011226 005077 170110
1678 011232 013777 001346 170104
1679 011240 005077 170102
1680 011244 005037 177776

```

```

*****
*TEST 50 TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
*****
TST50: SCOPE
CLR @GRSTAT ;CLEAR ENABLES
MOV #15,@GRIVA ;SET UP VECTORS
MOV BRLEV1,PSW ;CHANGE PSW
BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
NOP
NOP
NOP
NOP
ERROR 4 ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
15: CMP (SP)+,(SP)+
MOV GRIVSA,@GRIVA ;RESET INPUT VECTOR
CLR @GRIVSA
MOV GRIVSB,@GRIVB ;RESET OUTPUT VECTOR
CLR @GRIVSB
CLR PSW ;LOWER PSW
*****
*TEST 51 TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED VIA MAINT. INT
*****
TST51: SCOPE
CLR @GRSTAT ;CLEAR ENABLES
MOV #15,@GRIVA ;SET UP VECTORS
MOV BRLEV2,PSW ;CHANGE PSW
BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
NOP
NOP
NOP
SUB-TEST NOW LOWER THE PSW AND ALLOW THE INTERRUPT
MOV #25,@GRIVA ;RESET VECTOR
CLR PSW ;LOWER PSW
NOP
NOP
NOP
BR 35 ;ERROR
15: CMP (SP)+,(SP)+
CLR PSW
35: ERROR 6 ;UNEXPECTED INTERRUPT
BR TST52
25: CMP (SP)+,(SP)+ ;POP THE STACK
MOV GRIVSA,@GRIVA ;RESET INPUT VECTOR
CLR @GRIVSA
MOV GRIVSB,@GRIVB ;RESET OUTPUT VECTOR
CLR @GRIVSB
CLR PSW

```

```

1681
1682
1683
1684 011250 000004
1685 011252 012737 000001 001166
1686 011260 005737 001314
1687 011264 001415
1688 011266 032777 010000 167644
1689 011274 001024
1690 011276 162737 000010 001316
1691 011304 062737 000010 001320
1692 011312 005337 001314
1693 011316 000413
1694 011320 013737 001306 001316
1695 011326 013737 001310 001320
1696 011334 013737 001312 001314
1697 011342 000137 011356
1698 011346 012700 177777
1699 011352 000137 003342
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710 011356
1711 011356 000004
1712 011360 005037 001102
1713 011364 005037 001166
1714 011370 005237 001100
1715 011374 042737 100000 001100
1716 011402 005327
1717 011404 000001
1718 011406 003022
1719 011410 012737
1720 011412 000001
1721 011414 011404
1722 011416 104401 011463
1723 011422 013746 001100
1724 011426 104405
1725 011430 104401 011460
1726 011434 013700 000042
1727 011440 001405
1728 011442 000005
1729 011444 004710
1730 011446 000240
1731 011450 000240
1732 011452 000240
1733 011454
1734 011454 000137
1735 011456 011346
1736 011460 377 377 000

;*****
;TEST 52 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
;*****
↑ST52: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
BYPASS: TST NBEXT ;TEST IF ANY
SEQ IS ;BR IF NONE
BIT #SW12,$SWR ;TEST BIT12 OF SWR
BNE BYPAS1 ;INHIBIT TESTING NEXT DR11-K
SUB #10,$DRADD ;UPDATE DEVICE ADDRESS
ADD #10,$DRIV ;UPDATE DEVICE VECTOR
DEC NBEXT ;ANOTHER ONE ?
SR BYPAS1 ;BR IF ANOTHER
IS: MOV BASEBA,$DRADD ;RELOAD ADDRESS
MOV BASEIV,$DRIV ;RELOAD VECTOR
MOV NMBEXT,$NBEXT ;RELOAD NUMBER
JMP $EOP ;DONE
BYPAS1: MOV #-1,$RO
JMP RBEG2 ;TEST ANOTHER UNIT

.SBTTL END OF PASS ROUTINE

;*****
;INCREMENT THE PASS NUMBER ($PASS)
;INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
;IF THERES A MONITOR GO TO IT
;IF THERE ISN'T JUMP TO BYPAS1

$EOP:
SCOPE
CLR $STNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMEER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,$(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
TYPE $ENDMG ;;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 $42,$RO ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC,($RO) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
$DOAGN: JMP $(PC)+ ;;RETURN
$RTNAD: .WORD BYPAS1
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING

```

```

1737 011463 015 042412 042116
1738 011470 050040 051501 020123
1739 011476 000043
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752 011500
1753 011500 010046
1754 011502 010146
1755 011504 010246
1756 011506 010346
1757 011510 010546
1758 011512 012746 020200
1759 011516 016605 000020
1760 011522 100004
1761 011524 005405
1762 011526 112766 000055 000001
1763 011534 005000 1$:
1764 011536 012703 011714
1765 011542 112723 000040
1766 011546 005002 2$:
1767 011550 016001 011704
1768 011554 160105 3$:
1769 011556 002402
1770 011560 005202
1771 011562 000774
1772 011564 060105 4$:
1773 011566 005702
1774 011570 001002
1775 011572 105716
1776 011574 100407
1777 011576 106316 5$:
1778 011600 103003
1779 011602 116663 000001 177777
1780 011610 052702 000060 6$:
1781 011614 052702 000040 7$:
1782 011620 110223
1783 011622 005720
1784 011624 020027 000010
1785 011630 002746
1786 011632 003002
1787 011634 010502
1788 011636 000764
1789 011640 105726 8$:
1790 011642 100003
1791 011644 116663 177777 177776
1792 011652 105013 9$:
    
```

SENDMG: .ASCIZ <15><12>/END PASS #/

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;*      MOV      NUM,-(SP)      ;:PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS                    ;:GO TO THE ROUTINE
$TYPDS:
MOV      R0,-(SP)      ;:PUSH R0 ON STACK
MOV      R1,-(SP)      ;:PUSH R1 ON STACK
MOV      R2,-(SP)      ;:PUSH R2 ON STACK
MOV      R3,-(SP)      ;:PUSH R3 ON STACK
MOV      R5,-(SP)      ;:PUSH R5 ON STACK
MOV      #20200,-(SP)   ;:SET BLANK SWITCH AND SIGN
MOV      20(SP),R5     ;:GET THE INPUT NUMBER
BPL      1$            ;:BR IF INPUT IS POS.
NEG      R5            ;:MAKE THE BINARY NUMBER POS.
MOVB    #'-(SP)       ;:MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;:ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3     ;:SETUP THE OUTPUT POINTER
MOVB    #' ,(R3)+     ;:SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;:CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;:GET THE CONSTANT
3$:      SUB      R1,R5   ;:FORM THIS BCD DIGIT
BLT     4$            ;:BR IF DONE
INC     R2            ;:INCREASE THE BCD DIGIT BY 1
4$:      ADD      R1,R5   ;:ADD BACK THE CONSTANT
TST     R2            ;:CHECK IF BCD DIGIT=0
BNE     5$            ;:FALL THROUGH IF 0
TSTB   (SP)          ;:STILL DOING LEADING 0'S?
BMI     7$            ;:BR IF YES
5$:      ASLB    (SP)    ;:MSD?
BCC     6$            ;:BR IF NO
MOVB   1(SP),-1(R3)   ;:YES--SET THE SIGN
6$:      BIS     #'0,R2  ;:MAKE THE BCD DIGIT ASCII
7$:      BIS     #' ,R2  ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+     ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST    (R0)+        ;:JUST INCREMENTING
CMP    R0,#10       ;:CHECK THE TABLE INDEX
BLT    2$            ;:GO DO THE NEXT DIGIT
BGT    8$            ;:GO TO EXIT
MOV    R5,R2        ;:GET THE LSD
BR     6$            ;:GO CHANGE TO ASCII
8$:      TSTB   (SP)+   ;:WAS THE LSD THE FIRST NON-ZERO?
BPL    9$            ;:BR IF NO
MOVB   -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
9$:      CLRB   (R3)    ;:SET THE TERMINATOR
    
```



```

1793 011654 012605      MOV      (SP)+,R5      ;; POP STACK INTO R5
1794 011656 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
1795 011660 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
1796 011662 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
1797 011664 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
1799 011666 104401 011714  TYPE      $DBLK      ;; NOW TYPE THE NUMBER
1799 011672 016666 000002 000004  MOV      2(SP),4(SP)  ;; ADJUST THE STACK
1800 011700 012616      MOV      (SP)+,(SP)
1801 011702 000002      RTI
1802 011704 023420      $DTBL: 10000.
1803 011706 001750      1000.
1804 011710 000144      100.
1805 011712 000012      10.
1806 011714 000004      $DBLK: .BLKW 4
1807
1808
1809
1810
1811 011724 000004      ;;*****
1812 011726 004737 003402 012026  *TEST 53      MISC. EXTERNAL LOGIC TEST
1813 011732 012737 040000      ;;*****
1814 011740      TST53: SCOPE
1815 011740 005077 167364      EXTST: JSR      PC,SETADD      ;SET UP ADDRESS AND VECTOR
1815 011744 012777 125252 167356      2$:      MOV      #BIT14,EXTCNT      ;LOAD COUNT
1817 011752 017737 167350 012024      1$:
1818 011760 013777 012024 167340      CLR      @GRDIO      ;CLEAR OUTPUT REGISTER
1819 011766 005077 167336      MOV      #125252,@GRDIO      ;LOAD OUTPUT
1820 011772 012777 052525 167330      MOV      @GRDAI,EXTTMP      ;READ INPUT
1821 012000 017737 167322 012024      MOV      EXTTMP,@GRDAI      ;CLEAR INPUT
1822 012006 013777 012024 167312      CLR      @GRDIO      ;CLEAR OUTPUT REGISTER
1823 012014 005337 012026      MOV      #52525,@GRDIO      ;LOAD OUTPUT
1824 012020 001347      MOV      @GRDAI,EXTTMP      ;READ INPUT
1825 012022 000743      MOV      EXTTMP,@GRDAI      ;CLEAR INPUT
1826
1827 012024 000000      DEC      EXTCNT      ;FINISHED COUNT
1828 012026 000000      BNE      1$
1829
1830
1831
1832
1833
1834
1835
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
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

1829      ::*****
1830      :*TEST 54      COULTER INTERFACE TEST
1831      :*****
1832      012030 000004      †ST54: SCOPE
1833      012032 004737 003402      COULTR: JSR      PC,SETADD      ;SETUP BUS ADDRESS AND VECTOR
1834      012036 005077 167262      CLR      @GRSTAT      ;CLEAR INPUT STATUS REG
1835      012042 012777 012104 167270      MOV      #20@,@GRIVA      ;LOAD INPUT VECTOR
1836      012050 012777 000340 167264      MOV      #340,@GRIVSA
1837      012056 104401      1$:      TYPE
1838      012060 012272      RUNMSG      ;TELL OPERATOR TO "RUN SAMPLE"
1839      012062 012701 015646      MOV      #BUFFER,R1      ;LOAD POINTER FOR RESULTS
1840      012066 012777 000100 167230 4$:      MOV      #BIT6,@GRSTAT      ;ENABLE INTERRUPT
1841      012074 005037 177776      CLR      PS      ;LOWER PSW
1842      012100 000001      WAIT      ;WAIT FOR OPERATOR
1843      012102 000771      BR      4$
1844
1845      012104 022626      20$:     CMP      (SP)+,(SP)+      ;POP STACK
1846      012106 012737 000106 012270      MOV      #70,100$      ;LOAD # OF 1 MSEC DELAYS
1847      012114 012700 000240      6$:      MOV      #240,R0      ;LOAD 1 MSEC. DELAY WEIGHT
1848      ;240 FOR 11/05 TYPE
1849      ;620 FOR 11/40 TYPE
1850      ;DELAY
1851      012120 005300      3$:      DEC      R0
1852      012122 001376      BNE     3$
1853      012124 005337 012270      DEC      100$      ;FINISHED ALL MSEC. DELAY ?
1854      012130 001371      BNE     6$
1855      012132 017700 167170      MOV      @GRDAI,R0      ;BR IF NOT
1856      012136 010021      MOV      R0,(R1)+      ;SAVE RESULTS
1857      012140 012777 177777 167160      MOV      #-1,@GRDAI      ;SAVE IN BUFFER
1858      012146 042700 007777      BIC     #7777,R0      ;CLEAR INPUT
1859      012152 001345      BNE     4$      ;MASK
1860      012154 012741 123456      MOV      #123456,-(R1)      ;BR IF NOT LAST SAMPLE
1861      ;LOAD TERM.
1862      012160 012703 015646      5$:      MOV      #BUFFER,R3
1863      012164 012300      MOV      (R3)+,R0      ;LOAD RESULT POINTER
1864      012166 022700 123456      CMP      #123456,R0      ;GET RESULT
1865      012172 001731      BEQ     1$      ;TEST FOR TERM.
1866      012174 004737 012202      JSR     PC,40$      ;BR IF DONE
1867      012200 000771      BR      5$      ;CONVERT AND TYPE
1868      ;CONT. TYPING UNTIL DONE
1869
1870      012202 012737 000055 012314 40$:     ;SUBROUTINE FOR THE COULTER TEST
1871      012210 012702 012321      MOV      #55,MSGRUS+2      ;FIX ASCII MESSAGE
1872      012214 012737 000004 012270      MOV      #MSPNT1,R2      ;LOAD DEST. POINTER
1873      012222 000404      MOV      #4,100$      ;LOAD COUNT
1874      BR      12$
1875      012224 006000      10$:     ROR      R0
1876      012226 006000      ROR      R0
1877      012230 006000      ROR      R0
1878      012232 006000      ROR      R0
1879      012234 010001      12$:     MOV      R0,R1      ;LOAD R1
1880      012236 042701 177760      BIC     #177760,R1      ;MASK
1881      012242 052701 000060      BIS     #60,R1      ;MAKE DIGIT
1882      012246 110142      MOVB    R1,-(R2)      ;SAVE DIGIT
1883      012250 005337 012270      DEC      100$      ;FINISHED ?
1884      012254 001363      BNE     10$      ;BR IF NOT

```

1885	012256	000337	012314		SWAB	MSGRUS+2			;ADJUST MESSAGE
1886	012262	104401			TYPE				
1887	012264	012312			MSGRUS				
1889	012266	000207			RTS	PC			;EXIT
1889									
1890	012270	000000			100\$:	0			
1891	012272	005015	051012	047125	RUNMSG:	.ASCIZ	<15><12><12>/RUN SAMPLE/<15><12>		
1892	012300	051440	046501	046120					
1893	012306	006505	000012						
1894	012312	005015	047055	047116	MSGRUS:	.ASCII	<15><12>/-NNNN/		
1895	012320	116							
1896	012321	000			MSPNT1:	.BYTE	0		
1897					.EVEN				
1898									
1899	012322	052123	052101	051525	EM1:	.ASCIZ	/STATUS REGISTER IN ERROR/		
1900	012330	051040	043505	051511					
1901	012336	042524	020122	047111					
1902	012344	042440	051122	051117					
1903	012352	000							
1904	012353	111	050116	052125	EM2:	.ASCIZ	/INPUT REGISTER IN ERROR/		
1905	012360	051040	043505	051511					
1906	012366	042524	020122	047111					
1907	012374	042440	051122	051117					
1908	012402	000							
1909	012403	117	052125	052520	EM3:	.ASCIZ	/OUTPUT REGISTER IN ERROR/		
1910	012410	020124	042522	044507					
1911	012416	052123	051105	044440					
1912	012424	020116	051105	047522					
1913	012432	000122							
1914	012434	047111	052520	020124	EM4:	.ASCIZ	/INPUT FAILED TO INTERRUPT/		
1915	012442	040506	046111	042105					
1916	012450	052040	020117	047111					
1917	012456	042524	051122	050125					
1918	012464	000124							
1919	012466	052517	050124	052125	EM5:	.ASCIZ	/OUTPUT FAILED TO INTERRUPT/		
1920	012474	043040	044501	042514					
1921	012502	020104	047524	044440					
1922	012510	052116	051105	052522					
1923	012516	052120	000						
1924	012521	125	042516	050130	EM6:	.ASCIZ	/UNEXPECTED INTERRUPT/		
1925	012526	041505	042524	020104					
1926	012534	047111	042524	051122					
1927	012542	050125	000124						
1928	012546	050117	051105	052101	EM7:	.ASCIZ	/OPERATOR INTERVENTION ERROR/		
1929	012554	051117	044440	052116					
1930	012562	051105	042526	052116					
1931	012570	047511	020116	051105					
1932	012576	047522	000122						
1933	012602	047111	042524	051122	EM10:	.ASCIZ	/INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/		
1934	012610	050125	020124	047111					
1935	012616	052520	020124	044502					
1936	012624	020124	040506	046111					
1937	012632	042105	052040	020117					
1938	012640	042523	020124	047111					
1939	012646	052520	020124	042522					
1940	012654	042101	020131	046106					

1941	012662	043501	000						
1942	012665	116	047117	044455	EM11:	.ASCIZ	/NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/		
1943	012672	052116	051105	052522					
1944	012700	052120	044440	050116					
1945	012706	052125	041040	052111					
1946	012714	051440	052105	044440					
1947	012722	050116	052125	051040					
1948	012730	040505	054504	043040					
1949	012736	040514	000107						
1950									
1951	012742	051105	050122	020103	DH1:	.ASCIZ	/ERRPC DRADD STATUS EXPECTED/		
1952	012750	020040	051104	042101					
1953	012756	020104	020040	052123					
1954	012764	052101	051525	020040					
1955	012772	054105	042520	052103					
1956	013000	042105	000						
1957	013003	105	051122	041520	DH2:	.ASCIZ	/ERRPC DRADD INPUT EXPECTED/		
1958	013010	020040	042040	040522					
1959	013016	042104	020040	044440					
1960	013024	050116	052125	020040					
1961	013032	042440	050130	041505					
1962	013040	042524	000104						
1963	013044	051105	050122	020103	DH3:	.ASCIZ	/ERRPC DRADD OUTPUT EXPECTED/		
1964	013052	020040	051104	042101					
1965	013060	020104	020040	052517					
1966	013066	050124	052125	020040					
1967	013074	054105	042520	052103					
1968	013102	042105	000						
1969	013105	105	051122	041520	DH4:	.ASCIZ	/ERRPC DRADD/		
1970	013112	020040	042040	040522					
1971	013120	042104	000						
1972	013123	105	051122	041520	DH10:	.ASCIZ	/ERRPC DRADD STATUS EXPECT INPUT BIT/		
1973	013130	020040	042040	040522					
1974	013136	042104	020040	051440					
1975	013144	040524	052524	020123					
1976	013152	042440	050130	041505					
1977	013160	020124	044440	050116					
1978	013166	052125	041040	052111					
1979	013174	000							
1980		013176							
1981	013176	001116	001316	001126	DT1:	.EVEN	\$ERRPC, DRADD, \$BDDAT, \$GDDAT, 0		
1982	013204	001124	000000						
1983	013210	001116	001316	000000	DT4:	\$ERRPC, DRADD, 0			
1984	013216	001116	001316	001126	DT10:	\$ERRPC, DRADD, \$BDDAT, \$GDDAT, BRLEV3, 0			
1985	013224	001124	001356	000000					

1986
1987
1989
1999
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041

013232
013232 104407
013234 032777 040000 165676
013242 001111
013244 000416
013246 013746 000004
013252 012737 013272 000004
013260 005737 177060
013264 012637 000004
013270 000463
013272 022626
013274 012637 000004
013300 000423
013302
013302 032777 000400 165630
013310 001404
013312 127737 165622 001102
013320 001462
013322 105737 001103
013326 001421
013330 123737 001115 001103
013336 101015
013340 032777 001000 165572
013346 001404
013350 013737 001110 001106
013356 000443
013360 105037 001103
013364 005037 001166
013370 000415
013372 032777 004000 165540
013400 001011
013402 005737 001100
013406 001406
013410 005237 001104
013414 023737 001166 001104
013422 002021
013424 012737 000001 001104
013432 013737 013502 001166
013440 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

$SCOPE:
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
1$: BIT #BIT14,$SWR ;:LOOP ON PRESENT TEST?
BNE $OVER ;:YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;: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 #5,$@#ERRVEC ;:SET FOR TIMEOUT
TST @#177060 ;:TIME OUT ON XOR?
MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $SVLAD ;:GO TO THE NEXT TEST
5$: CMP (SP)+, (SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
BR 7$ ;:LOOP ON THE PRESENT TEST
6$; *****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,$SWR ;:LOOP ON SPEC. TEST?
BEQ 2$ ;:BR IF NO
CMPB @SWR,$STNM ;:ON THE RIGHT TEST? SWR<7:0>
BEQ $OVER ;:BR IF YES
2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
BEQ 3$ ;:BR IF NO
CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3$ ;:BR IF NO
BIT #BIT09,$SWR ;:LOOP ON ERROR?
BEQ 4$ ;:BR IF NO
7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1$ ;:ESCAPE TO THE NEXT TEST
3$: BIT #BIT11,$SWR ;:INHIBIT ITERATIONS?
BNE 1$ ;:BR IF YES
TST $PASS ;:IF FIRST PASS OF PROGRAM
BEQ 1$ ;:INHIBIT ITERATIONS
INC $ICNT ;:INCREMENT ITERATION COUNT
CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
BGE $OVER ;:BR IF MORE ITERATION REQUIRED
1$: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
$SVLAD: INCB $STNM ;:COUNT TEST NUMBERS
```

```

2042 013444 011637 001106      MOV      (SP), $LPADR      ;; SAVE SCOPE LOOP ADDRESS
2043 013450 011637 001110      MOV      (SP), $LPERR     ;; SAVE ERROR LOOP ADDRESS
2044 013454 005037 001170      CLR      $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2045 013460 112737 000001 001115  MOVVB    #1, $ERMAX       ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2046 013466 013777 001102 165446 $OVER:   MOV      $STNM, @DISPLAY  ;; DISPLAY TEST NUMBER
2047 013474 013776 001106      MOV      $LPADR, (SP)    ;; FUDGE RETURN ADDRESS
2048 013500 000002      RTI                    ;; FIXES PS
2049 013502 003720      $MXCNT: 2000.          ;; MAX. NUMBER OF ITERATIONS
2050      .SBTTL  ERROR HANDLER ROUTINE
2051
2052      ;*****
2053      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2054      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2055      ;*AND GO TO $ERRTYP ON ERROR
2056      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2057      ;*SW15=1      HALT ON ERROR
2058      ;*SW13=1      INHIBIT ERROR TYPEOUTS
2059      ;*SW09=1     LOOP ON ERROR
2060      ;*CALL
2061      ;*
2062      ;*      ERROR      N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
2063
2064      $ERROR:
2065      013504 104407      CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
2066 013506 105237 001103 7$:      INCB      $ERFLG      ;; SET THE ERROR FLAG
2067 013512 001775      BEQ      7$          ;; DON'T LET THE FLAG GO TO ZERO
2068 013514 013777 001102 165420  MOV      $STNM, @DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
2069 013522 005237 001112      INC      $ERTTL      ;; INC THE ERROR COUNT
2070 013526 011637 001116      MOV      (SP), $ERRPC   ;; GET ADDRESS OF ERROR INSTRUCTION
2071 013532 162737 000002 001116  SUB      #2, $ERRPC
2072 013540 117737 165352 001114  MOVVB    @ $ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
2073 013546 032777 020000 165364  BIT      #BIT13, @SWR   ;; SKIP TYPEOUT IF SET
2074 013554 001004      BNE      20$         ;; SKIP TYPEOUTS
2075 013556 004737 013642      JSR      PC, $ERRTYP   ;; GO TO USER ERROR ROUTINE
2076 013562 104401 001173      TYPE    , $CRLF
2077 013566 005777 165346 20$:    TST      @SWR        ;; HALT ON ERROR
2078 013572 100002      BPL      3$          ;; SKIP IF CONTINUE
2079 013574 000000      HALT
2080 013576 104407      CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
2081 013600 032777 001000 165332 3$:    BIT      #BIT09, @SWR  ;; LOOP ON ERROR SWITCH SET?
2082 013606 001402      BEQ      4$          ;; BR IF NO
2083 013610 013716 001110      MOV      $LPERR, (SP)  ;; FUDGE RETURN FOR LOOPING
2084 013614 005737 001170 4$:    TST      $ESCAPE     ;; CHECK FOR AN ESCAPE ADDRESS
2085 013620 001402      BEQ      5$          ;; BR IF NONE
2086 013622 013715 001170      MOV      $ESCAPE, (SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
2087 013626
2088 013626 022737 011444 000042 5$:    CMP      #$ENDAD, @#42 ;; ACT-11 AUTO-ACCEPT?
2089 013634 001001      BNE      6$          ;; BRANCH IF NO
2090 013636 000000      HALT                ;; YES
2091 013640
2092 013640 000002 6$:    RTI                    ;; RETURN

```

```

2093
2094
2095
2096
2097
2098
2099
2100
2101 013642
2102 013642 104401 001173
2103 013646 010046
2104 013650 005000
2105 013652 153700 001114
2106 013656 001004
2107
2108 013660 013746 001116
2109
2110 013664 104402
2111 013666 000426
2112 013670 005300
2113 013672 006300
2114 013674 006300
2115 013676 006300
2116 013700 062700 001176
2117 013704 012037 013714
2118 013710 001404
2119 013712 104401
2120 013714 000000
2121 013716 104401 001173
2122 013722 012037 013732
2123 013726 001404
2124 013730 104401
2125 013732 000000
2126 013734 104401 001173
2127 013740 011000
2128 013742 001004
2129 013744 012600
2130 013746 104401 001173
2131 013752 000207
2132 013754
2133 013754 013046
2134 013756 104402
2135 013760 005710
2136 013762 001770
2137 013764 104401 013772
2138 013770 000771
2139 013772 020040 000
2140 013776

```

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

```



```

2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167 013776 017646 000000
2168 014002 116637 000001 014221
2169 014010 112637 014223
2170 014014 062716 000002
2171 014020 000406
2172 014022 112737 000001 014221
2173 014030 112737 000006 014223
2174 014036 112737 000005 014220
2175 014044 010346
2176 014046 010446
2177 014050 010546
2178 014052 113704 014223
2179 014056 005404
2180 014060 062704 000006
2181 014064 110437 014222
2182 014070 113704 014221
2183 014074 016605 000012
2184 014100 005003
2185 014102 006105
2186 014104 000404
2187 014106 006105
2188 014110 006105
2189 014112 006105
2190 014114 010503
2191 014116 006103
2192 014120 105337 014222
2193 014124 100016
2194 014126 042703 177770
2195 014132 001002
2196 014134 005704

.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.
;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;* MOV NUM,-(SP) ;;NUMBER TO BE TYPED
;* TYPOS ;;CALL FOR TYPEOUT
;* .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;* .BYTE M ;;M=1 OR 0
; ;;1=TYPE LEADING ZEROS
; ;;0=SUPPRESS LEADING ZEROS
;$STYPON---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
;$TYPOS: MOV 2(SP),-(SP) ;;PICKUP THE MODE
MOV 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
MOVB (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
ADD #2,(SP) ;;ADJUST RETURN ADDRESS
BR $TYPON
;$TYPOC: MOV #1,$OFILL ;;SET THE ZERO FILL SWITCH
MOV #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
;$TYPON: MOV #5,$OCNT ;;SET THE ITERATION COUNT
MOV R3,-(SP) ;;SAVE R3
MOV R4,-(SP) ;;SAVE R4
MOV R5,-(SP) ;;SAVE R5
MOV #5,$OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
NEG R4
ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
MOV R4,$OMODE ;;SAVE IT FOR USE
MOV $OFILL,R4 ;;GET THE ZERO FILL SWITCH
MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
CLR R3 ;;CLEAR THE OUTPUT WORD
1$: ROL R5 ;;ROTATE MSB INTO "C"
BR 3$ ;;GO DO MSB
2$: ROL R5 ;;FORM THIS DIGIT
ROL R5
MOV R5,R3
3$: ROL R3 ;;GET LSB OF THIS DIGIT
DECB $OMODE ;;TYPE THIS DIGIT?
BPL 7$ ;;BR IF NO
BIC #177770,R3 ;;GET RID OF JUNK
BNE 4$ ;;TEST FOR 0
TST R4 ;;SUPPRESS THIS 0?
    
```

2197	014136	001403			BEQ	5\$:: BR IF YES
2198	014140	005204			INC	R4		:: DON'T SUPPRESS ANYMORE 0'S
2199	014142	052703	000060		BIS	#'0,R3		:: MAKE THIS DIGIT ASCII
2200	014146	052703	000040		BIS	#' R3		:: MAKE ASCII IF NOT ALREADY
2201	014152	110337	014216		MOVB	R3,8\$:: SAVE FOR TYPING
2202	014156	104401	014216		TYPE	8\$:: GO TYPE THIS DIGIT
2203	014162	105337	014220		DECB	\$OCNT		:: COUNT BY 1
2204	014166	003347			BGT	2\$:: BR IF MORE TO DO
2205	014170	002402			BLT	6\$:: BR IF DONE
2206	014172	005204			INC	R4		:: INSURE LAST DIGIT ISN'T A BLANK
2207	014174	000744			BR	2\$:: GO DO THE LAST DIGIT
2208	014176	012605			MOV	(SP)+,R5		:: RESTORE R5
2209	014200	012604			MOV	(SP)+,R4		:: RESTORE R4
2210	014202	012603			MOV	(SP)+,R3		:: RESTORE R3
2211	014204	016666	000002	000004	MOV	2(SP),4(SP)		:: SET THE STACK FOR RETURNING
2212	014212	012616			MOV	(SP)+,(SP)		
2213	014214	000002			RTI			:: RETURN
2214	014216	000			.BYTE	0		:: STORAGE FOR ASCII DIGIT
2215	014217	000			.BYTE	0		:: TERMINATOR FOR TYPE ROUTINE
2216	014220	000			\$OCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
2217	014221	000			\$OFILL:	.BYTE	0	:: ZERO FILL SWITCH
2218	014222	000000			\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

.SBTTL POWER DOWN AND JP ROUTINES

::*****

:POWER DOWN ROUTINE

014224	012737	014370	000024
014232	012737	000340	000026
014240	010046		
014242	010146		
014244	010246		
014246	010346		
014250	010446		
014252	010546		
014254	017746	164660	
014260	010637	014374	
014264	012737	014276	000024
014272	000000		
014274	000776		

```

$PWRDN: MOV $SILLUP,2#PWRVEC ::SET FOR FAST UP
MOV $340,2#PWRVEC+2 ::PRIO:7
MOV RO,-(SP) ::PUSH RO ON STACK
MOV R1,-(SP) ::PUSH R1 ON STACK
MOV R2,-(SP) ::PUSH R2 ON STACK
MOV R3,-(SP) ::PUSH R3 ON STACK
MOV R4,-(SP) ::PUSH R4 ON STACK
MOV R5,-(SP) ::PUSH R5 ON STACK
MOV 2SWR,-(SP) ::PUSH 2SWR ON STACK
MOV SP,$SAVR6 ::SAVE SP
MOV $PWRUP,2#PWRVEC ::SET UP VECTOR
HALT
BR -2 ::HANG UP

```

::*****

:POWER UP ROUTINE

014276	012737	014370	000024
014304	013706	014374	
014310	005037	014374	
014314	005237	014374	
014320	001375		
014322	012677	164612	
014326	012605		
014330	012604		
014332	012603		
014334	012602		
014336	012601		
014340	012600		
014342	012737	014224	000024
014350	012737	000340	000026
014356	104401		
014360	014376		
014362	012716		
014364	003316		
014366	000002		
014370	000000		
014372	000776		
014374	000000		
014376	005015	042522	052123
014404	051101	044524	043516
014412	040440	052106	051105
014420	040440	050040	053517
014426	051105	043040	044501
014434	052514	042522	005015
014442	000		
014444			

```

$PWRUP: MOV $SILLUP,2#PWRVEC ::SET FOR FAST DOWN
MOV $SAVR6,SP ::GET SP
CLR $SAVR6 ::WAIT LOOP FOR THE TTY
15: INC $SAVR6 ::WAIT FOR THE INC
BNE 15 ::OF WORD
MOV (SP)+,2SWR ::POP STACK INTO 2SWR
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)+,RO ::POP STACK INTO RO
MOV $PWRDN,2#PWRVEC ::SET UP THE POWER DOWN VECTOR
MOV $340,2#PWRVEC+2 ::PRIO:7
TYPE PWRMSG ::REPORT THE POWER FAILURE
MOV (PC)+,(SP) ::POWER FAIL MESSAGE POINTER
$PWRPD: .WORD IOTEST ::RESTART AT IOTEST
RTI ::RESTART ADDRESS
$SILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
BR -2 ::BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0 ::PUT THE SP HERE
PWRMSG: .ASCIIZ <15><12>/RESTARTING AFTER A POWER FAILURE<<15><12>

```

.EVEN

2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325

.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 \$TFPLG ;; IS THERE A TERMINAL?
BPL 1\$;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3\$;; LEAVE
1\$: MOV RC,-(SP) ;; SAVE RO
MOV 22(SP),RO ;; GET ADDRESS OF ASCIZ STRING
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
6\$: MOV (SP)+,RO ;; RESTORE RO
3\$: ADD #2,(SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4\$: CMPB #HT,(SP) ;; BRANCH IF <HT>
BEQ 8\$
CMPB #CR LF, (SP) ;; BRANCH IF NOT <CR LF>
BNE 5\$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
\$CHARCNT
CLRB ;; 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 2\$ OF FILLER CHARS. NEEDED
\$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
DECB \$CHARCNT ;; DO NOT COUNT AS A COUNT
BR 7\$;; LOOP

;HORIZONTAL TAB PROCESSOR

8\$: MOVB # (SP) ;; REPLACE TAB WITH SPACE
9\$: JSR PC,\$TYPEC ;; TYPE A SPACE
BITB #7,\$CHARCNT ;; BRANCH IF NOT AT
BNE 9\$;; TAB STOP

2326	014610	005726				TST	(SP)+	:: POP SPACE OFF STACK
2327	014612	000724				BR	25	:: GET NEXT CHARACTER
2328	014614	105777	164330		\$TYPEC:	TSTB	\$STPS	:: WAIT UNTIL PRINTER IS READY
2329	014620	100375				BPL	\$TYPEC	
2330	014622	116677	000002	164322		MOVB	2(SP), \$STPB	:: LOAD CHAR TO BE TYPED INTO DATA REG.
2331	014630	122766	000015	000002		CMPB	#CR, 2(SP)	:: IS CHARACTER A CARRIAGE RETURN?
2332	014636	001203				SNE	15	:: BRANCH IF NO
2333	014640	105037	014660			CLRB	\$CHARCNT	:: YES--CLEAR CHARACTER COUNT
2334	014644	000406				BR	\$TYPEX	:: EXIT
2335	014646	122766	0JQC12	000002	15:	CMPB	#LF, 2(SP)	:: IS CHARACTER A LINE FEED?
2336	014654	001402				BEQ	\$TYPEX	:: BRANCH IF YES
2337	014656	125227				INCB	(PC)+	:: COUNT THE CHARACTER
2338	014658	000000			\$CHARCNT:	.WORD	0	:: CHARACTER COUNT STORAGE
2339	014662	000207			\$TYPEX:	RTS	PC	

```

2341 .SBTTL TTY INPUT ROUTINE
2342
2343 :*****
2344 .ENABL LSB
2345
2346 :*****
2347 :*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2348 :*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2349 :*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2350 :*WHEN OPERATING IN TTY FLAG MODE.
2351 014664 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
2352 014672 001074 BNE 15$ ;; BRANCH IF NO
2353 014674 105777 164244 TSTB @STKS ;; CHAR THERE?
2354 014700 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
2355 014702 117746 164240 MOVB @STKB,-(SP) ;; SAVE THE CHAR
2356 014706 042716 177600 BIC #177,(SP) ;; STRIP-OFF THE ASCII
2357 014712 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
2358 014716 001062 BNE 15$ ;; NO, RETURN TO USER
2359 014720 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
2360 014726 001456 BEQ 15$ ;; BRANCH IF YES
2361
2362 014730 104401 015536 $GTSWR: TYPE .SCNTLG ;; ECHO THE CONTROL-G (↑G)
2363 014734 104401 015543 TYPE $MSWR ;; TYPE CURRENT CONTENTS
2364 014740 013746 000176 MOV $WREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
2365 014744 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2366 014746 104401 015554 TYPE .SMNEW ;; PROMPT FOR NEW SWR
2367 014752 025046 19$: CLR -(SP) ;; CLEAR COUNTER
2368 014754 005046 CLR -(SP) ;; THE NEW SWR
2369 014756 105777 164162 7$: TSTB @STKS ;; CHAR THERE?
2370 014762 100375 BPL 7$ ;; IF NOT TRY AGAIN
2371
2372 014764 117746 164156 MOVB @STKB,-(SP) ;; PICK UP CHAR
2373 014770 042716 177600 BIC #177,(SP) ;; MAKE IT 7-BIT ASCII
2374
2375
2376
2377 014774 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
2378 015000 001005 BNE 10$ ;; BRANCH IF NOT
2379 015002 104401 015531 TYPE .SCNTLU ;; YES, ECHO CONTROL-U (↑U)
2380 015006 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
2381 015012 000757 BR 19$ ;; LET'S TRY IT AGAIN
2382
2383
2384 015014 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
2385 015020 001022 BNE 16$ ;; BRANCH IF NO
2386 015022 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
2387 015026 001403 BEQ 11$ ;; BRANCH IF YES
2388 015030 016677 000002 164102 MOV 2(SP),@SWR ;; SAVE NEW SWR
2389 015036 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
2390 015042 104401 001173 14$: TYPE .SCRLF ;; ECHO <CR> AND <LF>
2391 015046 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
2392 015054 001003 BNE 15$ ;; BRANCH IF NOT
2393 015056 012777 000100 164060 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
2394 015064 000002 15$: RTI ;; RETURN
2395 015066 004737 014614 16$: JSR PC,$TYPEC ;; ECHO CHAR
2396 015072 021627 000060 CMP (SP),#60 ;; CHAR < 0?
    
```

```

2397 015076 002420          BLT      18$          ;; BRANCH IF YES
2398 015100 021627 000067    CMP      (SP),#67    ;; CHAR > 7?
2399 015104 003015          BGT      18$          ;; BRANCH IF YES
2400 015106 042726 000060    BIC      #60,(SP)+   ;; STRIP-OFF ASCII
2401 015112 005766 000002    TST      2(SP)      ;; IS THIS THE FIRST CHAR
2402 015116 001403          BEQ      17$          ;; BRANCH IF YES
2403 015120 006316          ASL      (SP)       ;; NO, SHIFT PRESENT
2404 015122 006316          ASL      (SP)       ;; CHAR OVER TO MAKE
2405 015124 006316          ASL      (SP)       ;; ROOM FOR NEW ONE.
2406 015126 005266 000002    17$: INC      2(SP)    ;; KEEP COUNT OF CHAR
2407 015132 056616 177776    BIS      -2(SP),(SP) ;; SET IN NEW CHAR
2408 015136 000707          BR       7$          ;; GET THE NEXT ONE
2409 015140 104401 001172    18$: TYPE $QUES    ;; TYPE ?<CR><LF>
2410 015144 000720          BR       20$          ;; SIMULATE CONTROL-U
2411          .DSABL  LSB
2412
2413
2414
2415 *****
2416 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2417 *CALL:
2418 *      RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
2419 *      RETURN HERE   ;; CHARACTER IS ON THE STACK
2420 *                   ;; WITH PARITY BIT STRIPPED OFF
2421
2422 $RDCHR: MOV      (SP),-(SP) ;; PUSH DOWN THE PC
2423 015146 011546          MOV      4(SP),2(SP) ;; SAVE THE PS
2424 015150 016666 000004 000002 1$: TSTB     2$TKS     ;; WAIT FOR
2425 015156 105777 163762    BPL      1$          ;; A CHARACTER
2426 015162 100375          MOVB     2$TKB,4(SP) ;; READ THE TTY
2427 015164 117766 163756 000004 BIC      #1C(177),4(SP) ;; GET RID OF JUNK IF ANY
2428 015172 042766 177600 000004 CMP      4(SP),#23   ;; IS IT A CONTROL-S?
2429 015200 026627 000004 000023 BNE      3$          ;; BRANCH IF NO
2430 015210 105777 163730 2$: TSTB     2$TKS     ;; WAIT FOR A CHARACTER
2431 015214 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
2432 015216 117746 163724    MOVB     2$TKB,-(SP) ;; GET CHARACTER
2433 015222 042716 177600    BIC      #1C177,(SP) ;; MAKE IT 7-BIT ASCII
2434 015226 022627 000021    CMP      (SP)+,#21   ;; IS IT A CONTROL-Q?
2435 015232 001366          BNE      2$          ;; IF NOT DISCARD IT
2436 015234 000750          BR       1$          ;; YES, RESUME
2437 015236 026627 000004 000140 3$: CMP      4(SP),#140  ;; IS IT UPPER CASE?
2438 015244 002407          BLT      4$          ;; BRANCH IF YES
2439 015246 026627 000004 000175 CMP      4(SP),#175  ;; IS IT A SPECIAL CHAR?
2440 015254 003003          BGT      4$          ;; BRANCH IF YES
2441 015256 042766 000040 000004 BIC      #40,4(SP)   ;; MAKE IT UPPER CASE
2442 015264 000002          RTI             ;; GO BACK TO USER
2443 *****
2444 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2445 *CALL:
2446 *      RDLIN         ;; INPUT A STRING FROM THE TTY
2447 *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2448 *                   ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2449
2450 $RDLIN: MOV      R3,-(SP) ;; SAVE R3
2451 015266 010346          CLR      -(SP)     ;; CLEAR THE RUBOUT KEY
2452 015270 005046          1$: MOV      $TTYIN,R3 ;; GET ADDRESS
2453 015272 012703 015522

```



```

2453 015276 022703 015531 2$: CMP #STTYIN+7,R3 ;; BUFFER FULL?
2454 015302 101456 BLOS 4$ ;; BR IF YES
2455 015304 104410 RDCHR ;; GO READ ONE CHARACTER FROM THE TTY
2456 015306 112613 MOVB (SP)+,(R3) ;; GET CHARACTER
2457 015210 122713 000177 10$: CMPB #177,(R3) ;; IS IT A RUBOUT
2458 015314 001022 BNE 5$ ;; BR IF NO
2459 015316 005716 TST (SP) ;; IS THIS THE FIRST RUBOUT?
2460 015320 001007 BNE 6$ ;; BR IF NO
2461 015322 112737 000134 015520 MOVB #'\\,9$ ;; TYPE A BACK SLASH
2462 015330 104401 015520 TYPE 9$
2463 015334 012716 177777 MOV #-1,(SP) ;; SET THE RUBOUT KEY
2464 015340 005303 6$: DEC R3 ;; BACKUP BY ONE
2465 015342 020327 015522 CMP R3,#STTYIN ;; STACK EMPTY?
2466 015346 103434 BLO 4$ ;; BR IF YES
2467 015350 111337 015520 MOVB (R3),9$ ;; SETUP TO TYPEOUT THE DELETED CHAR.
2468 015354 104401 015520 TYPE 9$ ;; GO TYPE
2469 015360 000746 BR 2$ ;; GO READ ANOTHER CHAR.
2470 015362 005716 5$: TST (SP) ;; RUBOUT KEY SET?
2471 015364 001406 BEQ 7$ ;; BR IF NO
2472 015366 112737 000134 015520 MOVB #'\\,9$ ;; TYPE A BACK SLASH
2473 015374 104401 015520 TYPE 9$
2474 015400 005016 CLR (SP) ;; CLEAR THE RUBOUT KEY
2475 015402 122713 000025 7$: CMPB #25,(R3) ;; IS CHARACTER A CTRL U?
2476 015406 001003 BNE 8$ ;; BR IF NO
2477 015410 104401 015531 TYPE ,SCNTLU ;; TYPE A CONTROL "U"
2478 015414 000726 BR 1$ ;; GO START OVER
2479 015416 122713 000022 8$: CMPB #22,(R3) ;; IS CHARACTER A "↑"?
2480 015422 001011 BNE 3$ ;; BRANCH IF NO
2481 015424 105013 CLRB (R3) ;; CLEAR THE CHARACTER
2482 015426 104401 001173 TYPE ,SCRLF ;; TYPE A "CR" & "LF"
2483 015432 104401 015522 TYPE ,STTYIN ;; TYPE THE INPUT STRING
2484 015436 000717 BR 2$ ;; GO PICKUP ANOTHER CHARACTER
2485 015440 104401 001172 4$: TYPE ,SQUES ;; TYPE A '?'
2486 015444 000712 BR 1$ ;; CLEAR THE BUFFER AND LOOP
2487 015446 111337 015520 3$: MOVB (R3),9$ ;; ECHO THE CHARACTER
2488 015452 104401 015520 TYPE 9$
2489 015456 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN
2490 015462 001305 BNE 2$ ;; LOOP IF NOT RETURN
2491 015464 105063 177777 CLRB -(R3) ;; CLEAR RETURN (THE 15)
2492 015470 104401 001174 TYPE ,SLF ;; TYPE A LINE FEED
2493 015474 005726 TST (SP)+ ;; CLEAN RUBOUT KEY FROM THE STACK
2494 015476 012603 MOV (SP)+,R3 ;; RESTORE R3
2495 015500 011646 MOV (SP),-(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2496 015502 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
2497 015510 012766 015522 000004 MOV #STTYIN,4(SP)
2498 015516 000002 RTI ;; RETURN
2499 015520 000 9$: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
2500 015521 000 .BYTE 0 ;; TERMINATOR
2501 015522 000007 $TTYIN: .BLKB 7 ;; RESERVE 7 BYTES FOR TTY INPUT
2502 015531 136 006525 000012 $CNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
2503 015536 043536 005015 000 $CNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
2504 015543 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR = /
2505 015550 036440 000040 $MNEW: .ASCIZ / NEW = /
2506 015554 020040 042516 020127
2507 015562 020075 000
2508 015566 .EVEN
    
```

2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554

.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
```

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

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

.SBTTL TRAP TABLE

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

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

```
BUFFER: 0 ; 10 WORD BUFFER FOR THE COULTER INTERFACE TEST
.END
```


IOTTS1	003530	734	750*																		
IOTVEC=	000020	348*	560*	561*																	
LF	= 000012	254*	2335	2341																	
MINSIN	001362	524*	667*	690	752	1073															
MSGRUS	012312	1870*	1885*	1887	1894*																
MSPNT1	012321	1871	1896*																		
NBEXT	001314	502*	722*	1686	1692*	1696*															
NMBEXT	001312	501*	548*	604	722	1696															
NOTLCH	001334	513*	629*	1062	1092	1097	1131	1156	1162	1181	1258										
ODDJMP	00136C	523*	751*	752*	753*	1003	1015	1018	1030	1033	1045	1048	1057	1060							
PC	=%000007	1095	1133	1158	1182	1257	1337														
		274*	733*	749*	1716*	1719*	1729*	1734	1812*	1833*	1865*	1888*	2074*	2131*							
		2256	2309*	2316*	2323*	2337*	2339*	2395*													
PIRQ	= 177772	260*																			
PIRQVE=	000240	354*																			
PRO	= 000000	277*																			
PR1	= 000040	278*																			
PR2	= 000100	279*																			
PR3	= 000140	280*																			
PR4	= 000200	281*																			
PR5	= 000240	282*																			
PR6	= 000300	283*																			
PR7	= 000340	284*																			
PS	= 177776	257*	258	1841*																	
PSW	= 177776	258*	1380*	1419*	1426*	1445*	1455*	1475*	1481*	1508*	1614*	1640*	1652*	1659*							
		1666*	1672*	1680*																	
PWRMSG	014376	2255	2262*																		
PWRVEC=	000024	349*	566*	567*	2224*	2225*	2234*	2240*	2252*	2253*											
RBEG	001376	531	533*																		
RBEG1	001462	551*																			
RBEG2	003342	723*	1699																		
RDCHR	= 104410	2455	2550*																		
RDLIN	= 104411	2551*																			
RESVEC=	000010	344*																			
RUNMSG	012272	1838	1891*																		
RO	=%000000	265*	530*	532*	592	1183*	1187	1191*	1192	1196*	1197	1201*	1202	1206*							
		1207	1211*	1212	1216*	1217	1221*	1222	1226*	1227	1231*	1232	1236*	1237							
		1241*	1259*	1263	1267*	1268	1272*	1273	1277*	1278	1282*	1283	1287*	1288							
		1292*	1293	1297*	1298	1302*	1303	1307*	1308	1312*	1313	1317*	1364*	1396*							
		1499*	1601*	1698*	1726*	1729	1753	1763*	1767	1783	1784	1797*	1847*	1950*							
		1854*	1855	1857*	1862*	1863	1875*	1876*	1877*	1878*	1879	2103	2104*	2105*							
		2112*	2113*	2114*	2115*	2116*	2117	2122	2127*	2129*	2133	2135	2226	2251*							
		2292	2293*	2294	2297*	2518	2519*	2520	2521*	2522	2523*	2524*									
R1	=%000001	266*	725*	726*	727*	728	1189*	1190	1194*	1195	1199*	1200	1204*	1205							
		1209*	1210	1214*	1215	1219*	1220	1224*	1225	1229*	1230	1234*	1235	1239*							
		1240	1243	1265*	1266	1270*	1271	1275*	1276	1280*	1281	1285*	1286	1290*							
		1291	1295*	1296	1300*	1301	1305*	1306	1310*	1311	1315*	1316	1319	1754							
		1767*	1768	1772	1796*	1839*	1855*	1859*	1879*	1880*	1881*	1882	2227	2250*							
R2	=%000002	267*	536*	538	539*	724*	726	728*	729	730	1755	1766*	1770*	1773							
		1780*	1781*	1782	1787*	1795*	1871*	1882*	2228	2249*											
R3	=%000003	268*	537*	540*	543	547*	548	1756	1764*	1765*	1779*	1782*	1791*	1792*							
		1794*	1861*	1862	2175	2184*	2190*	2191*	2194*	2199*	2200*	2201	2210*	2229							
		2248*	2450	2452*	2453	2456*	2457	2464*	2465	2467	2475	2479	2481*	2487							
		2489	2491*	2494*																	
R4	=%000004	269*	2176	2178*	2179*	2180*	2181	2182*	2196	2198*	2206*	2209*	2230	2247*							
R5	=%000005	270*	1757	1759*	1761*	1768*	1772*	1787	1793*	2177	2183*	2185*	2187*	2188*							

TST11	004234	862*																		
TST12	004332	875	880*																	
TST13	004430	893	899*																	
TST14	004516	915*																		
TST15	004604	930*																		
TST16	004654	937	943*																	
TST17	004716	949	955*																	
TST2	003624	770*																		
TST20	004754	960	966*																	
TST21	005016	972	978*																	
TST22	005060	985	993*																	
TST23	005150	1005	1011*																	
TST24	005234	1020	1026*																	
TST25	005320	1025	1041*																	
TST26	005404	1050	1055*																	
TST27	005564	1058	1089*																	
TST3	003662	775	780*																	
TST30	005744	1125*																		
TST31	006062	1150*																		
TST32	006222	1178*																		
TST33	006616	1248	1254*																	
TST34	007212	1324	1329*																	
TST35	007270	1338	1344*																	
TST36	007340	1351	1359*																	
TST37	007406	1366	1376*																	
TST4	003720	786	792*																	
TST40	007520	1392	1402*																	
TST41	007626	1422	1430*																	
TST42	007762	1449	1451	1459*																
TST43	010066	1478	1486*																	
TST44	010240	1491	1506	1518*																
TST45	010430	1520	1557*																	
TST46	010562	1559	1588*																	
TST47	010734	1593	1608	1620*																
TST5	003756	797	803*																	
TST50	011030	1637*																		
TST51	011124	1656*																		
TST52	011250	1674	1684*																	
TST53	011724	1811*																		
TST54	012030	1832*																		
TST6	004014	808	814*																	
TST7	004072	828*																		
TYPDS =	104405	1724	2545*																	
TYPE =	104401	596	600	607	611	615	630	634	649	653	668	672	701	705						
		1722	1725	1798	1837	1886	2075	2102	2119	2121	2124	2126	2130	2137						
		2202	2254	2305	2362	2363	2366	2379	2390	2409	2462	2468	2473	2477						
		2482	2483	2485	2488	2492	2541*													
		2110	2134	2365	2542*															
TYPOC =	104402	2544*																		
TYPON =	104404	605	2543*																	
TYPOS =	104403	406*	627*	646*	665*	684*	717*	2359	2508											
\$AUTOB	001134	401*																		
\$BDADR	001122	403*	773*	774	785*	795*	796	806*	807	819*	820	834*	835	850*						
\$2DDAT	001126	851*	852	873*	874	891*	892	905*	906	921*	922	935*	936	947*						
		948	958*	959	970*	971	984*	1002*	1003*	1004	1017*	1018*	1019	1032*						
		1033*	1034	1047*	1048*	1049	1067*	1068	1077*	1078	1101*	1102*	1103	1111*						

CLERIN	990#	999	1013	1028	1043	1154	1185	1261	1332	1530	1569				
CLEROT	999#	997	1012	1027	1042	1137	1153	1165	1184	1188	1193	1198	1203	1208	1213
	1218	1223	1228	1233	1238	1260	1264	1269	1274	1279	1284	1289	1294	1299	1304
	1309	1314	1331	1529	1568	1814	1819								
CLRVCY	355#	755	1510	1610	1648	1676									
COMMEN	355#														
ENDCOM	355#														
ERROR	249#	694	776	787	798	809	822	837	854	876	894	908	924	938	950
	961	973	986	1006	1021	1036	1051	1070	1080	1105	1116	1141	1170	1249	1325
	1339	1352	1367	1394	1397	1412	1425	1441	1450	1454	1468	1480	1505	1540	1550
	1578	1607	1625	1628	1646	1673									
ESCAPE	355#														
GETPRI	355#														
GETSWR	233#	355#	620	639	658	677	710								
MULT	355#														
NEWST	355#	759	767	777	789	800	811	825	941	859	877	896	912	927	940
	952	963	975	990	1008	1023	1038	1052	1085	1122	1147	1175	1251	1326	1341
	1356	1373	1399	1427	1456	1483	1515	1554	1585	1617	1634	1653	1681	1808	1829
PJP	355#														
PUSH	355#	1792	2245	2246											
REPORT	355#	1752	2226	2232											
SCOPE	355#														
	250#	762	770	780	792	803	814	828	844	862	880	899	915	930	943
	955	966	978	993	1011	1026	1041	1055	1088	1125	1150	1179	1254	1329	1344
	1359	1376	1402	1430	1459	1486	1518	1557	1588	1620	1637	1656	1684	1711	1811
	1832														
SETPRI	355#														
SETTRA	253#	2542	2543	2544	2545	2547	2549	2550	2551						
SETUP	355#	552													
SKIP	355#	775	786	797	808	821	875	893	937	949	960	972	985	1005	1020
	1035	1050	1058	1104	1115	1140	1169	1248	1324	1338	1351	1366	1392	1422	1449
	1451	1478	1491	1506	1520	1528	1539	1549	1559	1567	1577	1593	1608	1674	
SLASH	355#														
SPACE	355#														
STARS	355#	384	428	759	761	767	769	777	779	789	791	800	802	811	813
	825	827	841	843	859	861	877	879	896	898	912	914	927	923	940
	942	952	954	963	965	975	977	990	992	1008	1010	1023	1025	1038	1040
	1052	1054	1085	1087	1122	1124	1147	1149	1175	1177	1251	1253	1326	1328	1341
	1243	1356	1358	1373	1375	1399	1401	1427	1429	1456	1458	1483	1485	1515	1517
	1554	1556	1585	1587	1617	1619	1634	1636	1653	1655	1683	1683	1703	1742	1808
	1810	1829	1831	1989	2052	2096	2144	2222	2238	2273	2343	2346	2414	2443	2512
SWRSU	355#	574#													
TRMTRP	253#														
TYPBIN	355#														
TYPDEC	355#	1723													
TYPNAM	355#														
TYPNUM	355#														
TYPOCS	355#														
TYPOCT	355#	2108	2132	2364											
TYPTXT	355#	595	600	607	611	615	630	634	649	653	668	672	701	705	
SSCMRE	382#	421	422												
SSCMTM	382#														
SSESCA	355#														
SSNEWT	355#	759	767	777	789	800	811	825	841	859	877	896	912	927	940
	952	963	975	990	1008	1023	1038	1052	1085	1122	1147	1175	1251	1326	1341
	1356	1373	1399	1427	1456	1483	1515	1554	1585	1617	1634	1653	1681	1808	1829
SSSET	253#	2542	2543	2544	2545	2547	2549	2550	2551						

F06

DR11-K LOGIC TEST MAINDEC-11-DZDRG-C MACY11 27(732) 21-SEP-76 13:09 PAGE 73
 DZDRGC.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

ADD	737	739	741	744	746	748	1633	1691	1772	2116	2170	2180	2298	2380	2389
ASL	839	856	1119	1144	1173	1552	1583	2113	2114	2115	2403	2404	2405	2522	
ASLB	1777														
BCC	699	1084	1778												
BEQ	593	691	693	775	786	797	808	821	836	853	875	993	907	923	960
	972	985	995	1005	1020	1035	1050	1058	1061	1069	1079	1098	1104	1115	1140
	1169	1248	1324	1338	1351	1449	1490	1528	1592	1687	1727	1864	2018	2020	2022
	2026	2035	2066	2082	2085	2118	2123	2136	2197	2301	2336	2360	2387	2402	2471
BGE	2038														
BGT	1718	1786	2204	2399	2440										
BHI	2024														
BIC	849	1003	1015	1018	1030	1033	1045	1048	1066	1102	1110	1112	1162	1163	1181
	1182	1257	1258	1337	1503	1533	1544	1572	1605	1623	1715	1857	1880	2194	2356
	2373	2400	2427	2433	2441										
BIS	752	753	933	1076	1143	1172	1190	1195	1200	1205	1210	1215	1220	1225	1230
	1235	1240	1266	1271	1276	1281	1286	1291	1296	1301	1306	1311	1316	1410	1439
	1466	1532	1534	1545	1571	1573	1641	1660	1780	1781	1881	2199	2200	2407	
BISB	2105														
BIT	690	692	936	948	994	1060	1062	1073	1095	1097	1114	1131	1133	1156	1158
	1448	1489	1519	1527	1558	1566	1591	1688	2003	2017	2025	2032	2072	2091	
.TB	2324														
BLO	2466														
BLOS	2454														
BLT	1769	1795	2205	2315	2397	2438									
BMI	1366	1539	1776												
BNE	544	557	581	591	622	624	641	643	660	662	679	681	712	714	731
	924	840	857	910	926	937	949	1063	1074	1096	1120	1132	1134	1145	1157
	1159	1174	1520	1553	1559	1567	1584	1624	1627	1689	1774	1924	1851	1853	1858
	1984	2004	2033	2073	2089	2106	2128	2195	2244	2295	2303	2311	2325	2332	2352
	2358	2378	2385	2392	2429	2435	2458	2460	2476	2480	2490				
BPL	1549	1577	1760	1790	2078	2193	2289	2329	2354	2370	2425	2431			
BR	531	541	583	597	601	608	612	616	626	631	635	645	650	654	664
	669	673	683	702	705	716	734	1392	1422	1451	1478	1491	1506	1593	1600
	1670	1674	1693	1771	1788	1825	1843	1866	1873	2006	2012	2015	2028	2031	2111
	2138	2171	2186	2207	2236	2260	2291	2308	2318	2327	2334	2381	2408	2410	2436
	2469	2478	2484	2486											
CLR	530	537	550	555	569	570	727	751	756	758	763	764	765	782	817
	832	931	944	967	980	981	998	1000	1012	1027	1042	1113	1128	1137	1153
	1165	1184	1188	1193	1198	1203	1208	1213	1218	1223	1228	1233	1238	1260	1264
	1269	1274	1279	1284	1289	1294	1299	1304	1309	1314	1331	1334	1346	1361	1379
	1380	1391	1405	1419	1426	1433	1445	1446	1455	1461	1475	1481	1492	1496	1508
	1509	1511	1513	1524	1529	1531	1535	1546	1547	1564	1568	1570	1574	1594	1599
	1611	1613	1614	1615	1638	1649	1651	1652	1657	1666	1672	1677	1679	1680	1712
	1713	1763	1766	1815	1819	1834	1841	2030	2044	2104	2184	2242	2367	2368	2451
	2474														
CLRB	1792	2029	2307	2333	2481	2491									
CMP	542	556	580	623	642	661	680	713	730	774	796	807	820	835	852
	874	892	906	922	959	971	1004	1019	1034	1049	1068	1078	1103	1139	1168
	1247	1323	1348	1350	1363	1418	1424	1444	1453	1474	1479	1507	1609	1626	1647
	1671	1675	1784	1845	1863	2013	2037	2088	2351	2357	2377	2384	2396	2398	2428
	2434	2437	2439	2453	2465										
CMPB	2019	2023	2300	2302	2310	2331	2335	2359	2391	2457	2475	2479	2489		
COM	851	865	866	867	868	869	870	871	872	983	884	885	996	897	998
	889	890	1093	1191	1196	1201	1206	1211	1216	1221	1226	1231	1236	1241	1267
	1272	1277	1282	1287	1292	1297	1302	1307	1312	1317					
DEC	547	1692	1716	1823	1850	1852	1883	2112	2464						

.ASCIZ	427	599	603	610	614	618	633	637	652	656	671	675	704	708	1737
	1891	1899	1904	1909	1914	1919	1924	1928	1933	1942	1951	1957	1963	1969	1972
	2139	2262	2502	2503	2504	2506									
.BLKB	2501														
.BLKW	1906														
.BYTE	391	392	397	398	406	407	415	416	417	418	1736	1896	2214	2215	2216
	2217	2499	2500												
.DSABL	2411														
.ENABL	232	2344													
.END	2554														
.ENDC	237	249	341	355	363	365	366	367	379	385	389	391	419	423	424
	425	429	481	558	559	562	564	566	568	569	570	572	574	590	599
	603	610	614	618	623	629	633	637	642	648	652	656	661	667	671
	675	680	686	704	708	713	719	760	761	762	763	768	769	770	771
	776	778	779	780	781	782	787	790	791	792	793	798	801	802	803
	804	809	812	813	814	815	816	822	826	827	828	829	842	843	844
	845	850	861	852	863	876	878	879	880	881	894	897	898	899	900
	901	913	914	915	916	917	928	929	930	931	938	941	942	943	944
	950	953	954	955	956	961	964	965	966	967	973	976	977	978	979
	980	986	991	992	993	994	1006	1009	1010	1011	1012	1021	1024	1025	1026
	1027	1036	1039	1040	1041	1042	1051	1053	1054	1055	1056	1057	1059	1086	1087
	1088	1089	1105	1116	1123	1124	1125	1126	1141	1148	1149	1150	1151	1170	1176
	1177	1178	1179	1249	1252	1253	1254	1255	1325	1327	1328	1329	1330	1331	1339
	1342	1343	1344	1345	1352	1357	1358	1359	1360	1367	1374	1375	1376	1377	1378
	1393	1400	1401	1402	1403	1423	1428	1429	1430	1431	1450	1452	1457	1458	1459
	1460	1479	1484	1485	1486	1487	1492	1507	1516	1517	1518	1519	1521	1529	1540
	1550	1555	1556	1557	1558	1560	1568	1578	1586	1587	1588	1589	1594	1609	1618
	1619	1620	1621	1622	1635	1636	1637	1638	1654	1655	1656	1657	1675	1682	1683
	1684	1685	1686	1704	1706	1707	1709	1712	1718	1721	1722	1726	1728	1734	1736
	1737	1740	1743	1809	1810	1811	1812	1830	1831	1832	1833	1990	1993	1998	2003
	2005	2016	2019	2020	2021	2023	2025	2032	2036	2041	2042	2046	2049	2050	2053
	2056	2065	2069	2074	2075	2076	2077	2088	2092	2093	2097	2112	2141	2145	2223
	2232	2233	2239	2245	2246	2256	2258	2262	2274	2294	2344	2345	2347	2375	2411
	2415	2443	2444	2452	2454	2457	2485	2502	2508	2513	2519	2522	2541	2542	2543
	2544	2545	2546	2547	2548	2549	2550	2551	2552						
.EQUIV	249	250	258	273	274	303	304	305	306	307	308	309	310	311	312
	331	332	333	334	335	336	337	338	339	340					
.EVEN	599	603	610	614	618	633	637	652	656	671	675	704	708	1897	1980
	2140	2269	2508												
.IF	233	247	313	341	362	364	365	366	367	377	384	389	390	419	423
	424	425	428	429	481	553	558	560	562	564	566	568	569	570	572
	590	598	602	609	613	617	620	623	632	636	639	642	651	655	659
	661	670	674	677	680	703	707	710	713	759	761	763	767	769	771
	775	777	779	781	782	786	789	791	793	797	800	802	804	808	811
	813	815	816	821	825	827	829	841	843	845	859	861	863	875	877
	879	881	893	896	898	900	901	912	914	916	917	927	929	931	937
	940	942	944	949	952	954	956	960	963	965	967	972	975	977	979
	980	985	990	992	994	1005	1008	1010	1012	1020	1023	1025	1027	1035	1038
	1040	1042	1050	1052	1054	1056	1057	1058	1085	1087	1099	1104	1115	1122	1124
	1126	1140	1147	1149	1151	1169	1175	1177	1179	1248	1251	1253	1255	1324	1326
	1328	1330	1331	1338	1341	1343	1345	1351	1356	1358	1360	1366	1373	1375	1377
	1378	1392	1399	1401	1403	1422	1427	1429	1431	1449	1451	1456	1458	1460	1478
	1483	1485	1487	1491	1506	1515	1517	1519	1520	1528	1539	1549	1554	1556	1558
	1559	1567	1577	1585	1587	1589	1593	1608	1617	1619	1621	1622	1634	1636	1638
	1653	1655	1657	1674	1681	1683	1685	1686	1703	1704	1705	1706	1707	1708	1709
	1711	1717	1720	1722	1726	1728	1734	1736	1737	1742	1808	1810	1812	1829	1931

	1833	1989	1992	1997	2003	2015	2017	2018	2019	2021	2022	2023	2032	2034	2042
	2043	2048	2049	2050	2052	2055	2065	2068	2072	2074	2075	2077	2081	2088	2092
	2093	2096	2111	2127	2144	2222	2232	2233	2238	2245	2246	2254	2256	2258	2262
	2273	2294	2343	2345	2346	2347	2375	2414	2415	2443	2451	2453	2457	2458	2501
	2502	2508	2512	2518	2522	2533	2542	2543	2544	2545	2546	2547	2549	2550	2551
.IFF	247	362	364	366	367	385	388	390	419	429	558	760	761	762	763
	768	769	770	771	776	778	779	780	781	787	790	791	792	793	798
	801	802	803	804	809	812	813	814	815	821	826	827	828	829	842
	843	844	845	860	861	862	863	876	878	879	980	881	894	897	898
	899	900	913	914	915	916	928	929	930	931	938	941	942	943	944
	950	953	954	955	956	961	964	965	966	967	973	976	977	978	979
	996	991	992	993	994	1006	1009	1010	1011	1012	1021	1024	1025	1026	1027
	1036	1039	1040	1041	1042	1051	1053	1054	1055	1056	1057	1059	1086	1087	1088
	1089	1104	1115	1123	1124	1125	1126	1140	1148	1149	1150	1151	1169	1176	1177
	1178	1179	1249	1252	1253	1254	1255	1325	1327	1328	1329	1330	1339	1342	1343
	1344	1345	1352	1357	1358	1359	1360	1367	1374	1375	1376	1377	1393	1400	1401
	1402	1403	1423	1429	1429	1430	1431	1450	1452	1457	1458	1459	1460	1479	1484
	1485	1486	1487	1492	1507	1516	1517	1518	1519	1521	1528	1539	1549	1555	1556
	1557	1558	1560	1567	1577	1586	1587	1588	1589	1594	1609	1618	1619	1620	1621
	1622	1635	1636	1637	1638	1654	1655	1656	1657	1675	1682	1683	1684	1685	1686
	1704	1708	1712	1717	1720	1736	1743	1809	1810	1811	1812	1830	1831	1832	1833
	1990	2016	2019	2020	2023	2049	2050	2053	2055	2069	2088	2093	2097	2112	2141
	2145	2223	2239	2254	2274	2344	2347	2415	2417	2422	2443	2444	2453	2485	2501
	2513	2519													
.IFT	599	603	610	614	618	633	637	652	656	671	675	704	708	2031	2075
	2417	2422													
.IFTF	599	603	610	614	618	633	637	652	656	671	675	704	708	2029	2074
	2362	2415	2418												
.IIF	232	237	242	359	360	361	363	366	367	374	428	559	562	568	569
	570	572	573	1706	1712	1713	1724	1736	1740	1993	1994	1995	1996	1997	1998
	2002	2030	2031	2046	2049	2050	2056	2057	2058	2059	2064	2080	2088	2093	2109
	2134	2341	2344	2365	2493	2502	2508	2541	2542	2543	2544	2545	2547	2549	2550
	2551														
.IRP	481	759	767	777	789	800	811	825	841	859	877	896	912	927	940
	952	963	975	990	1008	1023	1038	1052	1085	1122	1147	1175	1251	1326	1341
	1356	1373	1399	1427	1456	1483	1515	1554	1585	1617	1634	1653	1681	1753	1793
	1808	1829	2226	2232	2245	2246									
.LIST	232	355	366	374	419	421	422	423	481	574	599	603	610	614	618
	633	637	652	656	671	675	704	708	759	763	767	771	777	781	789
	793	800	804	811	815	825	829	841	845	859	863	877	881	896	900
	912	916	927	931	940	944	952	956	963	967	975	979	990	994	1008
	1012	1023	1027	1038	1042	1052	1056	1085	1089	1122	1126	1147	1151	1175	1179
	1251	1255	1326	1330	1341	1345	1356	1360	1373	1377	1399	1403	1427	1431	1456
	1460	1483	1487	1515	1519	1554	1558	1585	1589	1617	1621	1634	1638	1653	1657
	1681	1685	1712	1728	1808	1812	1829	1833	1997	2088	2443	2533	2541	2542	2543
	2544	2545	2546	2547	2548	2549	2550	2551	2552						
.MACRO	367	382	527	989	990	2533									
.MCALL	232	355	574												
.NLIST	232	355	366	374	419	421	422	423	481	574	599	603	610	614	618
	633	637	652	656	671	675	704	708	759	763	767	771	777	781	789
	793	800	804	811	815	825	829	841	845	859	863	877	881	896	900
	912	916	927	931	940	944	952	956	963	967	975	979	990	994	1008
	1012	1023	1027	1038	1042	1052	1056	1085	1089	1122	1126	1147	1151	1175	1179
	1251	1255	1326	1330	1341	1345	1356	1360	1373	1377	1399	1403	1427	1431	1456
	1460	1483	1487	1515	1519	1554	1558	1585	1589	1617	1621	1634	1638	1653	1657

	1681	1685	1712	1728	1808	1812	1829	1833	1997	2089	2443	2533	2541	2542	2543
.PAGE	2544	2545	2546	2547	2548	2549	2550	2551	2552						
.REM	355	382	429												
.REPT	1														
.SBTTL	374	421	865	883	1187	1263									
	245	355	368	377	382	429	552	620	639	658	677	688	710	759	767
	777	789	800	811	825	841	859	877	896	912	927	940	952	963	975
	990	1008	1023	1038	1052	1085	1122	1147	1175	1251	1326	1341	1356	1373	1399
.TITLE	1427	1456	1483	1515	1554	1585	1617	1634	1653	1681	1701	1740	1808	1829	1987
.WORD	2050	2094	2142	2220	2271	2341	2510	2533							
	232														
	374	375	376	390	393	394	395	396	399	400	401	402	403	404	405
	409	409	410	419	421	422	606	1717	1720	1735	2120	2125	2218	2255	2257
	2338	2540													

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

* DZDRGC.SEG/SOL/CRF=DZDRGC.P11
 RUN-TIME: 50 31 6 SECONDS
 RUN-TIME RATIO: 317'88=3.5
 CORE USED: 24K (47 PAGES)

