

AD11-K

PERFORMANCE TEST
MD-11-DZADL-B

EP-DZADL-B-DL-B
COPYRIGHT © 1976
FICHE 1 OF 1

DEC 1976
digital
MADE IN USA

This microfiche card contains a grid of frames. The first column of frames contains a series of small, illegible text labels. The subsequent columns contain data, including what appears to be a performance test table with multiple rows and columns of numbers and text. The right half of the card is a large, dark, mostly blank area, possibly representing a larger frame or a section of the original document that is not clearly legible in this scan.

107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160

2.2 STORAGE

THIS PROGRAM USES ALL BK OF MEMORY AND IS NOT "CHAINABLE" ON AN BK CPU. THE PROGRAM IS "CHAINABLE" ON 12K OR GREATER. THE PROGRAM WILL DESTROY "ABSOLUTE LOADER" ON AN BK CPU, IF "W" OR "A" IS SELECTED.

3.0 LOADING PROCEDURE

PROCEDURE FOR LOADING NORMAL BINARY TAPES SHOULD BE FOLLOWED.

4.0 STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

STANDARD PDP-11 FORMAT

SW15=1 HALT ON ERROR
SW14=1 LOOP ON TEST
SW13=1 INHIBIT ERROR TYPEOUTS
SW12=1 HALT FOR VTSS DISPLAY
SW11=1 INHIBIT ITERATIONS
SW10=1 BELL ON ERROR
SW9 =1 LOOP ON ERROR
SW8 =1 LOOP ON TEST IN SWR <7:0>

200 IS THE STARTING ADDRESS OF THE DIAGNOSTIC FOR STANDARD TOLERANCES. 204 IS THE RESTART ADDRESS. 210 IS THE STARTING ADDRESS OF THE DIAGNOSTIC FOR THE OPTION TEST AREA'S TIGHTER TOLERANCES.

5.0 OPERATING PROCEDURE

START THE DIAGNOSTIC AT 200 OR 210. THE PROGRAM HEADING AND THE LIST OF TESTS AVAILABLE, WILL BE PRINTED OUT FOLLOWED BY A MESSAGE "TYPE THE LETTER AND CARRIAGE RETURN FOR THE DESIRED

E01

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 4

161
162

TEST:". THEN TYPE THE LETTER YOU WANT, ACCORDING TO THE TABLE LISTED AND HIT CARRIAGE RETURN.

163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203

TWO CONTROL CHARACTERS, ↑A AND ↑C, ARE SET ASIDE FOR INTERRUPTING A TEST AND TRANSFERRING CONTROL TO EITHER THE BEGINNING OF THE DIAGNOSTIC (↑C) OR TO THE BEGINNING OF THE SPECIFIC TEST WHICH WAS IN PROGRESS (↑A). DURING THE LOGIC TESTS WHILE A RESET IS BEING PERFORMED, ↑C OR ↑A WILL NOT BE EXECUTED UNTIL AFTER THE RESET HAS BEEN COMPLETED, THEREFORE HIT ↑C OR ↑A UNTIL IT IS SUCCESSFUL.

FOR MACHINES WITHOUT A HARDWARE SWITCH REGISTER, LOCATION SWREG (176) IS USED AS A SOFTWARE SWITCH REGISTER. TO MODIFY THE CONTENTS OF SWREG, TYPE ↑G. THE PROGRAM RESPONDS WITH THE CURRENT CONTENTS OF SWREG AND A SLASH. TYPE THE DESIRED NEW CONTENTS OF SWREG FOLLOWED BY A CARRIAGE RETURN.

IF "W" IS TYPED, THE PROGRAM WILL TYPE "XX AD11K'S FOUND". WHERE XX IS THE NUMBER OF AD11K'S IN OCTAL. IF THE NUMBER IS GREATER THAN 1, THE TEST WILL BE RUN SUCCESSIVELY ON EACH AD11K. THE PROGRAM WILL RUN THROUGH THE LOGIC SUBTESTS, THE NOISE TEST ON 8 EDGES, THE INTERCHANNEL SETTLING TEST ON 8 EDGES, AND THE DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST. A G5036 WRAPAROUND MODULE IS REQUIRED. THE PROGRAM SUPPORTS AD11K EXPANSION BEYOND 16. CHANNELS. TO RUN THIS TEST ON A GROUP OF CHANNELS OTHER THAN 0-17, LOAD 20, 40, OR 60 INTO LOCATION BASECH (1336) FOR CHANNELS 20-37, 40-57, 60-77.

IF "C" IS TYPED, THE PROGRAM WILL RUN THE CALIBRATION TEST AND WILL LOOP ON THAT TEST UNTIL THE OPERATOR HALTS IT. IF A CERTAIN AD11K IS TO BE TESTED, ITS STATUS REGISTER ADDRESS MUST BE LOADED INTO \$BASE (1250), AND ITS VECTOR ADDRESS MUST BE LOADED INTO THE LOW BYTE OF \$VECT1 (1244) (THE HIGH BYTE CONTAINING THE PRIORITY).

IF "N" IS TYPED, THE PROGRAM WILL RUN THE NOISE TEST TAGGED "BEGINN" AND WILL LOOP ON THIS TEST UNTIL THE OPERATOR HALTS IT. IF A CERTAIN AD11K IS TO BE TESTED ITS STATUS REGISTER ADDRESS MUST BE LOADED INTO \$BASE (1250), AND ITS VECTOR ADDRESS MUST BE LOADED INTO THE LOW BYTE OF \$VECT1 (1244) (THE HIGH BYTE CONTAINING THE PRIORITY).

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

IF "S" IS TYPED, THE PROGRAM WILL RUN THE INTERCHANNEL SETTling TEST TAGGED "BEGINS" AND WILL LOOP ON THIS TEST UNTIL THE OPERATOR HALTS IT. AT THE BEGINNING OF THIS TEST, THE OPERATOR MUST RESPOND TO THE STATEMENTS ASKING FOR THE "FROM" CHANNEL AND THE "TO" CHANNEL BY TYPING IN THE CHANNEL VALUE IN OCTAL AND HITTING CARRIAGE RETURN. IF A CERTAIN AD11K IS TO BE TESTED ITS STATUS REGISTER ADDRESS MUST BE LOADED INTO \$BASE (1250), AND ITS VECTOR ADDRESS MUST BE LOADED INTO \$VECT1 (1244) (THE HIGH BYTE CONTAINING THE PRIORITY).

IF "A" IS TYPED, THE PROGRAM WILL EXECUTE THE LOGIC TESTS, ANALOG TESTS, NOISE, SETTLE AND DIFFERENTIAL LINEARITY. AT THE BEGINNING OF THE TEST THE PROGRAM WILL TYPE "XX AD11K'S FOUND". WHERE XX IS THE NUMBER OF AD11K'S IN OCTAL IF THE NUMBER IS GREATER THAN 1, THE TEST WILL BE RUN SUCCESSIVELY ON EACH AD11K. THE PROGRAM SUPPORTS AD11K EXPANSION BEYOND 16. CHANNELS. TO RUN THIS TEST ON A GROUP OF CHANNELS OTHER THAN 0-17, LOAD 20,40, OR 60 INTO LOCATION BASECH (1336) FOR CHANNELS 20-37, 40-57, 60-77.

IF "L" IS TYPED, THE PROGRAM WILL EXECUTE THE LOGIC TESTS, PRINTING "END PASS" WHEN IT HAS COMPLETED AN ENTIRE PASS. AT THE BEGINNING OF THE TEST THE PROGRAM WILL TYPE "XX AD11K'S FOUND". WHERE XX IS THE NUMBER OF AD11K'S IN OCTAL IF THE NUMBER IS GREATER THAN 1, THE TEST WILL BE RUN SUCCESSIVELY ON EACH AD11K.

B

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

6.0 ERRORS

THIS PROGRAM USES THE DIAGNOSTIC "SYSMAC" PACKAGE FOR ERROR REPORTING AND TYPEOUT. THE ERROR INFORMATION CONSISTS OF THE FOLLOWING:

ERRPC: LOCATION AT WHICH AN ERROR WAS DETECTED.
STREG: ADDRESS OF THE STATUS REGISTER.
ADBUFF: ADDRESS OF THE BUFFER
CHANL: CHANNEL VALUE
NOMINAL: EXPECTED CORRECT DATA
TOLERANCE: THE ACCEPTABLE DEVIATION FROM THE NOMINAL
ACTUAL: ACTUAL DATA
EXPECTED: EXPECTED CORRECT DATA

7.0 MISCELLANEOUS

7.1 EXECUTION TIME

EXECUTION TIME FOR EACH OF THE TESTS IS:

CALIBRATION: 8 CONVERSIONS/5 SECONDS @ 110 BAUD
WRAPAROUND TEST: 17 MINUTES FIRST PASS; 35 MINUTES
FOR SUCCESSIVE PASSES
SETTLING TEST: 1 MINUTE
NOISE TEST: 1 MINUTE
LOGIC TEST: 1 MINUTE
AUTO TEST: 18 MINUTES FIRST PASS, 36 MINUTES
FOR SUCCESSIVE PASSES

7.2 STATUS REGISTER AND VECTOR ADDRESSES AND PRIORITY

WHEN TESTING MORE THAN ONE AD11K, THE DIFFERENCE IN ADDRESSES IS PRESENTLY 40 FOR BUS ADDRESS AND VECTOR ADDRESS. THESE VALUES ARE IN VADR (BUS ADDRESS) (1332) AND VVCT (VECTOR ADDRESS) (1334). THE FIRST AD11K'S STATUS REGISTER ADDRESS MUST BE IN \$BASE (1250), ITS VECTOR ADDRESS MUST BE IN THE LOW BYTE OF \$VECT1 (1244), AND THE PRIORITY MUST BE IN THE HIGH BYTE OF \$VECT1.

7.3 AD11K PRIORITY

IF AD11K IS SET FOR A PRIORITY OTHER THAN 6, THE HIGH BYTE OF \$VECT1 (1244) MUST BE ADJUSTED ACCORDINGLY (THE LOW BYTE

I01

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 8

286
287

CONTAINING THE VECTOR ADDRESS). IF MORE THAN ONE AD11K IS BEING
TESTED, ALL MUST BE SET AT THE SAME PRIORITY.

288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341

7.4 SWITCH REGISTER

IF A HARDWARE SWITCH REGISTER IS PRESENT AND THE OPERATOR DESIRES TO USE A SOFTWARE SWITCH REGISTER AND THE IG FEATURE; IT IS NECESSARY TO LOAD THE STARTING ADDRESS, SET THE 'HARDWARE' SWITCH REGISTER TO ALL ONES (-1), AND HIT START. THE PROGRAM WILL THEN RUN WITH THE SOFTWARE SWITCH REGISTER.

7.5 VT55 GRAPHIC OUTPUT

THE SCREEN DISPLAY MAY BE HALTED FOR EXAMINATION BY SETTING BIT 12. AND THEN JUST HIT CONTINUE TO COMPLETE THE PROGRAM'S EXECUTION.

8.0 RESTRICTIONS

8.1 A G5036 WRAPAROUND MODULE MUST BE PRESENT WHEN RUNNING THE AUTO TEST AND THE WRAPAROUND TEST.

SWITCH ON G5036 MUST BE IN '0' POSITION.

THE WRAPAROUND (G5036) MODULE MUST BE CONNECTED AS FOLLOWS:
AD11K TO BCOBR CONNECTION A-A, VV-VV
BCOBR TO G5036 CONNECTION "UPSIDE-DOWN" A-VV, VV-A

9.0 PROGRAM DESCRIPTION

9.1 LOGIC TESTS

THESE 14 LOGIC SUBTESTS RUN SEQUENTIALLY WITHOUT FURTHER OPERATOR INTERVENTION AFTER HE/SHE HAS TYPED IN THE NUMBER OF AD11K'S TO BE TESTED. ITS PURPOSE IS TO CHECK THAT EACH OF THE MUX BITS CAN BE LOADED AND PROPERLY READ BACK; THAT INITIALIZE CLEARS THE EXTERNAL START ENABLE BIT, THE DONE BIT, THE INTERRUPT ENABLE BIT, THE OVERFLOW BIT, THE ERROR FLAG, AND THE A/D START BIT. IT ALSO CHECKS THAT THE A/D DONE FLAG SETS AT END OF CONVERSION AND CLEARS WHEN THE CONVERTED VALUE IS READ. IT CHECKS THE INTERRUPT

K01

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 10

342

LOGIC AND THE CORRECT SETTING OF THE ERROR FLAG.

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389

9.2 CALIBRATION TEST

THIS TEST BEGINS WHEN THE OPERATOR TYPES "C". IT THEN LOADS THE CHANNEL FROM THE SWITCH REGISTER BITS 0-7 AND DOES A CONVERSION ON THAT CHANNEL. IF SWR BIT 13 IS DOWN, IT PRINTS OUT THE CONVERTED VALUE ON THE TELETYPE; OTHERWISE, IF SWR BIT 13 IS UP, IT PUTS THE CONVERTED VALUE IN THE DISPLAY REGISTER. THE OPERATOR MAY CHANGE THE CHANNEL AT ANY TIME DURING THE TEST, HOWEVER THE NEW VALUES FROM THE NEW CHANNEL WILL NOT BE PRINTED UNTIL THE NEXT LINE OF 8 VALUES IS PRINTED. THE 8 VALUES ON EACH LINE CORRESPOND TO ONLY ONE CHANNEL.

9.3 DIFFERENTIAL LINEARITY

THIS TEST IS TO DETERMINE IF A CHANGE IN THE INPUT VOLTAGE REPRESENTS A SIMILAR CHANGE IN THE RESULTING CONVERTED BINARY VALUE.

9.4 SETTling TEST

THE PURPOSE OF THIS TEST IS TO CHECK THAT THE TIME NEEDED TO SETTLE AND CORRECTLY REPORT A NEW INPUT VALUE AFTER SWITCHING CHANNELS DOES NOT EXCEED THE EXPECTED AMOUNT OF TIME FOR SUCH A CHANGE.

9.5 NOISE TEST

THIS TEST MEASURES THE INTERNAL SHORT-TERM REPEATABILITY NOISE WITHIN THE A/D. RMS NOISE EQUALS 1 STANDARD DEVIATION OF THE GAUSSIAN CURVE. PEAK NOISE EQUALS 2.3 STANDARD DEVIATION OF THE GAUSSIAN CURVE.

9.6 ANALOG TESTS

THESE 11 SUBTESTS CHECK THE CHANNELS AND THEIR OUTPUT.
#

```

390      .TITLE MAINDEC-11-DZADL-B
391      ;*COPYRIGHT (C) 1976
392      ;*DIGITAL EQUIPMENT CORP.
393      ;*MAYNARD, MASS. 01754
394      ;*
395      ;*PROGRAM BY VERA BREUER
396      ;*
397      ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
398      ;*PACKAGE (MAINDEC-11-DZQAC-CD),MAR 21, 1976.
399      ;*
400      .SBTTL BASIC DEFINITIONS
401
402      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
403      STACK= 1100
404      001100
405      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
406      .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
407
408      ;*MISCELLANEOUS DEFINITIONS
409      HT= 11                ;;CODE FOR HORIZONTAL TAB
410      LF= 12                ;;CODE FOR LINE FEED
411      CR= 15                ;;CODE FOR CARRIAGE RETURN
412      CRLF= 200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
413      PS= 177776           ;;PROCESSOR STATUS WORD
414      .EQUIV PS,PSW
415      STKLMT= 177774       ;;STACK LIMIT REGISTER
416      PIRQ= 177772        ;;PROGRAM INTERRUPT REQUEST REGISTER
417      DSWR= 177570        ;;HARDWARE SWITCH REGISTER
418      DDISP= 177570       ;;HARDWARE DISPLAY REGISTER
419
420      ;*GENERAL PURPOSE REGISTER DEFINITIONS
421      R0= %0                ;;GENERAL REGISTER
422      R1= %1                ;;GENERAL REGISTER
423      R2= %2                ;;GENERAL REGISTER
424      R3= %3                ;;GENERAL REGISTER
425      R4= %4                ;;GENERAL REGISTER
426      R5= %5                ;;GENERAL REGISTER
427      R6= %6                ;;GENERAL REGISTER
428      R7= %7                ;;GENERAL REGISTER
429      .EQUIV R6,SP          ;;STACK POINTER
430      .EQUIV R7,PC          ;;PROGRAM COUNTER
431
432      ;*PRIORITY LEVEL DEFINITIONS
433      PR0= 0                ;;PRIORITY LEVEL 0
434      PR1= 40               ;;PRIORITY LEVEL 1
435      PR2= 100              ;;PRIORITY LEVEL 2
436      PR3= 140              ;;PRIORITY LEVEL 3
437      PR4= 200              ;;PRIORITY LEVEL 4
438      PR5= 240              ;;PRIORITY LEVEL 5
439      PR6= 300              ;;PRIORITY LEVEL 6
440      PR7= 340              ;;PRIORITY LEVEL 7
441
442      ;*"SWITCH REGISTER" SWITCH DEFINITIONS
443      SW15= 100000
444      SW14= 40000

```

444 020000
 445 010000
 446 004000
 447 002000
 448 001000
 449 000400
 450 000200
 451 000100
 452 000040
 453 000020
 454 000010
 455 000004
 456 000002
 457 000001
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470 100000
 471 040000
 472 020000
 473 010000
 474 004000
 475 002000
 476 001000
 477 000400
 478 000200
 479 000100
 480 000040
 481 000020
 482 000010
 483 000004
 484 000002
 485 000001
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497

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

```

498 000004 ERRVEC= 4 :: TIME OUT AND OTHER ERRORS
499 000010 RESVEC= 10 :: RESERVED AND ILLEGAL INSTRUCTIONS
500 000014 TBITVEC= 14 :: "T" BIT
501 000014 TRIVEC= 14 :: TRACE TRAP
502 000014 BPTVEC= 14 :: BREAKPOINT TRAP (BPT)
503 000020 IOTVEC= 20 :: INPUT/OUTPUT TRAP (IOT) **SCOPE**
504 000024 PWRVEC= 24 :: POWER FAIL
505 000030 EMTVEC= 30 :: EMULATOR TRAP (EMT) **ERROR**
506 000034 TRAPVEC= 34 :: "TRAP" TRAP
507 000060 TKVEC= 60 :: TTY KEYBOARD VECTOR
508 000064 TPVEC= 64 :: TTY PRINTER VECTOR
509 000240 PIRQVEC= 240 :: PROGRAM INTERRUPT REQUEST VECTOR
510 .SBTTL OPERATIONAL SWITCH SETTINGS
511
512 *
513 * SWITCH USE
514 * -----
515 * 15 HALT ON ERROR
516 * 14 LOOP ON TEST
517 * 13 INHIBIT ERROR TYPEOUTS
518 * 12 HALT FOR VTSS DISPLAY
519 * 11 INHIBIT ITERATIONS
520 * 10 BELL ON ERROR
521 * 9 LOOP ON ERROR
522 * 8 LOOP ON TEST IN SWR<7:0>
523
524 170400 ABASE= 170400
525 140340 AVECT1= 140340
526 000300 APRIOR= 300
527
528 .SBTTL TRAP CATCHER
529
530 000000 .=0
531
532 *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
533 *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
534 *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
535
536 000174 .=174
537 000174 000000 DISPREG: .WORD 0 :: SOFTWARE DISPLAY REGISTER
538 000176 000000 SWREG: .WORD 0 :: SOFTWARE SWITCH REGISTER
539
540 .SBTTL STARTING ADDRESS(ES)
541 000200 000137 001620 JMP @#BEGIN ;; JUMP TO STARTING ADDRESS OF PROGRAM
542 000204 000137 002256 JMP @#BEG2 ;RESTART ADDRESS
543 000210 000137 001626 JMP @#BEGIN2 ;START ADDRESS FOR OPTION TEST AREA
    
```

539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571

000214
000046
011776
000052
000000
000214
001000

001000
000024
000200
000044
001000
001000

001000
001000
000000
001002
001174
001004
002260
001006
000764
001010
003244
001012
000031

```
.SBTTL ACT11 HOOKS
:*****
:HOOKS REQUIRED BY ACT11
    $SVPC=.          ;SAVE PC
    =46              ;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
    $ENDAD           ;
    =52              ;2)SET LOC.52 TO ZERO
    .WORD 0          ;
    =$SVPC           ; RESTORE PC
.=1000
.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
    .SX=.           ;SAVE CURRENT LOCATION
    =24             ;SET POWER FAIL TO POINT TO START OF PROGRAM
    200             ;FOR APT START UP
    =44             ;POINT TO APT INDIRECT ADDRESS PNTR.
    $APTHDR         ;POINT TO APT HEADER BLOCK
    =.$X            ;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
$APTHD:
$SHIBTS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MADR: .WORD $MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 1200. ;RUN TIM OF LONGEST TEST
$PASTM: .WORD 500. ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 1700. ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)
```


572
573
574
575
576
577
578 001100
579 001100 001100
580 001100 000000
581 001102 000
582 001103 000
583 001104 000000
584 001106 000000
585 001110 000000
586 001112 000000
587 001114 000
588 001115 001
589 001116 000000
590 001120 000000
591 001122 000000
592 001124 000000
593 001126 000000
594 001130 000000
595 001132 000000
596 001134 000
597 001135 000
598 001136 000000
599 001140 177570
600 001142 177570
601 001144 177560
602 001146 177562
603 001150 177564
604 001152 177566
605 001154 000
606 001155 002
607 001156 012
608 001157 000
609 001160 000000
610 001162 000000
611 001164 177607 000377
612 001170 077
613 001171 015
614 001172 000012
615
616
617
618
619
620 001174
621 001174 000000
622 001176 000000
623 001200 000000
624 001202 000000
625 001204 000000

.SBTTL COMMON TAGS

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

SCMTAG: . =1100 ;: START OF COMMON TAGS
 .WORD 0
 STSTNM: .BYTE 00 ;: CONTAINS THE TEST NUMBER
 SERFLG: .BYTE 00 ;: CONTAINS ERROR FLAG
 SICNT: .WORD 00 ;: CONTAINS SUBTEST ITERATION COUNT
 SLPADR: .WORD 00 ;: CONTAINS SCOPE LOOP ADDRESS
 SLPERR: .WORD 00 ;: CONTAINS SCOPE RETURN FOR ERRORS
 SERTTL: .WORD 00 ;: CONTAINS TOTAL ERRORS DETECTED
 SITEMB: .BYTE 00 ;: CONTAINS ITEM CONTROL BYTE
 SERMAX: .BYTE 01 ;: CONTAINS MAX. ERRORS PER TEST
 SERRPC: .WORD 00 ;: CONTAINS PC OF LAST ERROR INSTRUCTION
 SGDADR: .WORD 00 ;: CONTAINS ADDRESS OF 'GOOD' DATA
 SBDADR: .WORD 00 ;: CONTAINS ADDRESS OF 'BAD' DATA
 SGDDAT: .WORD 00 ;: CONTAINS 'GOOD' DATA
 SBDDAT: .WORD 00 ;: CONTAINS 'BAD' DATA
 .WORD 00 ;: RESERVED--NOT TO BE USED
 SAUTOB: .BYTE 00 ;: AUTOMATIC MODE INDICATOR
 SINTAG: .BYTE 00 ;: INTERRUPT MODE INDICATOR
 .WORD 0
 SWR: .WORD DSWR ;: ADDRESS OF SWITCH REGISTER
 DISPLAY: .WORD DDISP ;: ADDRESS OF DISPLAY REGISTER
 STKS: 177560 ;: TTY KBD STATUS
 STKB: 177562 ;: TTY KBD BUFFER
 STPS: 177564 ;: TTY PRINTER STATUS REG. ADDRESS
 STPB: 177566 ;: TTY PRINTER BUFFER REG. ADDRESS
 \$NULL: .BYTE 0 ;: CONTAINS NULL CHARACTER FOR FILLS
 \$FILLS: .BYTE 2 ;: CONTAINS # OF FILLER CHARACTERS REQUIRED
 \$FILLC: .BYTE 12 ;: INSERT FILL CHARS. AFTER A "LINE FEED"
 \$TPFLG: .BYTE 0 ;: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
 \$TIMES: 0 ;: MAX. NUMBER OF ITERATIONS
 \$ESCAPE: 0 ;: ESCAPE ON ERROR ADDRESS
 \$BELL: .ASCIZ <207><377><377> ;: CODE FOR BELL
 \$QUES: .ASCII /?/ ;: QUESTION MARK
 \$CRLF: .ASCII <15> ;: CARRIAGE RETURN
 \$LF: .ASCIZ <12> ;: LINE FEED

.SBTTL APT MAILBOX-ETABLE

::*****
 .EVEN
 \$MAIL: ;: APT MAILBOX
 \$MSGTY: .WORD AMSGTY ;: MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL ;: FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN ;: TEST NUMBER
 \$PASS: .WORD APASS ;: PASS COUNT
 \$DEVCT: .WORD ADEVCT ;: DEVICE COUNT

626	001206	000000	\$UNIT: .WORD	AUNIT	:: I/O UNIT NUMBER
627	001210	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
628	001212	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
629	001214		\$ETABLE:		:: APT ENVIRONMENT TABLE
630	001214	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
631	001215	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
632	001216	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
633	001220	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
634	001222	000000	\$CPUOP: .WORD	ACPUOP	:: CPU TYPE, OPTIONS
635			:: *		BITS 15-11=CPU TYPE
636			:: *		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
637			:: *		11/70=06, PDQ=07, Q=10
638			:: *		BIT 10=REAL TIME CLOCK
639			:: *		BIT 9=FLOATING POINT PROCESSOR
640			:: *		BIT 8=MEMORY MANAGEMENT
641	001224	000	\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
642	001225	000	\$MTYP1: .BYTE	AMTYP1	:: MEM. TYPE, BLK#1
643			:: *		MEM. TYPE BYTE -- (HIGH BYTE)
644			:: *		900 NSEC CORE=001
645			:: *		300 NSEC BIPOLAR=002
646			:: *		500 NSEC MOS=003
647	001226	000000	\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
648			:: *		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
649	001230	000	\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
650	001231	000	\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
651	001232	000000	\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
652	001234	000	\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
653	001235	000	\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
654	001236	000000	\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
655	001240	000	\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
656	001241	000	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
657	001242	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
658	001244	140340	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
659	001246	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
660	001250	170400	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
661	001252	000000	\$DEVN: .WORD	ADEVN	:: DEVICE MAP
662	001254	000000	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
663	001256		\$ETEND:		
664			.MEXIT		

665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708

.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 REG. ERROR
	DH1	: ERRPC STREG EXPECTED ACTUAL
	DT1	: \$ERRPC, STREG, \$GDDAT, \$BDDAT
	DF1	
: ITEM 2	EM2	: FAILED TO INTERRUPT
	DH3	: ERRPC STREG ACTUAL
	DT3	: \$ERRPC, STREG, \$BDDAT
	DF1	
: ITEM 3	EM3	: UNEXPECTED INTERRUPT
	DH3	: ERRPC STREG
	DT3	: \$ERRPC, STREG
	DF1	
: ITEM 4	EM4	: ERROR ON A/D CHANNEL
	DH2	: ERRPC STREG CHAN NOMINAL TOL ACTUAL
	DT2	: \$ERRPC, STREG, CHANL, \$GDDAT, SPREAD, \$BDDAT
	DF1	

001256		
001256	014146	
001260	014306	
001262	014472	
001264	014532	
001266	014174	
001270	014427	
001272	014522	
001274	014532	
001276	014224	
001300	014427	
001302	014522	
001304	014532	
001306	014255	
001310	014344	
001312	014504	
001314	014532	

Address	Offset	Value	Label	Description	
709			.SBTTL	MISCELLANEOUS, TEMPORARY, AND STORAGE LOCATIONS	
710	001316	170400	STREG:	ABASE ; ADDRESS OF STATUS REGISTER	
711	001320	170401	ADST1:	ABASE+1 ; UPPER BYTE OF STATUS REG.	
712	001322	170402	ADBUFF:	ABASE+2 ; ADDRESS OF A/D BUFFER	
713	001324	140340	VECTOR:	AVECT1 ; VECTOR ADDRESS	
714	001326	000300	BASEBR:	APRIOR ; INTERRUPT PRIORITY LEVEL	
715	001330	140342	VECTR1:	AVECT1+2	
716	001332	000040	VADR:	40 ; INCREMENT FOR BUS ADDRESS	
717	001334	000040	VVCT:	40 ; INCREMENT FOR VECTOR ADDRESS	
718	001336	000000	BASECH:	0 ; BASE CHANNEL	
719	001340	000060	KBVECT:	60	
720	001342	000000	WIDE:	0 ; NO. OF WIDE STATES	
721	001344	000000	NARROW:	0 ; NO. OF NARROW STATES	
722	001346	000000	FIRST:	0	
723	001350	000000	SKIPST:	0 ; NO. OF SKIPPED STATES	
724	001352	000000	TEMP:	0 ; WORK AREA	
725	001354	000000	CH1:	0 ; FIRST CHANNEL	
726	001356	000000	CH2:	0 ; SECOND CHANNEL	
727	001360	000000	NBEXT:	0 ; NO. OF AD11K'S TO BE TESTED	
728	001362	000000	NMBEXT:	0 ; NO. OF AD11K'S TO BE TESTED	
729	001364	000000	DUMMY:	0 ; DUMMY CHANNEL	
730	001366	000000	CHANL:	0 ; CHANNEL VALUE	
731	001370	000000	TADDR:	0 ; TEST ADDRESS	
732	001372	000000	RNA:	0 ; RANDOM	
733	001374	000000	RNB:	0 ; NUMBER	
734	001376	000000	RNC:	0 ; VALUES	
735	001400	000000	RMS:	0 ; RMS NOISE VALUE	
736	001402	000000	PEAK:	0 ; PEAK NOISE VALUE	
737	001404	000000	FLAG:	0 ; VTSS FLAG	
738	001406	000000	SPREAD:	0 ; DEVIATION FROM THE NOMINAL	
739	001410	000000	DAC:	0 ; SAR VALUE	
740	001412	000000	DELAY:	0 ; TIME DELAY COUNTER	
741	001414	000000	EDGE:	0 ; EDGE VALUE	
742	001416	000000	BITPNT:	0	
743	001420	000000	MIN:	0 ; MIN VALUE	
744	001422	000000	WFTST:	0 ; OPTION TEST AREA FLAG	
745	001424	000000	MAX:	0 ; MAX VALUE	
746	001426	000000	PERCNT:	0 ; PERCENT FOR SAR ROUTINE	
747	001430	000000	OUT:	0	
748					
749	001432		UNEXP:		
750	001432	012737	001446	001162	MOV #1\$, \$ESCAPE ; ; ESCAPE TO 1\$ ON ERROR
751	001440	005237	001103		INC \$ERFLG
752	001444	104003			ERROR 3
753	001446	005037	001162		1\$: CLR \$ESCAPE ; RETURN ESCAPE TO NORMAL
754	001452	000002			RTI ; UNEXPECTED INTERRUPT

```

755          .SBTTL      CONTROL A AND C DECODERS
756 001454 010046          ISERV:  MOV      RO, -(SP)          ;SAVE RO
757 001456 017700 177464  MOV      @STKB,RO      ;GET CHARACTER
758 001462 042700 177600  BIC      #177600,RO
759 001466 120027 000003  CMPB    RO,#3          ;IS IT ↑C?
760 001472 001010          BNE     1$
761 001474 104400 012146  TYPE    ,CMMSG        ;ECHO CHARACTER
762 001500 012706 001100  MOV      #STACK,SP
763 001504 004737 011266  JSR     PC,RST        ;RESET & SET INTRPT. EN.
764 001510 000137 002256  JMP     BEG2
765 001514 120027 000001  1$:  CMPB    RO,#1          ;IS IT ↑A?
766 001520 001010          BNE     2$
767 001522 104400 012141  TYPE    ,AMMSG        ;ECHO CHARACTER
768 001526 012706 001100  MOV      #STACK,SP
769 001532 004737 011266  JSR     PC,RST        ;RESET & SET INTRPT. EN.
770 001536 000177 177626  JMP     @TADDR        ;RETURN TO TEST
771 001542 120027 000007  2$:  CMPB    RO,#7          ;IS IT ↑G?
772 001546 001021          BNE     NONE
773 001550 023727 001140 177570  CMP     SWR,#177570   ;HARDWARE SWREG?
774 001556 001415          BEQ     NONE
775 001560 104400 012153  TYPE    ,GMSG        ;ECHO CHARACTER
776 001564 017746 177350  MOV      @SWR,-(SP)   ;;SAVE @SWR FOR TYPEOUT
777          TYPOS      ;;TYPE SWREG
778 001570 104402          .BYTE    6          ;;GO TYPE--OCTAL ASCII
779 001572          .BYTE    1          ;;TYPE 6 DIGITS
780 001573          .BYTE    1          ;;TYPE LEADING ZEROS
781 001574 104400 012333  TYPE    ,SLASH
782 001600 104406          RDOCT
783 001602 012677 177332  MOV     (SP)+,@SWR    ;READ NEW VALUE
784 001606 012600          MOV     (SP)+,RO     ;LOAD NEW SWREG VALUE
785 001610 000002  POPRO:  RTI
786 001612 104400 012137  NONE:  TYPE    ,QUEST   ;TYPE "?"
787 001616 000773          BR      POPRO

```

MAINDEC-11-DZADL-8
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 21
INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE

```

788          .SBTTL          INITIAL START-UP, HOUSEKEEPING, AND DIALOGUE
789 001620 005037 001422 BEGIN: CLR      WFTST
790 001624 000403          BR      RBEG
791 001626 012737 000001 001422 BEGIN2: MOV     #1, WFTST
792 001634 000005          RBEG:  RESET
793          .SBTTL          INITIALIZE THE COMMON TAGS
794          ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
795 001636 012706 001100          MOV     #SCMTAG, R6          ;; FIRST LOCATION TO BE CLEARED
796 001642 005026          CLR      (R6)+          ;; CLEAR MEMORY LOCATION
797 001644 022706 001140          CMP     #SWR, R6          ;; DONE?
798 001650 001374          BNE     -6          ;; LOOP BACK IF NO
799 001652 012706 001100          MOV     #STACK, SP          ;; SETUP THE STACK POINTER
800          ;; INITIALIZE A FEW VECTORS
801 001656 012737 015130 000020          MOV     $$SCOPE, @IOTVEC          ;; IOT VECTOR FOR SCOPE ROUTINE
802 001664 012737 000340 000022          MOV     #340, @IOTVEC+2          ;; LEVEL 7
803 001672 012737 015406 000030          MOV     $ERROR, @EMTVEC          ;; EMT VECTOR FOR ERROR ROUTINE
804 001700 012737 000340 000032          MOV     #340, @EMTVEC+2          ;; LEVEL 7
805 001706 012737 016714 000034          MOV     $TRAP, @TRAPVEC          ;; TRAP VECTOR FOR TRAP CALLS
806 001714 012737 000340 000036          MOV     #340, @TRAPVEC+2          ;; LEVEL 7
807 001722 012737 016754 000024          MOV     $PWDN, @PWRVEC          ;; POWER FAILURE VECTOR
808 001730 012737 000340 000026          MOV     #340, @PWRVEC+2          ;; LEVEL 7
809 001736 013737 011756 011750          MOV     SENDCT, $EOPCT          ;; SETUP END-OF-PROGRAM COUNTER
810 001744 005037 001160          CLR      $TIMES          ;; INITIALIZE NUMBER OF ITERATIONS
811 001750 005037 001162          CLR      $ESCAPE          ;; CLEAR THE ESCAPE ON ERROR ADDRESS
812 001754 112737 000001 001115          MOV     #1, $ERMAX          ;; ALLOW ONE ERROR PER TEST
813 001762 012737 001762 001106          MOV     #., $LPADR          ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
814 001770 012737 001770 001110          MOV     #., $LPERR          ;; SETUP THE ERROR LOOP ADDRESS
815          ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
816          ;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
817 001776 013746 000004          MOV     @ERRVEC, -(SP)          ;; SAVE ERROR VECTOR
818 002002 012737 002036 000004          MOV     #64$, @ERRVEC          ;; SET UP ERROR VECTOR
819 002010 012737 177570 001140          MOV     #DSWR, SWR          ;; SETUP FOR A HARDWARE SWITCH REGISTER
820 002016 012737 177570 001142          MOV     #DDISP, DISPLAY          ;; AND A HARDWARE DISPLAY REGISTER
821 002024 022777 177777 177106          CMP     #-1, @SWR          ;; TRY TO REFERENCE HARDWARE SWR
822 002032 001012          BNE     66$          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
823          ;; AND THE HARDWARE SWR IS NOT = -1
824 002034 000403          BR      65$          ;; BRANCH IF NO TIMEOUT
825 002036 012716 002044          64$: MOV     #65$, (SP)          ;; SET UP FOR TRAP RETURN
826 002042 000002          RTI
827 002044 012737 000176 001140          65$: MOV     #SWREG, SWR          ;; POINT TO SOFTWARE SWR
828 002052 012737 000174 001142          MOV     #DISPREG, DISPLAY
829 002060 012637 000004          66$: MOV     (SP)+, @ERRVEC          ;; RESTORE ERROR VECTOR
830
831 002064 005037 001202          CLR      $PASS          ;; CLEAR PASS COUNT
832 002070 132737 000200 001215          BITB   #APTSIZE, $ENVM          ;; TEST USER SIZE UNDER APT
833 002076 001403          BEQ     67$          ;; YES, USE NON-APT SWITCH
834 002100 012737 001216 001140          MOV     $$SWREG, SWR          ;; NO, USE APT SWITCH REGISTER
835 002106          67$:

```

836	002106	005037	001404		CLR	FLAG	;CLEAR VT55 FLAG
837	002112	005737	000042		TST	@#42	;IS IT CHAINED?
838	002116	001033			BNE	REST1	
839							
840	002120	042777	000100	177016	.SBTTL	BIC	DETERMINE IF VT55 TYPE TERMINAL IS PRESENT
841	002126	104400	013573		TYPE	#100,@\$TKS	
842	002132	004737	002530		JSR	PC,VTFLG	;TYPE ASCIZ STRING
843	002136	020027	000033		CMP	RO,#33	;GET A CHARACTER
844	002142	001017			BNE	NOVT55	;NO VT55 PRESENT
845	002144	004737	002530		JSR	PC,VTFLG	;GET A CHARACTER
846	002150	020027	000057		CMP	RO,#57	
847	002154	001012			BNE	NOVT55	;NO VT55 PRESENT
848	002156	004737	002530		JSR	PC,VTFLG	;GET A CHARACTER
849	002162	020027	000103		CMP	RO,#103	
850	002166	001403			BEQ	VT55	;VT55 IS PRESENT
851	002170	020027	000105		CMP	RO,#105	
852	002174	001002			BNE	NOVT55	
853	002176	005237	001404		VT55: INC	FLAG	

```

854      .SBTTL      DIALOGUE TO DETERMINE WHICH TEST TO RUN
855 002202 104400 013736 NOVT55: TYPE      ,HEAD1
856 002206 000005 REST1:  RESET
857 002210 004737 005504      JSR      PC, FIXONE      ;INITIALIZE ADDRESSES
858 002214 013700 001340      MOV      KBVECT, RO
859 002220 012720 001454      MOV      #ISERV, (RO)+
860 002224 012710 000340      MOV      #340, (RO)
861 002230 012737 062341 001372      MOV      #62341, RNA      ;RANDOM NO, VARIABLES
862 002236 012737 142315 001374      MOV      #142315, RNB
863 002244 012737 127623 001376      MOV      #127623, RNC
864 002252 004737 011552      JSR      PC, WFADJ      ;STANDARD OR OPTION TEST TOLERANCES?
865 002256 000005      BEG2:  RESET      ;RESTART ADDRESS
866 002260 012706 001100      MOV      #STACK, SP      ;RESET STACK IN CASE RESTARTED
867 002264 005737 000042      TST      @#42      ;IS IT CHAINED?
868 002270 001402      BEQ      1$
869 002272 000137 005234      JMP      BEGL      ;GO TO LOGIC TESTS
870 002276 104400 013401      1$:  TYPE      ,MSG71
871 002302 104405      TRYAG: RDLIN
872 002304 052777 000100 176632      BIS      #100, @STKS
873 002312 005037 177776      CLR      PSW
874 002316 012600      MOV      (SP)+, RO      ;READ ANSWER
875 002320 142710 000040      BICB    #40, (RO)
876 002324 121027 000101      CMPB    (RO), #'A
877 002330 001002      BNE     1$      ;;NO, TRY C      ;IS IT A?
878 002332 000137 005272      JMP     BEGINA      ;GO TO AUTO TEST
879 002336 121027 000103      1$:  CMPB    (RO), #'C      ;IS IT C?
880 002342 001002      BNE     2$      ;;NO, TRY L      ;GO TO CALIBRATION TEST
881 002344 000137 005050      JMP     BEGINC      ;IS IT L?
882 002350 121027 000114      2$:  CMPB    (RO), #'L
883 002354 001002      BNE     3$      ;;NO, TRY N      ;GO TO LOGIC TESTS
884 002356 000137 005234      JMP     BEGL      ;IS IT N?
885 002362 121027 000116      3$:  CMPB    (RO), #'N
886 002366 001002      BNE     4$      ;;NO, TRY S      ;GO TO NOISE TEST
887 002370 000137 005660      JMP     BEGINN      ;IS IT S?
888 002374 121027 000123      4$:  CMPB    (RO), #'S
889 002400 001002      BNE     5$      ;;NO, TRY W      ;GO TO SETTLE TEST
890 002402 000137 005730      JMP     BEGINS      ;IS IT W?
891 002406 121027 000127      5$:  CMPB    (RO), #'W
892 002412 001002      BNE     6$      ;;NO, TRY AGAIN      ;GO TO WRAPAROUND TEST
893 002414 000137 005360      JMP     BEGINW
894 002420 104400 012137      6$:  TYPE      ,QUEST
895 002424 000726      BR      †TRYAG      ;WAIT FOR CHARACTER
896 002426 013737 001250 001126      TESTAD: MOV     $BASE, $BDDAT      ;SETUP TO TEST FOR AD11K'S
897 002434 013746 000004      MOV     @#ERRVEC, -(SP)      ;SAVE ERRVEC
898 002440 012737 002472 000004      MOV     #2$, ERRVEC      ;SET UP FOR TIME OUT ERROR
899 002446 005037 001360      CLR     NBEXT      ;CLEAR AD11K COUNTER
900 002452 005777 176450      1$:  TST     @BDDAT      ;ADDRESS AD11K
901 002456 005237 001360      INC     NBEXT      ;INCREMENT AD11K COUNTER
902 002462 063737 001332 001126      ADD     VADR, $BDDAT      ;GET NEXT AD11K
903 002470 000770      BR     1$      ;;TRY NEXT AD11K

```


904	002472	022626		2\$:	CMP	(SP)+,(SP)+		;POP 2 WORDS OFF STACK
905	002474	013746	001360		MOV	NBEXT,-(SP)		::SAVE NBEXT FOR TYPEOUT
906								::TYPE NUMBER OF AD11K'S
907	002500	104402			TYPOS			::GO TYPE--OCTAL ASCII
908	002502	002			.BYTE	2		::TYPE 2 DIGIT(S)
909	002503	000			.BYTE	0		::SUPPRESS LEADING ZEROS
910	002504	104400	012741		TYPE	,MSG50		
911	002510	005337	001360		DEC	NBEXT		;ADJUST AD11K COUNT
912	002514	013737	001360	001362	MOV	NBEXT,NMBEXT		;KEEP COUNT OF NUMBER
913	002522	012637	000004		MOV	(SP)+,ERRVEC		;RESTORE ERRVEC
914	002526	000207			RTS	PC		
915								
916	002530	005000		VTFLG:	CLR	RO		;TEST FOR PRESENCE
917	002532	105777	176406	1\$:	TSTB	Q\$TKS		;OF VT55
918	002536	100404			BMI	2\$::VT55 RESPONDS WITH <33><57>[<103> OR <105>]
919	002540	005300			DEC	RO		
920	002542	001373			BNE	1\$::
921	002544	005726			TST	(SP)+		;POP A WORD OFF STACK
922	002546	000615			BR	NOVT55		::NO VT55 PRESENT
923	002550	017700	176372	2\$:	MOV	Q\$TKB,RO		
924	002554	042700	177600		BIC	#177600,RO		;TEST VT55 CODE
925	002560	000207			RTS	PC		

```

926 002562          BEGINL:
927                ;:*****
928                ;*TEST 1      FLOAT A ONE THRU MULTIPLEXER BITS
929                ;:*****
930 002562 012737 002562 001106 †ST1:  MOV    #TST1,$LPADR
931 002570 012737 002562 001110      MOV    #TST1,$LPERR
932 002576 012737 000400 001124      MOV    #BIT8,$GDDAT      ;LOAD FIRST BIT
933 002604 004737 003660      2$:    JSR    PC,TESTIT
934 002610 104001      ERROR    1      ;FAILED TO LOAD + READ BIT
935 002612 006137 001124      1$:    ROL    $GDDAT      ;GET NEXT BIT
936 002616 023727 001124 040000      CMP    $GDDAT,#BIT14    ;FINISHED?
937 002624 001367      BNE     2$      ;;NO,GO TO NEXT TEST
938
939                ;:*****
940                ;*TEST 2      LOAD AND READ BACK INTERRUPT ENABLE BIT6
941                ;:*****
942 002626 000004      †ST2:  SCOPE
943 002630 012777 001432 176466      MOV    #UNEXP,$VECTOR    ;SETUP FOR UNEXPECTED INTERUPT
944 002636 012737 000100 001124      MOV    #BIT6,$GDDAT      ;LOAD EXPECTED DATA
945 002644 004737 003660      JSR    PC,TESTIT
946 002650 104001      ERROR    1      ;FAILED TO LOAD + READ INTERRUPT ENABLE
947
948                ;:*****
949                ;*TEST 3      LOAD AND READ BACK CLOCK OVERFLOW START ENABLE BITS
950                ;:*****
951 002652 000004      †ST3:  SCOPE
952 002654 012737 000040 001124      MOV    #BIT5,$GDDAT      ;LOAD EXPECTED DATA
953 002662 004737 003660      JSR    PC,TESTIT
954 002666 104001      ERROR    1      ;FAILED TO LOAD + READ CLOCK OVERFLOW START ENAB
955
956                ;:*****
957                ;*TEST 4      LOAD AND READ BACK EXTERNAL START ENABLE BIT4
958                ;:*****
959 002670 000004      †ST4:  SCOPE
960 002672 012737 000020 001124      MOV    #BIT4,$GDDAT      ;LOAD EXPECTED DATA
961 002700 004737 003660      JSR    PC,TESTIT
962 002704 104001      ERROR    1      ;FAILED TO LOAD + READ EXT. START ENABLE
963
964                ;:*****
965                ;*TEST 5      LOAD AND READ BACK ERROR FLAG BIT15
966                ;:*****
966 002706 000004      †ST5:  SCOPE
967 002710 012737 100000 001124      MOV    #BIT15,$GDDAT     ;LOAD EXPECTED DATA
968 002716 004737 003660      JSR    PC,TESTIT
969 002722 104001      ERROR    1      ;FAILED TO LOAD + READ ERROR FLAG

```

```

970
971
972
973 002724 000004
974 002726 012737 000300 001160
975 002734 005037 011124
976 002740 012777 137562 176350
977 002746 012737 000010 001352
978 002754 005337 001352
979 002760 001375
980 002762 000005
981 002764 052777 000100 176152
982 002772 017737 176320 001126
983 003000 001401
984 003002 104001
985
986
987
988
989 003004 000004
990 003006 012700 001000
991 003012 005277 176300
992 003016 012737 000200 001124
993 003024 005300
994 003026 001376
995 003030 042777 100000 176260
996 003036 004737 003666
997 003042 104001
998 003044 017700 176252
999
1000
1001
1002
1003 003050 000004
1004 003052 012737 000300 001160
1005 003060 005037 001124
1006 003064 005277 176226
1007 003070 105777 176222
1008 003074 100375
1009 003076 000005
1010 003100 004737 003666
1011 003104 104001
1012 003106 052777 000100 176030
1013
1014
1015
1016
1017 003114 000004
1018 003116 012777 000001 176172
1019 003124 105777 176166
1020 003130 100375
1021 003132 017700 176164
1022 003136 004737 003666
1023 003142 104001

```

```

*****
*TEST 6 TEST INIT CLEARS BITS 1,4-6,8-13,15
*****
TST6: SCOPE
MOV #300,$TIMES ;DO 300 ITERATIONS
CLR $GDDAT ;LOAD EXPECTED DATA
2$: MOV #137562,$STREG ;SET STATUS REGISTER
MOV #10,TEMP
1$: DEC TEMP ;DELAY FOR ONE-SHOT
BNE 1$ ;IN ANALOG PACKAGE
RESET ;INITIALIZE
BIS #100,$STKS ;SET INTRPT. ENABLE
MOV $STREG,$BDDAT ;READ STATUS REGISTER
BEQ TST7 ;NEXT TEST
ERROR 1 ;RESET FAILED TO CLEAR AD ST. REG. BITS

*****
*TEST 7 TEST DONE FLAG SETS AND BIT0 CLEARS ON END OF CONV.
*****
TST7: SCOPE
MOV #BIT9,RO ;STALL TIME COUNTER
INC $STREG ;START CONVERSION
MOV #BIT7,$GDDAT ;LOAD EXPECTED
1$: DEC RO ;STALL
BNE 1$ ;TIME
BIC #BIT15,$STREG ;MASK OUT ERROR BIT
JSR PC,TEST
ERROR 1 ;A/D DONE FLAG FAILED TO SET;BIT0 FAILED TO CLEAR
MOV $ADBUFF,RO ;CLEAR DONE FLAG FOR ITERATIONS

*****
*TEST 10 TEST INIT CLEARS DONE FLAG
*****
TST10: SCOPE
MOV #300,$TIMES ;DO 300 ITERATIONS
CLR $GDDAT ;CLEAR EXPECTED
INC $STREG ;START CONVERSION
2$: TSTB $STREG
BPL 2$
RESET
JSR PC,TEST
ERROR 1 ;DONE FLAG FAILED TO CLEAR
BIS #100,$STKS ;SET INTRPT. EN. BIT

*****
*TEST 11 TEST A/D DONE FLAG CLEARS WHEN READ CONVERTED VALUE
*****
TST11: SCOPE
MOV #BIT0,$STREG ;SET A/D START CONVERSION BIT
1$: TSTB $STREG ;WAIT FOR FLAG
BPL 1$
MOV $ADBUFF,RO ;READ CONVERTED VALUE
JSR PC,TEST
ERROR 1 ;DONE FLAG FAILED TO CLEAR

```

```

1024
1025
1026
1027 003144 000004
1028
1029
1030
1031
1032 003146 012700 000012
1033 003152 004737 011164
1034 003156 005037 177776
1035 003162 012777 003230 176134
1036 003170 012777 000101 176120
1037 003176 105777 176114
1038 003202 100375
1039 003204 017737 176106 001126
1040 003212 012737 000300 001124
1041 003220 104002
1042 003222 004737 011236
1043 003226 000403
1044 003230 022626
1045 003232 004737 011236
1046
1047
1048
1049
1050 003236 000004
1051
1052
1053
1054
1055 003240 012700 000013
1056 003244 004737 011164
1057 003250 013737 001326 177776
1058 003256 162737 000040 177776
1059 003264 012777 003320 176032
1060 003272 012777 000101 176016
1061 003300 105777 176012
1062 003304 100375
1063 003306 017737 176004 001126
1064 003314 104002
1065 003316 000401
1066 003320 022626
1067 003322 012777 003402 175774
1068 003330 013737 001326 177776
1069 003336 012777 000101 175752
1070 003344 105777 175746
1071 003350 100375
1072 003352 012777 001432 175744
1073 003360 005077 175732
1074 003364 005037 177776
1075 003370 017700 175726
1076 003374 004737 011236
1077 003400 000403

```

```

*****
*TEST 12 GENERATE INTERRUPT WHEN DONE FLAG SETS AFTER CONVERSION
*****
TST12: SCOPE
;* "ENTERING TEST 12" TYPED OUT TO TELL YOU THE NEXT
;* TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
;* THERE IS DANGER THAT THE UNIBUS COULD GET "HUNG" WHILE
;* EXECUTING TEST "12".
MOV #12,RO ;GET TEST NO.
JSR PC,DUMW ;PRINT MESSAGE
CLR PSW ;RESET PRIORITY
MOV #1$,AVECTOR ;INTERRUPT VECTOR ADDRESS
MOV #BIT6!BIT0,ASTREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
2$: TSTB ASTREG ;WAIT FOR DONE
BPL 2$ ;FLAG TO SET
MOV ASTREG,$BDDAT ;READ STATUS REGISTER
MOV #BIT7!BIT6,$GDDAT ;GOOD DATA
ERROR 2 ;FAILED TO INTERRUPT ON DONE
JSR PC,DUMC ;TYPE COMPLETED
BR TST13 ;BRANCH TO NEXT TEST
1$: CMP (SP)+,(SP)+ ;RESET STACK POINTER
JSR PC,DUMC ;TYPE COMPLETED

*****
*TEST 13 TEST FOR CORRECTNESS OF PRIORITY OF INTERRUPT
*****
TST13: SCOPE
;* "ENTERING TEST 13" TYPED OUT TO TELL YOU THE NEXT
;* TEST THAT IS GOING TO BE EXECUTED. IT IS ONLY TYPED ON PASS 0.
;* THERE IS DANGER THAT THE UNIBUS COULD GET "HUNG" WHILE
;* EXECUTING TEST "13".
MOV #13,RO ;GET TEST NO.
JSR PC,DUMW ;PRINT MESSAGE
MOV BASEBR,PSW ;GET ONE LEVEL LESS
SUB #40,PSW ;PRIORITY
MOV #1$,AVECTOR ;INTERRUPT VECTOR ADDRESS
MOV #BIT6!BIT0,ASTREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
2$: TSTB ASTREG ;WAIT FOR
BPL 2$ ;DONE FLAG
MOV ASTREG,$BDDAT ;READ ST. REG.
ERROR 2 ;INTERRUPT NOT SERVICED AT LEVEL 5
BR 3$
1$: CMP (SP)+,(SP)+ ;RESET STACK POINTER
3$: MOV #4$,AVECTOR ;INTERRUPT VECTOR ADDRESS
MOV BASEBR,PSW ;SET PRIORITY TO 6
MOV #BIT6!BIT0,ASTREG ;SET INTERRUPT ENABLE + START CONVERSION
5$: TSTB ASTREG ;WAIT FOR
BPL 5$ ;DONE FLAG
6$: MOV #UNEXP,AVECTOR ;UNEXPECTED INTERRUPT VECTOR
CLR ASTREG ;CLEAR BIT 6
CLR PSW ;RESET PRIORITY
MOV ADDBUFF,RO ;CLEAR DONE FLAG
JSR PC,DUMC ;TYPE COMPLETED
BR TST14 ;BRANCH TO NEXT TEST

```

```

1078 003402 022626 4$: CMP (SP)+,(SP)+ ;RESET STACK POINTER
1079 003404 104003 ERROR 3 ;INTERRUPTED AT LEVEL 6 - WRONG
1080 003406 000761 BR 6$

:*****
: *TEST 14 TEST DONE FLAG CLEARS AFTER INTERRUPT
:*****
1081
1082
1083
1084
1085 003410 000004 †ST14: SCOPE
1086 003412 012737 000300 001160 MOV #300,$TIMES ;DO 300 ITERATIONS
1087 003420 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD GOOD DATA
1088 003426 012777 003462 175670 MOV #1$,$VECTOR ;INTERRUPT VECTOR ADDRESS
1089 003434 012777 000101 175654 MOV #BIT6!BIT0,$STREG ;SET INTERRUPT ENABLE BIT + START CONVERSION
1090 003442 105777 175650 2$: TSTB $STREG ;WAIT FOR
1091 003446 100375 BPL 2$ ;DONE FLAG
1092 003450 017737 175642 001126 MOV $STREG,$BDDAT ;READ STATUS REGISTER
1093 003456 104002 ERROR 2 ;FAILED TO GENERATE INTERRUPT
1094 003460 000412 BR TST15 ;GO TO NEXT TEST
1095 003462 022626 1$: CMP (SP)+,(SP)+ ;RESET STACK POINTER
1096 003464 004737 003666 JSR PC,TEST
1097 003470 104001 ERROR 1 ;DONE FLAG NOT CLEARED AFTER AN INTERRUPT
1098 003472 017737 175624 001352 MOV $ADDBUFF,TEMP ;DUMMY READ BUFFER
1099 003500 012777 001432 175616 MOV #UNEXP,$VECTOR ;LOAD INTRPT. ADDRESS

:*****
: *TEST 15 TEST ERROR FLAG SETS IF 2ND CONVERSION ENDS BEFORE READING BUFFER
:*****
1100
1101
1102
1103
1104 003506 000004 †ST15: SCOPE
1105 003510 012777 000001 175600 MOV #BIT0,$STREG ;START CONVERSION
1106 003516 105777 175574 1$: TSTB $STREG ;WAIT FOR
1107 003522 100375 BPL 1$
1108 003524 012737 100200 001124 2$: MOV #BIT15!BIT7,$GDDAT ;LOAD EXPECTED VALUE
1109 003532 012777 000001 175556 MOV #BIT0,$STREG ;START 2ND CONVERSION
1110 003540 012700 001000 MOV #BIT9,$RO ;WAIT FOR 2ND
1111 003544 005300 3$: DEC $RO ;CONVERSION TO END
1112 003546 001376 BNE 3$
1113 003550 004737 003666 4$: JSR PC,TEST
1114 003554 104001 ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND
1115 ; CONVERT ENDS BEFORE READ BUFFER FROM FIRST
1116 003556 017700 175540 MOV $ADDBUFF,$RO ;CLEAR DONE FLAG

```

```

1117
1118
1119
1120 003562 000004
1121 003564 012737 100000 001124
1122 003572 012700 000100
1123 003576 012777 000001 175512
1124 003604 112777 000001 175504
1125 003612 005300
1126 003614 001376
1127 003616 017737 175474 001126
1128 003624 042737 077777 001126
1129 003632 023737 001124 001126
1130 003640 001401
1131 003642 104001
1132
1133 003644 017700 175452
1134 003650 005077 175442
1135 003654 000004
1136 003656 000207
1137
1138
1139
1140 003660 013777 001124 175430
1141 003666 017737 175424 001126
1142 003674 023737 001124 001126
1143 003702 001002
1144 003704 062716 000002
1145 003710 000207

::*****
:*TEST 16 TEST ERROR FLAG SETS IF START 2ND CONV. BEFORE DONE FLAG SETS
::*****
†ST16: SCOPE
MOV #BIT15,$GDDAT ;LOAD EXPECTED DATA
MOV #100,R0 ;STALL TIME COUNTER
MOV #BIT0,$STREG ;START CONVERSION
MOVB #BIT0,$STREG ;START NEXT CONVERSION
DECS: DEC R0 ;STALL TIME
BNE DECS
MOV $STREG,$BDDAT ;READ STATUS REGISTER
BIC #7777,$BDDAT ;MASK OUT BIT 15
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BEQ 1$ ;BRANCH OVER ERROR
ERROR 1 ;ERROR FLAG NOT SET WHEN 2ND
;CONVERT BEGINS BEFORE FIRST DONE
1$: MOV $ADDBUFF,R0 ;READ CONVERTED VALUE
CLR $STREG ;CLEAR STATUS REGISTER
SCOPE
RTS PC ;RETURN TO TEST SECTION

::SUBROUTINE FOR LOGIC TESTS:
TESTIT: MOV $GDDAT,$STREG ;LOAD EXPECTED VALUE
TEST: MOV $STREG,$BDDAT ;READ ST. REG.
CMP $GDDAT,$BDDAT ;COMPARE RESULTS
BNE RETERR ;:ERROR RETURN
ADD #2,(SP) ;BUMP RETURN ADDRESS TO GET AROUND ERROR
RETERR: RTS PC

```

```

1146
1147 003712
1148
1149
1150
1151 003712 000240
1152 003714 012737 000010 001160
1153 003722 012737 000017 001102
1154 003730 012737 003712 001110
1155 003736 012737 003712 001106
1156 003744 004537 011004
1157 003750 000014
1158 003752 004537 011116
1159 003756 004000
1160 003760 011630
1161 003762 104004
1162
1163
1164
1165
1166
1167
1168
1169 003764 000004
1170 003766 012737 000010 001160
1171 003774 005737 001336
1172 004000 001024
1173 004002 012777 000020 175306
1174 004010 012700 001000
1175 004014 012737 000220 001124
1176 004022 012777 000200 175272
1177
1178 004030 005300
1179 004032 001376
1180 004034 004737 003666
1181 004040 104001
1182 004042 017700 175254
1183 004046 005077 175244
1184
1185
1186
1187
1188
1189 004052 000004
1190 004054 012737 000010 001160
1191 004062 004537 011004
1192 004066 000000
1193 004070 004537 011116
1194 004074 004000
1195 004076 011622
1196 004100 104004

```

```

.SBTTL WRAPAROUND TEST SECTION
WRAP:
*****
*TEST 17 TEST CH14 GROUND
*****
TST17: NOP
MOV #10,STIMES ;;DO 10 ITERATIONS
MOV #STN-1,STSTNM
MOV #TST17,SLPERR
MOV #TST17,SLPADR
JSR RS,CONVRT ;DO 8 CONVERSIONS
14
JSR RS,COMPAR ;COMPARE RESULTS
4000 ;NOMINAL
V50 ;TOLERANCE
ERROR 4 ;ERROR-CH14 NOT GROUND-AD11K MUST BE IN
;SINGLE-ENDED CONFIGURATION,G5036 WRAPAROUND
;MODULE MUST BE PRESENT,CHECK CONNECTION A-VV,VV-A

*****
*TEST 20 TEST CONVERSION FROM EXT. START
*****
TST20: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
TST BASECH ;TESTING AN AM?
BNE TST21 ;;YES, GOTO NEXT TEST
MOV #BIT4,ASTREG ;ENABLE EXTERNAL START
MOV #BIT9,RO ;TIME DELAY COUNTER
MOV #BIT7,BIT4,$GDDAT ;LOAD EXPECTED
MOV #200,ADDBUFF ;SIMULATE EXT. START WITH G5036
;WRAPAROUND MODULE PRESENT

15: DEC RO
BNE 15
JSR PC,TEST
ERROR 1 ;FAILED TO DO CONVERSION FROM EXT. START
MOV ADDBUFF,RO
CLR ASTREG ;CLEAR ST. REG.

*****
*TEST 21 TEST CH0 GROUND
*****
TST21: SCOPE
MOV #10,STIMES ;;DO 10 ITERATIONS
JSR RS,CONVRT ;CONVERT 8 TIMES
0
JSR RS,COMPAR ;COMPARE RESULTS
4000 ;NOMINAL
V1 ;TOLERANCE
ERROR 4 ;ERROR ON A/D CHANNEL

```

F03

MAINDEC-11-DZADL-8
DZADLB.P11 T22

MACY11 27(665) 2-DEC-76 16:29 PAGE 31
TEST CH1 GROUND

1197				
1198				
1199				
1200	004102	000004		
1201	004104	012737	000010	001160
1202	004112	004537	011004	
1203	004116	000001		
1204	004120	004537	011116	
1205	004124	004000		
1206	004126	011626		
1207	004130	104004		
1208				
1209				
1210				
1211				
1212	004132	000004		
1213	004134	012737	000010	001160
1214	004142	004537	011004	
1215	004146	000002		
1216	004150	004537	011116	
1217	004154	004632		
1218	004156	011630		
1219	004160	104004		
1220				
1221				
1222				
1223				
1224				
1225	004162	000004		
1226	004164	012737	000010	001160
1227	004172	004537	011004	
1228	004176	000003		
1229	004200	004537	011116	
1230	004204	006000		
1231	004206	011636		
1232	004210	104004		
1233				
1234				
1235				
1236				
1237	004212	000004		
1238	004214	012737	000010	001160
1239	004222	004537	011004	
1240	004226	000004		
1241	004230	004537	011116	
1242	004234	002000		
1243	004236	011636		
1244	004240	104004		

```

*****
*TEST 22      TEST CH1 GROUND
*****

```

```

†ST22:  SCOPE
        MOV      #10,$TIMES      ;;DO 10 ITERATIONS
        JSR      R5,CONVRT      ;CONVERT 8 TIMES
        JSR      R5,CONVRT      ;CHANNEL 1
        JSR      R5,COMPAR      ;COMPARE RESULTS
        4000      ;NOMINAL
        V10      ;TOLERANCE
        ERROR    4              ;ERROR ON A/D CHANNEL

```

```

*****
*TEST 23      TEST CH2 +1 VOLT
*****

```

```

†ST23:  SCOPE
        MOV      #10,$TIMES      ;;DO 10 ITERATIONS
        JSR      R5,CONVRT      ;CONVERT 8 TIMES
        JSR      R5,CONVRT      ;CHANNEL 2
        JSR      R5,COMPAR      ;COMPARE RESULTS
        4632      ;NOMINAL
        V50      ;TOLERANCE
        ERROR    4              ;ERROR ON A/D CHANNEL
        ;AD11K MUST BE SET UP FOR +OR- 5V OR +OR- 5.12V

```

```

*****
*TEST 24      TEST CH3 +2.5 VOLTS
*****

```

```

†ST24:  SCOPE
        MOV      #10,$TIMES      ;;DO 10 ITERATIONS
        JSR      R5,CONVRT      ;CONVERT 8 TIMES
        JSR      R5,CONVRT      ;CHANNEL 3
        JSR      R5,COMPAR      ;COMPARE RESULTS
        6000      ;NOMINAL
        V240     ;TOLERANCE
        ERROR    4              ;ERROR ON A/D CHANNEL

```

```

*****
*TEST 25      TEST CH4 -2.5 VOLTS
*****

```

```

†ST25:  SCOPE
        MOV      #10,$TIMES      ;;DO 10 ITERATIONS
        JSR      R5,CONVRT      ;CONVERT 8 TIMES
        JSR      R5,CONVRT      ;CHANNEL 4
        JSR      R5,COMPAR      ;COMPARE RESULTS
        2000      ;NOMINAL
        V240     ;TOLERANCE
        ERROR    4

```



```

1245
1246
1247
1248 004242 000004
1249 004244 012737 000001 001160
1250 004252 005077 175044
1251 004256 004737 005040
1252 004262 004537 011004
1253 004266 000012
1254 004270 013704 001352
1255 004274 004537 011116
1256 004300 002376
1257 004302 011634
1258 004304 104004
1259 004306 005037 001424
1260 004312 012702 000001
1261 004316 010277 175000 1$:
1262 004322 004737 005040
1263 004326 004537 011004
1264 004332 000012
1265 004334 005737 001424
1266 004340 001010
1267 004342 023727 001352 004000
1268 004350 002404
1269 004352 005237 001424
1270 004356 010237 001420
1271 004362 020227 000200 2$:
1272 004366 001003
1273 004370 013737 001352 004460
1274 004376 013703 001352 3$:
1275 004402 160437 001352
1276 004406 010304
1277 004410 004537 011116
1278 004414 000006
1279 004416 011640
1280 004420 104004
1281 004422 005202
1282 004424 020227 000400
1283 004430 001332
1284 004432 000005
1285 004434 052777 000100 174502
1286 004442 004737 005040
1287 004446 004537 011004
1288 004452 000012
1289 004454 004537 011116
1290 004460 000000 4$:
1291 004462 011624
1292 004464 104004

```

```

*****
*TEST 26 TEST VERNIER OFFSET DAC ON CH12
*****
†ST26: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
CLR @ADBUFF ;VERNIER DAC=0
JSR PC,DAWAIT ;DELAY FOR DAC SETTling
JSR R5,CONVRT ;CONV. CH12, DIRECT VERNIER DAC
12
MOV TEMP,R4 ;SAVE VALUE IN R4
JSR R5,COMPAR ;COMPARE RESULTS
2376 ;WITH -1.875 VOLTS
V115 ;TOLERANCE OF 10%
ERROR 4
CLR MAX
MOV #1,R2
1$: MOV R2,@ADBUFF ;SET UP NEXT VERNIER DAC VALUE
JSR PC,DAWAIT ;DELAY FOR DAC SETTling
JSR R5,CONVRT ;CONVERT IT
12
TST MAX
BNE 2$
CMP TEMP,#4000
BLT 2$
INC MAX
MOV R2,MIN
2$: CMP R2,#200
BNE 3$
MOV TEMP,4$
3$: MOV TEMP,R3 ;SAVE VALUE
SUB R4,TEMP ;TEMP=DIFF. BETWEEN VALUE&PREVIOUS
MOV R3,R4 ;SET UP PREVIOUS VALUE FOR NEXT TIME THRU
JSR R5,COMPAR ;COMPARE RESULTS
6 ;WITH 15 MILLIVOLTS(1 DAC LSB)
V5
ERROR 4
INC R2
CMP R2,#400 ;DONE?
BNE 1$ ;NO-DO NEXT VERNIER DAC VALUE
RESET ;SHOULD SET VERNIER DAC TO 200
BIS #100,@STKS
JSR PC,DAWAIT ;LET DAC SETTLE
JSR R5,CONVRT ;CONVERT IT
12
JSR R5,COMPAR ;COMPARE RESULTS
0
V2
ERROR 4

```

H03

MAINDEC-11-DZADL-B
DZADLB.P11 T27

MACY11 27(665) 2-DEC-76 16:29 PAGE 33
TEST CH13 +2.5 VOLTS

```

1293      ::*****
1294      ::*TEST 27      TEST CH13 +2.5 VOLTS
1295      ::*****
1296 004466 000004      TST27: SCOPE
1297 004470 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1298 004476 004537 011004      JSR      RS,CONVRT      ;CONVERT 8 TIMES
1299 004502 000013      13
1300 004504 004537 011116      JSR      RS,COMPAR      ;COMPARE RESULTS
1301 004510 006000      6000      ;NOMINAL
1302 004512 011632      V144      ;TOLERANCE
1303 004514 104004      ERROR 4
1304      ::*****
1305      ::*TEST 30      TEST CH17 +4V
1306      ::*****
1307 004516 000004      TST30: SCOPE
1308 004520 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
1309 004526 004537 011004      JSR      RS,CONVRT      ;CONVERT 8 TIMES
1310 004532 000017      17      ;CHANNEL 17
1311 004534 004537 011116      JSR      RS,COMPAR      ;COMPARE RESULTS
1312 004540 007146      7146      ;NOMINAL
1313 004542 011636      V240      ;TOLERANCE
1314 004544 104004      ERROR 4      ;ERROR ON A/D CHANNEL

```

```

1315
1316
1317
1318 004546 000004
1319 004550 012737 000001 001160
1320 004556 013737 001336 001366
1321 004564 013737 001336 001364
1322 004572 012737 004001 001414
1323 004600 004537 006516
1324 004604 000062
1325 004606 013737 001410 001352
1326 004614 004537 006516
1327 004620 000062
1328 004622 063737 001410 001352
1329 004630 162737 000062 001352
1330 004636 013700 001420
1331 004642 006300
1332 004644 160037 001352
1333 004650 104400 013605
1334 004654 013702 001352
1335 004660 004737 011406
1336 004664 104400 013620
1337 004670 004537 011116
1338 004674 000000
1339 004676 011642
1340 004700 000401
1341 004702 000403
1342 004704 104400 012407
1343 004710 000402
1344 004712 104400 012376

```

```

*****
*TEST 31      OFFSET ON CHO
*****
TST31:  SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      BASECH,CHANL   ;LOAD CHANNEL
        MOV      BASECH,DUMMY   ;LOAD DUMMY
        MOV      #4001,EDGE
        JSR      R5,SARSUB
        SO.
        MOV      DAC,TEMP
        JSR      R5,SARSUB
        SO.
        ADD      DAC,TEMP
        SUB      #62,TEMP
        MOV      MIN,R0
        ASL      R0
        SUB      R0,TEMP
        TYPE     ,MOFSET        ;TYPE ASCIZ STRING
        MOV      TEMP,R2
        JSR      PC,DECTYP
        TYPE     ,MLSB          ;TYPE ASCIZ STRING
        JSR      R5,COMPAR      ;IS RESULT WITHIN LIMITS?
        O
        VSOD
        BR       OFFERR         ;NO-ERROR
        BR       OFFOK         ;YES-OK
        TYPE     ,ERMSG
        BR       TST32
        TYPE     ,OKMSG
        ;;GO TO NEXT TEST

```

```

1345
1346
1347
1348 004716 000004
1349 004720 012737 000001 001160
1350 004726 012737 000116 001352
1351 004734 004537 010576
1352 004740 000015
1353 004742 004537 010576
1354 004746 000007
1355 004750 004537 010576
1356 004754 000016
1357
1358
1359
1360
1361 004756 000004
1362 004760 012737 000001 001160
1363 004766 004537 006232
1364 004772 000015
1365 004774 000016
1366 004776 012737 000116 001352
1367 005004 004537 006232
1368 005010 000016
1369 005012 000015
1370
1371
1372
1373 005014 000004
1374 005016 012737 000001 001160
1375 005024 005737 001202
1376 005030 001402
1377 005032 004737 006724
1378 005036 000207
1379
1380 005040 005000
1381 005042 105300
1382 005044 001376
1383 005046 000207

```

```

*****
*TEST 32 NOISE TEST ON 8 EDGES
*****
TST32: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
MOV #116,TEMP ;DAC VALUE
JSR R5,NOI8 ;NOISE AT -FULL SCALE
15
JSR R5,NOI8 ;NOISE AT MID-RANGE
7
JSR R5,NOI8 ;NOISE AT +FULL SCALE
16

*****
*TEST 33 SETTLE TEST ON 8 EDGES
*****
TST33: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
JSR R5,SET8 ;SETTLE-POSITIVE DIRECTION
15
16
MOV #116,TEMP
JSR R5,SET8 ;SETTLE-NEGATIVE DIRECTION
16
15

*****
*TEST 34 DIFFERENTIAL LINEARITY AND RELATIVE ACCURACY TEST
*****
TST34: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST $PASS ;FIRST TIME-SKIP DIFLIN
BEQ LEND
JSR PC,DIFLIN
LEND: RTS PC ;RETURN TO TEST SECTION

DAWAIT: CLR RO
1$: DECB RO
BNE 1$
RTS PC

```

```

1384 .SBTTL CALIBRATION TEST
1385 005050 012737 005050 001370 BEGINC: MOV #BEGINC,TADDR ;TEST ADDRESS IN TADDR
1386 005056 005077 174234 CLR @STREG ;CLEAR STATUS REGISTER
1387 005062 104400 013515 TYPE HEADS ;TYPE OUT HEADING
1388 005066 005037 177776 CLR PSW
1389 005072 017700 174042 1$: MOV @SWR,RO ;READ CHANNEL FROM SWITCH REG.
1390 005076 042700 177700 BIC #177700,RO ;ISOLATE MUX BITS
1391 005102 032777 020000 174030 BIT #BIT13,@SWR ;IS BIT 13 SET?
1392 005110 001005 BNE 2$ ;;YES,SKIP TYPEOUT
1393 005112 104400 012221 TYPE CH
1394 005116 010046 MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
1395
1396 005120 104402 TYPOS ;;TYPE CHANNEL
1397 005122 002 .BYTE 2 ;;GO TYPE--OCTAL ASCII
1398 005123 000 .BYTE 0 ;;TYPE 2 DIGIT(S)
1399 005124 012777 001610 174172 2$: MOV #RETURN,@VECTOR ;;SUPPRESS LEADING ZEROS
1400 005132 000300 SWAB RO ;ADDRESS AFTER INTRPT.
1401 005134 052700 000100 BIS #BIT6,RO ;SWITCH BYTES
1402 005140 010077 174152 MOV RO,@STREG ;LOAD THE CHANNEL
1403 005144 012702 000010 MOV #10,R2 ;TYPEOUT COUNTER
1404 005150 005277 174142 3$: INC @STREG ;START CONVERSION
1405 005154 000001 WAIT ;WAIT FOR INTRPT.
1406 005156 017700 174140 MOV @ADBUFF,RO ;READ CONVERTED VALUE
1407 005162 032777 020000 173750 BIT #BIT13,@SWR ;IS BIT 13 SET?
1408 005170 001403 BEQ 4$ ;NOT SET, TYPE OUT LIST
1409 005172 010077 173744 MOV RO,@DISPLAY ;PUT VALUE IN DISPLAY FOR DISPLAY CONTROL
1410 005176 000735 BR 1$ ;REPEAT CONVERSION
1411 005200 104400 012224 4$: TYPE SPACE
1412 005204 010046 MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
1413
1414 005206 104402 TYPOS ;;PRINT OCTAL CONVERTED VALUE
1415 005210 004 .BYTE 4 ;;GO TYPE--OCTAL ASCII
1416 005211 001 .BYTE 1 ;;TYPE 4 DIGIT(S)
1417 005212 012701 010000 MOV #10000,R1 ;;TYPE LEADING ZEROS
1418 005216 005301 5$: DEC R1
1419 005220 001376 BNE 5$
1420 005222 005302 DEC R2 ;DECREMENT THE COUNTER
1421 005224 001351 BNE 3$ ;NO CARRIAGE RETURN
1422 005226 104400 001171 TYPE $CRLF ;CARRIAGE RETURN
1423 005232 000717 BR 1$ ;REPEAT CONVERSION

```

1424					.SBTTL		LOGIC TEST SECTION	
1425	005234	012737	005234	001370	BEGL:	MOV	#BEGL, TADDR	; TEST ADDRESS
1426	005242	004737	002426			JSR	PC, TESTAD	; NO OF ADDITIONAL AD'S
1427	005246	004737	002562		1\$:	JSR	PC, BEGINL	; LOGIC TESTS
1428	005252	004737	005422			JSR	PC, BUMPAD	; MORE TO TEST?
1429	005256	000773				BR	1\$; TEST NEXT A/D
1430	005260	012737	005246	011720		MOV	#1\$, AGTST	; ADDRESS FOR EOP
1431	005266	000137	011722			JMP	\$EOP	; TYPE END OF PASS
1432								
1433					.SBTTL		AUTO TEST	
1434	005272	012737	005272	001370	BEGINA:	MOV	#BEGINA, TADDR	; TEST ADDRESS
1435	005300	005037	001202			CLR	\$PASS	; CLEAR PASS COUNTER
1436	005304	004737	002426			JSR	PC, TESTAD	; NO. OF AD'S TO BE TESTED
1437	005310	004737	002562		1\$:	JSR	PC, BEGINL	; LOGIC TESTS
1438	005314	104400	012677			TYPE	.MEND	; TYPE END OF LOGIC TEST
1439	005320	013746	001316			MOV	\$STREG, -(SP)	; SAVE STREG FOR TYPEOUT
1440	005324	104402				TYPOS		; TYPE OCTAL NUMBER
1441	005326	006				.BYTE	6	; TYPE 6 DIGITS
1442	005327	001				.BYTE	1	; TYPE LEADING ZEROS
1443	005330	104400	001171			TYPE	\$.SCLF	; TYPE A CR, LF
1444	005334	004737	003712			JSR	PC, WRAP	
1445	005340	004737	005422			JSR	PC, BUMPAD	; TEST NEXT A/D
1446	005344	000761				BR	1\$; TEST NEXT AD
1447	005346	012737	005310	011720		MOV	#1\$, AGTST	; ADDRESS FOR EOP
1448	005354	000137	011722			JMP	\$EOP	; TYPE END OF PASS
1449								
1450					.SBTTL		WRAPAROUND TEST	
1451	005360	012737	005360	001370	BEGINW:	MOV	#BEGINW, TADDR	; TEST ADDRESS
1452	005366	005037	001202			CLR	\$PASS	; CLEAR PASS COUNT
1453	005372	004737	002426			JSR	PC, TESTAD	; NO. OF AD'S TO BE TESTED
1454	005376	004737	003712		1\$:	JSR	PC, WRAP	; WRAPAROUND TESTS
1455	005402	004737	005422			JSR	PC, BUMPAD	; MORE A/D'S TO BE TESTED?
1456	005406	000773				BR	1\$; YES-GO TEST NEXT AD11K
1457	005410	012737	005376	011720		MOV	#1\$, AGTST	
1458	005416	000137	011722			JMP	\$EOP	; INCREMENTS \$PASS

```

1459
1460 005422 005737 001360 .SBTTL DETERMINE IF MORE AD11K'S TO BE TESTED
1461 005426 001424 BUMPAD: TST NBEXT ;ADDITIONAL AD'S?
1462 005430 063737 001332 001316 BEQ FIXADR ;NO-INITIALIZE ADDRESSES
1463 005436 063737 001332 001320 ADD VADR,STREG ;SET UP NEW ST. REG.
1464 005444 063737 001332 001322 ADD VADR,ADST1 ;SET UP NEW ADST1
1465 005452 063737 001334 001324 ADD VADR,ADBUFF ;SET UP NEW BUFFER ADDRESS
1466 005460 063737 001334 001330 ADD VVCT,VECTOR ;SET UP NEW VECTOR
1467 005466 005077 173636 CLR VVCT,VECTR1
1468 005472 005337 001360 DEC NBEXT ;ONE LESS AD11K
1469 005476 000446 BR BYPASS
1470 005500 062716 000002 FIXADR: ADD #2,(SP)
1471 005504 013737 001250 001316 FIXONE: MOV $BASE,STREG ;RELOAD INITIAL ADDRESSES
1472 005512 013737 001250 001320 MOV $BASE,ADST1
1473 005520 013737 001250 001322 MOV $BASE,ADBUFF
1474 005526 005237 001320 INC ADST1
1475 005532 062737 000002 001322 ADD #2,ADBUFF
1476 005540 013737 001244 001324 MOV $VECT1,VECTOR
1477 005546 042737 170000 001324 BIC #170000,VECTOR
1478 005554 113737 001245 001326 MOVB $VECT1+1,BASEBR
1479 005562 105037 001327 CLR BASEBR+1 ;CLEAR HIGH BYTE
1480 005566 013737 001324 001330 MOV VECTOR,VECTR1
1481 005574 062737 000002 001330 ADD #2,VECTR1
1482 005602 005077 173522 CLR $VECTR1
1483 005606 013737 001362 001360 MOV NMBEXT,NBEXT ;RESET COUNTER
1484
1485 005614 012700 000216 .:LOAD +2 AND HALT TRAP CATCH;;
1486 005620 012701 000214 BYPASS: MOV #216,R0 ;FILL +2
1487 005624 020137 001340 1$: MOV #214,R1 ;LOAD HALT
1488 005630 001410 CMP R1,KBVECT
1489 005632 010021 BEQ 2$
1490 005634 005021 MOV R0,(R1)+
1491 005636 010100 CLR (R1)+
1492 005640 005720 MOV R1,R0
1493 005642 020027 001002 TST (R0)+
1494 005646 001366 CMP R0,#1002
1495 005650 000207 BNE 1$
1496 005652 022021 RTS PC ;TEST NEXT A/D
1497 005654 022021 2$: CMP (R0)+,(R1)+
1498 005656 000762 CMP (R0)+,(R1)+
1499 BR 1$
1500
1501 .SBTTL NOISE TEST, 1 EDGE
1502 005660 012737 005660 001370 BEGINN: MOV #BEGINN,TADDR ;TEST ADDRESS IN TADDR
1503 005666 104400 012030 TYPE ,NOIMSG ;ASK FOR CHANNEL
1504 005672 104400 013534 TYPE ,ASKCH
1505 005676 017737 173236 001354 1$: MOV $SWR,CH1 ;LOAD CHANNEL
1506 005704 042737 177700 001354 BIC #177700,CH1
1507 005712 012737 000200 001352 MOV #200,TEMP ;LOAD DAC VALUE
1508 005720 004537 010312 JSR R5,NOITST ;GO TO NOISE SUBROUTINE
1509 005724 001354 CH1
1510 005726 000763 BR 1$

```

```

1511          SBTTL      INTERCHANNEL SETTLING TEST, 1 EDGE
1512 005730 012737 005730 001370 BEGINS: MOV      #BEGINS, TADDR      ;TEST ADDRESS IN TADDR
1513 005736 104400 012050          TYPE      ,SETMSG          ;ASK FOR CHANNELS
1514 005742 104406          RDOCT
1515 005744 012637 001354          MOV      (SP)+, CH1
1516 005750 104400 012335          TYPE      ,TOMSG
1517 005754 104406          RDOCT
1518 005756 012637 001356          MOV      (SP)+, CH2
1519 005762 012737 000200 001352 BK3:  MOV      #200, TEMP          ;LOAD DAC
1520 005770 013737 001356 001366          MOV      CH2, CHANL
1521 005776 004737 006336          JSR      PC, GETEDG          ;GET EDGE VALUES
1522 006002 005002          CLR      R2
1523 006004 004737 006170          JSR      PC, SET1A          ;SCALING = .02 LSB
1524 006010 004737 006170          JSR      PC, SET1A          ;MAKE IT .01 LSB
1525 006014 100001          BPL      POSR2
1526 006016 005402          NEG      R2
1527 006020 010204          MOV      R2, R4
1528 006022 012737 000001 006514 POSR2: MOV      #1, EDGFLG
1529 006030 004737 006036          JSR      PC, TYPSET
1530 006034 000752          BR      BK3
1531 006036 004737 011406          JSR      PC, DECTYP
1532 006042 104400 012231          TYPE      LSB
1533 006046 013746 001356          MOV      CH2, -(SP)      ;; SAVE CH2 FOR TYPEOUT
1534          ;; TYPE CH
1535          ;; GO TYPE--OCTAL ASCII
1536 006052 104402          TYPOS
1537 006054 002          .BYTE 2
1538 006055 000          .BYTE 0
1539 006056 104400 013626          TYPE      MAT
1540 006062 004737 006452          JSR      PC, TYPEDG
1541 006066 104400 012244          TYPE      SETCH
1542 006072 013746 001354          MOV      CH1, -(SP)      ;; SAVE CH1 FOR TYPEOUT
1543          ;; TYPE CH
1544          ;; GO TYPE--OCTAL ASCII
1545 006076 104402          TYPOS
1546 006100 002          .BYTE 2
1547 006101 000          .BYTE 0
1548          ;; SUPPRESS LEADING ZEROS
1549          ;; TYPE ASCII STRING
1550 006102 104400 012266          TYPE      ATMSG
1551 006106 013737 001354 006134          MOV      CH1, 1$
1552 006114 163737 001336 006134          SUB      BASECH, 1$
1553 006122 012777 000200 173172          MOV      #200, JADBUFF
1554 006130 004537 011004          JSR      R5, CONVRT
1555 006134 000000          D
1556 006136 013746 001352          JS:  MOV      TEMP, -(SP)      ;; SAVE TEMP FOR TYPEOUT
1557          ;; TYPE VALUE
1558          ;; GO TYPE--OCTAL ASCII
1559 006142 104402          TYPOS
1560 006144 004          .BYTE 4
1561 006145 001          .BYTE 1
1562          ;; TYPE 4 DIGIT(S)
1563          ;; TYPE LEADING ZEROS
1564 006146 020437 011650          CMP      R4, VSET
1565 006152 003003          BGT      ERR
1566 006154 104400 012376          TYPE      OKMSG
1567 006160 000207          RTS      PC

```



```

1561 006162 104400 012407 ERR: TYPE ERMSG
1562 006166 000207 RTS PC
1563
1564
1565
1566
1567 006170 013737 001356 001364 ::SUBROUTINE FOR SETTling TESTS::
1568 006176 004537 006516 SET1A: MOV CH2,DUMMY ;LOAD DUMMY
1569 006202 000062 JSR RS,SARSUB ;DO SAR ROUTINE AT 50%
1570 006204 063702 001410 SO. ;ADD RESULT TO R2
1571 006210 013737 001354 001364 MOV CH1,DUMMY ;CHANGE DUMMY VALUE
1572 006216 004537 006516 JSR RS,SARSUB ;DO SAR ROUTINE AT 50%
1573 006222 000062 SO.
1574 006224 163702 001410 SUB DAC,R2 ;SUBTRACT RESULT FROM R2
1575 006230 000207 RTS PC ;RETURN
1576
1577 006232 012537 001354 SETB: MOV (R5)+,CH1 ;GET FIRST CHANNEL
1578 006236 012537 001356 MOV (R5)+,CH2 ;GET SECOND CHANNEL
1579 006242 063737 001336 001354 ADD BASECH,CH1
1580 006250 063737 001336 001356 ADD BASECH,CH2
1581 006256 004737 006336 JSR PC,GETEDG ;GET EDGE VALUES
1582 006262 005002 CLR R2
1583 006264 012703 000010 MOV #10,R3 ;SET UP COUNTER
1584 006270 004737 006170 SETAA: JSR PC,SET1A ;GET SETTLE VALUES
1585 006274 005237 001414 INC EDGE
1586 006300 005303 DEC R3
1587 006302 001372 BNE SETAA ;REPEAT 8 TIMES
1588 006304 162737 000010 001414 SUB #10,EDGE
1589 006312 005702 TST R2
1590 006314 100001 BPL R2POS
1591 006316 005402 NEG R2
1592 006320 010204 R2POS: MOV R2,R4
1593 006322 012737 000010 006514 MOV #8,EDGFLG
1594 006330 004737 006036 JSR PC,TYPSET ;TYPE OUT RESULTS
1595 006334 000205 RTS RS ;RETURN

```

C04

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 41
INTERCHANNEL SETTling TEST, 1 EDGE

```

1596                                     :SUBROUTINE TO GET EDGE VALUE
1597                                     :CALL=JSR PC,GETEDG
1598                                     :CONVERSIONS ON A/D CHANNEL 'CHANL'
1599                                     :RESULT IN EDGE, USES R0
1600 006336 013777 001352 172756 GETEDG: MOV TEMP,@ADBUFF
1601 006344 113700 001366          MOVB CHANL,R0
1602 006350 000300          SWAB R0
1603 006352 052700 000100          BIS #100,R0
1604 006356 010077 172734          MOV R0,@STREG
1605 006362 012700 000100          MOV #100,R0
1606 006366 005300          1$: DEC R0
1607 006370 001376          BNE 1$
1608 006372 005037 001414          CLR EDGE
1609 006376 012700 000010          MOV #10,R0
1610 006402 012777 001610 172714 CONV: MOV #RETURN,@VECTOR
1611 006410 005277 172702          INC @STREG
1612 006414 000001          WAIT
1613 006416 067737 172700 001414          ADD @ADBUFF,EDGE
1614 006424 005300          DEC R0
1615 006426 001370          BNE CONV
1616 006430 006237 001414          ASR EDGE
1617 006434 006237 001414          ASR EDGE
1618 006440 006237 001414          ASR EDGE
1619 006444 005537 001414          ADC EDGE
1620 006450 000207          RTS PC
1621
1622                                     ;;SUBROUTINE TO TYPE EDGE VALUES;;
1623 006452 013703 001414 †TYPEDG: MOV EDGE,R3
1624 006456 010346          MOV R3,-(SP)
1625                                     ;;SAVE R3 FOR TYPEOUT
1626 006460 104402          TYPOS
1627 006462 004          .BYTE 4
1628 006463 001          .BYTE 1
1629 006464 023727 006514 000001 CMP EDGFLG,#1
1630 006472 001407          BEQ RET
1631 006474 062703 000007          ADD #7,R3
1632 006500 104400 013576          TYPE C1
1633 006504 010346          MOV R3,-(SP)
1634                                     ;;TYPE ASCIZ STRING
1635 006506 104402          TYPOS
1636 006510 004          .BYTE 4
1637 006511 001          .BYTE 1
1638 006512 000207          RET: RTS
1639 006514 000000          EDGFLG: 0

```

```

1640      :SUBROUTINE TO DO SUCCESSIVE APPROXIMATION ROUTINE
1641      :CALL=JSR  R5,SARSUB
1642      :   XXX:XXX=PERCENT
1643      :RESULT RETURNED IN 'DAC' USES R0,R1,R4
1644      SARSUB: MOV      (R5)+,PERCNT      ;GET PERCENT
1645              ASL      PERCNT
1646              ASL      PERCNT
1647              ASL      PERCNT      ;RESCALE PERCENT FOR 1600.
1648              ASL      PERCNT      ;POINTS PER BURST
1649              SAR1:  MOV      #200,BITPNT ;INITIALIZE BIT POINTER AT MSB
1650              CLR      DAC              ;INITIALIZE DAC VALUE
1651              TRY:   CLR      R0
1652              ADD      BITPNT,DAC        ;TRY BIT
1653              MOV      DAC,DABUFF
1654              MOV      #100,DELAY
1655              1$:    DEC      DELAY      ;STALL TIME
1656              BNE     1$
1657              MOV      #1600,R1          ;SET UP FOR 1600. CONVERSIONS
1658              NXTCVT:MOV      DUMMY,DAST1 ;PRESET MUX TO DUMMY CHANNEL
1659              MOV      #RETURN,DVECTOR  ;RETURN ADDRESS
1660              BIS     #101,DSTREG       ;CONVERSION ON DUMMY CHANNEL
1661              WAIT
1662              MOV      DABUFF,R4        ;DUMMY READ
1663              MOV      CHANL,R4
1664              SWAB    R4
1665              BIS     #101,R4           ;INTERRUPT ENABLE START
1666              MOV      R4,DSTREG        ;JUMP TO CHANNEL + START CONVERT
1667              WAIT
1668              CMP     DABUFF,EDGE       ;WAIT FOR INTERRUPT
1669              BGE     2$
1670              INC     R0                ;COUNT RESULTS .LT. EDGE
1671              2$:    DEC      R1
1672              BNE     NXTCVT
1673              CMP     R0,PERCNT
1674              BGT     SHIFT
1675              SUB     BITPNT,DAC        ;TAKE THE BIT OUT
1676              ASR     BITPNT
1677              BNE     TRY
1678              RTS

```

```

1679      ::DIFFERENTIAL LINEARITY SUBROUTINE;;
1680 006724 104400 013022  DIFLIN: TYPE      MSG20
1681 006730 005037 001430      CLR      OUT
1682 006734 012700 017756      MOV      #BUFFER,R0
1683 006740 012701 010000      MOV      #4096.,R1      ;4096 WORDS FOR HISTOGRAM
1684 006744 005020      CLEAR1: CLR      (R0)+      ;CLEAR BUFFER AREA
1685 006746 005301      DEC      R1
1686 006750 001375      BNE     CLEAR1
1687 006752 012700 017136      MOV      #DIST,R0      ;DISTRIBUTION BUFFER POINTER
1688 006756 012701 000310      MOV      #200.,R1      ;200. WORDS FOR DISTRIBUTION
1689 006762 005003      CLR      R3
1690 006764 005037 001430      CLR      OUT
1691 006770 005037 001342      CLR      WIDE
1692 006774 005037 001344      CLR      NARROW
1693 007000 005037 001346      CLR      FIRST
1694 007004 005037 001350      CLR      SKIPST
1695 007010 005020      CLEAR2: CLR      (R0)+      ;CLEAR DISTRIBUTION BUFFER AREA
1696 007012 005301      DEC      R1
1697 007014 001375      BNE     CLEAR2
1698 007016 012700 000011      CHANNL: MOV     #11,R0      ;CHANNEL 11
1699 007022 063700 001336      ADD     BASECH,R0
1700 007026 000300      SWAB    R0      ;LOAD MUX BITS
1701 007030 052700 000100      BIS     #100,R0
1702 007034 010077 172256      MOV     R0,ASTREG
1703 007040 012700 001440      MOV     #800.,R0      ;NOMINAL STATE WIDTH - 1 LSB
1704 007044 012777 001610 172252      MOV     #RETURN,VECTOR
1705 007052 012701 007776      AGAIN: MOV     #4094.,R1
1706 007056 004737 010722      NEXT:  JSR     PC,RANDY      ;GET RANDOM NUMBER
1707 007062 013702 001372      MOV     R1A,R2
1708 007066 042702 177760      BIC     #177760,R2      ;MASK IT TO 4 BITS ONLY
1709 007072 001402      BEQ     CONVR
1710 007074 005302      DELAY3: DEC     R2      ;STALL
1711 007076 001376      BNE     DELAY3      ;TIME
1712 007100 005277 172212      CONVR: INC     ASTREG      ;START CONVERSION
1713 007104 000001      WAIT
1714 007106 017702 172210      MOV     ADDBUFF,R2      ;GET CONVERTED VALUE
1715 007112 001413      BEQ     DELAY1      ;IGNORE IF =0
1716 007114 020227 007777      CMP     R2,#7777      ;IGNORE IF =7777
1717 007120 001413      BEQ     DELAY2
1718 007122 006302      ASL    R2
1719 007124 005262 017756      INC     BUFFER(R2)      ;MAKE HISTOGRAM
1720 007130 100013      BPL    OKAY
1721 007132 012762 077777 017756      MOV     #077777,BUFFER(R2) ;PREVENT OVERFLOW
1722 007140 000407      BR     OKAY
1723 007142 020227 007777      DELAY1: CMP     R2,#7777      ;EQUALIZE LOOP TIME
1724 007146 001400      BEQ     DELAY2      ;WITH DUMMY INSTR.
1725 007150 005201      DELAY2: INC     R1
1726 007152 005263 001352      INC     TEMP(R3)
1727 007156 100403      BMI    NOTOK
1728 007160 005301      OKAY:  DEC     R1
1729 007162 001335      BNE    NEXT
1730 007164 000403      BR     AROUND
1731 007166 005037 001352      NOTOK: CLR     TEMP
1732 007172 000772      BR     OKAY

```

```

1733 007174 005300          AROUND: DEC      R0
1734 007176 001325          BNE      AGAIN
1735 007200 012700 007776   MOV      #4094, R0
1736 007204 012701 017760   MOV      #BUFFER+2, R1
1737 007210 012102          READ:  MOV      (R1)+, R2      ;GET STATE WIDTH
1738 007212 006202          ASR     R2      ;1 LSB = 800.
1739 007214 006202          ASR     R2
1740 007216 006202          ASR     R2
1741 007220 005502          ADC     R2      ;1 LSB = 100.
1742 007222 020227 000310   CMP     R2, #200. ;OUT OF RANGE?
1743 007226 002403          BLT     INRNGE
1744 007230 005237 001430   INC     OUT      ;YES - INCREMENT COUNTER
1745 007234 000423          BR      TYPBAD
1746 007236 006302          INRNGE: ASL     R2
1747 007240 005262 017136   INC     DIST(R2) ;MAKE STATE WIDTH DISTRIBUTION
1748 007244 006202          ASR     R2
1749 007246 020227 000062   CMP     R2, #50. ;IS IT 1/2 LSB?
1750 007252 002007          BGE     NOTNAR
1751 007254 005237 001344   INC     NARROW
1752 007260 005702          TST     R2      ;IS IT A SKIPPED STATE?
1753 007262 001002          BNE     31$
1754 007264 005237 001350   INC     SKIPST
1755 007270 000405          31$:  BR      TYPBAD
1756 007272 020227 000226   NOTNAR: CMP    R2, #150. ;IS IT 1.5 LSB?
1757 007276 003426          BLE     LAST
1758 007300 005237 001342          INC     WIDE
1759 007304 005737 001346   TYPBAD: TST    FIRST
1760 007310 001004          BNE     60$
1761 007312 005237 001346          INC     FIRST
1762 007316 104400 012201          TYPE   STATE
1763 007322 010103          60$:  MOV     R1, R3
1764 007324 162703 017760   SUB     #BUFFER+2, R3
1765 007330 006203          ASR     R3
1766 007332 010346          MOV     R3, -(SP)      ;;SAVE R3 FOR TYPEOUT
1767                                ;;TYPE STATE
1768 007334 104402          TYPOS  4      ;;GO TYPE--OCTAL ASCII
1769 007336 004      .BYTE  4      ;;TYPE 4 DIGIT(S)
1770 007337 001      .BYTE  1      ;;TYPE LEADING ZEROS
1771 007340 104400 012175          TYPE   DASH
1772 007344 004737 011406          JSR    PC, DECTYP
1773 007350 104400 012166          TYPE   LSBMSG
1774 007354 005300          LAST:  DEC     R0
1775 007356 001314          BNE     READ
1776 007360 112737 000177 014465  MOVB   #177, DECPNT
1777 007366 013702 001350          MOV    SKIPST, R2      ;GET NO. OF SKIPPED STATES
1778 007372 004737 011406          JSR    PC, DECTYP      ;TYPE IT
1779 007376 104400 012424          TYPE   SKPMSG      ;TYPE MESSAGE
1780 007402 005737 001350          TST    SKIPST
1781 007406 001403          BEQ    1$
1782 007410 104400 012407          TYPE   ,ERMSG      ;TYPE "ERROR"
1783 007414 000402          BR     NAR
1784 007416 104400 012376          1$:  TYPE   ,OKMSG      ;TYPE #OK#

```

```

1785 007422 013702 001344      NAR:  MOV  NARROW,R2      ;GET NO. OF NARROW STATES
1786 007426 004737 011406      JSR  PC,DECTYP      ;TYPE IT
1787 007432 104400 012446      TYPE ,NARMSG      ;TYPE MESSAGE
1788 007436 013702 001342      MOV  WIDE,R2
1789 007442 C63702 001430      ADD  OUT,R2
1790 007446 004737 011406      JSR  PC,DECTYP      ;TYPE NO. OF WIDE STATES
1791 007452 104400 012505      TYPE ,WIDMSG      ;TYPE MESSAGE
1792 007456 013702 001430      MOV  OUT,R2
1793 007462 004737 011406      JSR  PC,DECTYP      ;TYPE NO. OF STATES OUTSIDE 2 LSB
1794 007466 104400 012544      TYPE ,OUTMSG      ;TYPE MESSAGE
1795 007472 005737 001430      TST  OUT
1796 007476 001403      BEQ  11$
1797 007500 104400 012407      TYPE ,ERMSG      ;TYPE "ERROR"
1798 007504 000402      BR   HALF
1799 007506 104400 012376      11$: TYPE ,OKMSG      ;TYPE "OK"
1800 007512 013702 001344      HALF: MOV  NARROW,R2
1801 007516 063702 001342      ADD  WIDE,R2
1802 007522 063702 001430      ADD  OUT,R2
1803 007526 010200      MOV  R2,R0
1804 007530 004737 011406      JSR  PC,DECTYP      ;TYPE NO. OF STATES OUTSIDE LIMITS
1805 007534 112737 000056 014465  MOVB  #56,DECPNT
1806 007542 104400 012577      TYPE ,HAFMSG
1807 007546 020027 000051      CMP  R0,#41.      ;COMPARE IT TO NOMINAL
1808 007552 003403      BLE  21$
1809 007554 104400 012407      TYPE ,ERMSG      ;TYPE "ERROR"
1810 007560 000402      BR   SWDIST
1811 007562 104400 012376      21$: TYPE ,OKMSG      ;TYPE "OK"
1812 007566 005737 001404      SWDIST: TST  FLAG      ;VT55?
1813 007572 001426      BEQ  RELACC
1814 007574 004737 010254      JSR  PC,DELCLR      ;WAIT AWHILE, THEN CLEAR VT55
1815 007600 104400 013054      TYPE ,MSG16
1816 007604 104400 013655      TYPE ,BUFF1      ;TYPE BUFF1-PRINT GRID
1817 007610 012700 017136      MOV  #DIST,R0      ;POINTER TO STATE WIDTH DISTRIBUTION
1818 007614 012701 000310      MOV  #200.,R1      ;GO 200. TIMES UP TO 2 LSB
1819 007620 012002      NXTY1: MOV  (R0)+,R2
1820 007622 004737 011304      JSR  PC,LOADY
1821 007626 005002      CLR  R2
1822 007630 004737 011304      JSR  PC,LOADY
1823 007634 005301      DEC  R1
1824 007636 001370      BNE  NXTY1
1825 007640 104400 013600      TYPE ,C2      ;TYPE ASCIZ STRING
1826 007644 004737 010254      JSR  PC,DELCLR
1827

```

H04

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 46
INTERCHANNEL SETTLING TEST, 1 EDGE

```

1828                                     ;CHANGE HISTOGRAM ERROR TO RELATIVE ACCURACY ERROR
1829
1830 007650 005001 RELACC: CLR R1 ;RUNNING ERROR = 0
1831 007652 005003 CLR R3 ;MAXIMUM ERROR = 0
1832 007654 104400 013447 TYPE ,MSG21
1833 007660 012700 017760 MOV #BUFFER+2,R0
1834 007664 011002 NXTSTA: MOV (R0),R2 ;STATE WIDTH = R2
1835 007666 162702 001440 SUB #800.,R2 ;STATE WIDTH ERROR IN R2
1836 007672 060201 ADD R2,R1 ;UPDATE RUNNING ERROR
1837 007674 010120 MOV R1,(R0)+ ;SAVE IN BUFFER
1838 007676 010104 MOV R1,R4 ;SAVE IN R4 ALSO
1839 007700 100001 BPL PLUS ;IS IT POSITIVE?
1840 007702 005404 NEG R4 ;NO - MAKE IT POSITIVE
1841 007704 020403 PLUS: CMP R4,R3 ;CHECK AGAINST PREVIOUS MAX. ERROR
1842 007706 003405 BLE NOTNEW ;NOT A NEW MAXIMUM
1843 007710 010403 MOV R4,R3 ;UPDATE MAXIMUM IN R3
1844 007712 010005 MOV R0,R5
1845 007714 162705 017760 SUB #BUFFER+2,R5
1846 007720 006205 ASR R5 ;R5=EDGE VALUE AT MAX. RELACC
1847 007722 020027 037754 NOTNEW: CMP R0,#BUFFER+8190. ;DONE?
1848 007726 001356 BNE NXTSTA ;NO - REPEAT
1849 007730 006203 ASR R3 ;RESCALE FROM 1 LSB = 800. SCALING
1850 007732 006203 ASR R3 ;TO 1 LSB = 100. SCALING
1851 007734 006203 ASR R3
1852 007736 005503 ADC R3
1853 007740 010302 MOV R3,R2
1854 007742 004737 011406 JSR PC,DECTYP
1855 007746 104400 013474 TYPE LINEA
1856 007752 010546 MOV R5,-(SP) ;:SAVE R5 FOR TYPEOUT
1857 ;:TYPE VALUE
1858 007754 104402 TYPOS ;:GO TYPE--OCTAL ASCII
1859 007756 004 .BYTE 4 ;:TYPE 4 DIGIT(S)
1860 007757 001 .BYTE 1 ;:TYPE LEADING ZEROS
1861 007760 104400 012333 TYPE ,SLASH ;:PRINT '/'
1862 007764 005205 INC R5
1863 007766 010546 MOV R5,-(SP) ;:SAVE R5 FOR TYPEOUT
1864 ;:TYPE VALUE
1865 007770 104402 TYPOS ;:GO TYPE--OCTAL ASCII
1866 007772 004 .BYTE 4 ;:TYPE 4 DIGIT(S)
1867 007773 001 .BYTE 1 ;:TYPE LEADING ZEROS
1868 007774 020337 011652 CMP R3,VLIN
1869 010000 003403 BLE 41$
1870 010002 104400 012407 TYPE ,ERMSG
1871 010006 000402 BR 42$
1872 010010 104400 012376 41$: TYPE ,OKMSG
1873 010014 005737 001404 42$: TST FLAG ;VT55?
1874 010020 001503 BEQ L02
1875 010022 012700 017756 MOV #BUFFER,R0
1876 010026 012701 010000 MOV #4096.,R1

```

1877	010032	011002		GETDAT:	MOV	(R0),R2		;GET RELATIVE ACCURACY ERROR SCALED 1LSB = 800.
1878	010034	006202			ASR	R2		;RESCALE IT TO 1 LSB = 100.
1879	010036	006202			ASR	R2		
1880	010040	006202			ASR	R2		
1881	010042	005502			ADC	R2		
1882	010044	062702	000166		ADD	#118.,R2		;AND MOVE IT TO MID-SCREEN
1883	010050	010220			MOV	R2,(R0)+		;PUT IT BACK INTO BUFFER
1884	010052	005301			DEC	R1		
1885	010054	001366			BNE	GETDAT		
1886	010056	012700	017756		MOV	#BUFFER,R0		
1887	010062	012704	017756		MOV	#BUFFER,R4		
1888	010066	012705	017760		MOV	#BUFFER+2,R5		
1889	010072	012701	001000		MOV	#512.,R1		
1890	010076	012702	000007	NXT8:	MOV	#7.,R2		
1891	010102	012003			MOV	(R0)+,R3		
1892	010104	010337	001420		MOV	R3,MIN		;MINIMUM
1893	010110	010337	001424		MOV	R3,MAX		;MAXIMUM
1894	010114	012003		NXTCMP:	MOV	(R0)+,R3		
1895	010116	020337	001420		CMP	R3,MIN		
1896	010122	002002			BGE	MAXTST		
1897	010124	010337	001420		MOV	R3,MIN		;NEW MINIMUM
1898	010130	020337	001424	MAXTST:	CMP	R3,MAX		
1899	010134	003402			BLE	TST8		
1900	010136	010337	001424		MOV	R3,MAX		;NEW MAXIMUM
1901	010142	005302		TST8:	DEC	R2		
1902	010144	001363			BNE	NXTCMP		
1903	010146	013724	001420		MOV	MIN,(R4)+		
1904	010152	013725	001424		MOV	MAX,(R5)+		
1905	010156	022425			CMP	(R4)+,(R5)+		;BUMP EACH ONCE MORE
1906	010160	005301			DEC	R1		
1907	010162	001345			BNE	NXT8		
1908	010164	104400	012762		TYPE	,MSG18		
1909	010170	104400	013703		TYPE	,BUFF2		;TYPE BUFF2
1910	010174	012700	017756		MOV	#BUFFER,R0		
1911	010200	004737	010232		JSR	PC,LOAD		
1912	010204	104400	013603		TYPE	,C3		;TYPE ASCIZ STRING
1913	010210	012700	017760		MOV	#BUFFER+2,R0		
1914	010214	004737	010232		JSR	PC,LOAD		
1915	010220	104400	013600		TYPE	,C2		;TYPE ASCIZ STRING
1916	010224	004737	010254		JSR	PC,DELCLR		
1917	010230	000207		LO2:	RTS	PC		
1918	010232	012701	001000	LOAD:	MOV	#512.,R1		
1919	010236	012002		LOADO:	MOV	(R0)+,R2		
1920	010240	005720			TST	(R0)+		
1921	010242	004737	011304		JSR	PC,LOADY		
1922	010246	005301			DEC	R1		
1923	010250	001372			BNE	LOADO		
1924	010252	000207			RTS	PC		

J04

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 48
INTERCHANNEL SETTLING TEST, 1 EDGE

```

1925 010254 005000          DELCLR: CLR      RO
1926 010256 012701 000020          MOV      #20,R1      ;DELAY BEFORE CLEANING SCREEN
1927 010262 005300          1$:      DEC      RO
1928 010264 001376          BNE      1$
1929 010266 005301          DEC      R1
1930 010270 001374          BNE      1$
1931 010272 032777 010000 170640          BIT      #BIT12,DSWR      ;TEST FOR HALT FOR DISPLAY
1932 010300 001401          BEQ      2$          ;;DON'T HALT FOR DISPLAY
1933 010302 000000          HALT
1934 010304 104400 013723          2$:      TYPE     VTINIT
1935 010310 000207          RTS      PC
1936          ;;NOISE SUBROUTINE;;
1937 010312 013537 001366          NOITST: MOV      @R5)+,CHANL      ;LOAD CHANNEL
1938 010316 013737 001366 001364          MOV      CHANL,DUMMY      ;LOAD DUMMY CHANNEL
1939 010324 004737 006336          JSR      PC,GETEDG      ;GET EDGE VALUE
1940 010330 004737 010504          JSR      PC,NOIA      ;GET RMS AND PEAK VALUES
1941 010334 012737 000001 006514          MOV      #1,EDGFLG
1942 010342 004737 010350          JSR      PC,TYPRP      ;TYPE RMS AND PEAK VALUES
1943 010346 000205          RTS      R5
1944
1945
1946
1947
1948
1949          ;;TYPE RMS AND PEAK VALUES;;
1950 010350 104400 012273          TYPRP:  TYPE     NOI
1951 010354 005737 001400          TST      RMS
1952 010360 100002          BPL      POSRMS
1953 010362 005037 001400          CLR      RMS          ;RMS<0,SET RMS=0
1954 010366 005737 001402          POSRMS: TST      PEAK
1955 010372 100002          BPL      POSPEA
1956 010374 005037 001402          CLR      PEAK          ;PEAK<0,SET PEAK=0
1957 010400 013702 001400          POSPEA: MOV      RMS,R2
1958 010404 004737 011406          JSR      PC,DECTYP
1959 010410 104400 012646          TYPE     MESR
1960 010414 013702 001402          MOV      PEAK,R2
1961 010420 004737 011406          JSR      PC,DECTYP
1962 010424 104400 012661          TYPE     MESP
1963 010430 004737 006452          JSR      PC,TYPEDG
1964 010434 104400 012303          TYPE     CHAN
1965 010440 013746 001366          MOV      CHANL,-(SP)      ;;SAVE CHANL FOR TYPEOUT
1966          ;;TYPE CHANL
1967 010444 104402          TYPOS
1968 010446 002          .BYTE 2
1969 010447 000          .BYTE 0
1970 010450 023737 001400 011644          CMP      RMS,VNR      ;;GO TYPE--OCTAL ASCII
1971 010456 003007          BGT      ER          ;;TYPE 2 DIGIT(S)
1972 010460 023737 001402 011646          CMP      PEAK,VNP      ;;SUPPRESS LEADING ZEROS
1973 010466 003003          BGT      ER          ;WITHIN LIMITS?
1974 010470 104400 012376          TYPE     OKMSG
1975 010474 000207          RTS      PC
1976 010476 104400 012407          ER:      TYPE     ERMSG
1977 010502 000207          RTS      PC

```

```

1978                                     ::SUBROUTINES FOR NOISE TEST;;
1979 010504 005037 001400 NOIA: CLR RMS ;CLEAR RMS VLAUE
1980 010510 005037 001402 CLR PEAK ;CLEAR PEAK VALUE
1981 010514 004537 006516 NOI1: JSR R5,SAR SUB ;DO SAR ROUTINE AT 16%
1982 010520 000020 16.
1983 010522 063737 001410 001400 ADD DAC,RMS ;ADD RESULT TO RMS
1984 010530 004537 006516 JSR R5,SAR SUB ;DO SAR ROUTINE AT 84%
1985 010534 000124 84.
1986 010536 163737 001410 001400 SUB DAC,RMS ;SUBTRACT RESULT FROM RMS
1987 010544 004537 006516 JSR R5,SAR SUB ;DO SAR ROUTINE AT 1%
1988 010550 000001 1.
1989 010552 063737 001410 001402 ADD DAC,PEAK ;ADD RESULT TO PEAK
1990 010560 004537 006516 JSR R5,SAR SUB ;DO SAR ROUTINE AT 99%
1991 010564 000143 99.
1992 010566 163737 001410 001402 SUB DAC,PEAK ;SUBTRACT RESULT FROM PEAK
1993 010574 000207 RTS PC ;RETURN
1994
1995 010576 012537 001366 NOI8: MOV (R5)+,CHANL ;GET CHANNEL VALUE
1996 010602 063737 001336 001366 ADD BASECH,CHANL
1997 010610 013737 001366 001364 MOV CHANL,DUMMY ;LOAD DUMMY CHANNEL
1998 010616 004737 006336 JSR PC,GETEDG ;GET EDGE VALUES
1999 010622 005037 001400 CLR RMS ;CLEAR RMS VALUE
2000 010626 005037 001402 CLR PEAK ;CLEAR PEAK VALUE
2001 010632 012737 000010 010720 MOV #10,10$ ;SET UP COUNTER
2002 010640 004737 010514 1$: JSR PC,NOI1 ;GET NOISE VALUES
2003 010644 005237 001414 INC EDGE
2004 010650 005337 010720 DEC 10$
2005 010654 001371 BNE 1$ ;REPEAT 8 TIMES
2006 010656 162737 000010 001414 SUB #10,EDGE ;SCALE IT TO 1 LSB=100.
2007 010664 006237 001400 ASR RMS
2008 010670 005537 001400 ADC RMS
2009 010674 006237 001402 ASR PEAK
2010 010700 005537 001402 ADC PEAK
2011 010704 012737 000010 006514 MOV #8,EDGFLG
2012 010712 004737 010350 JSR PC,TYPRP ;RETURN TYPE RESULTS
2013 010716 000205 RTS R5
2014 010720 000000 10$: 0 ;COUNTER
2015
2016
2017                                     ::RANDOM NUMBER GENERATOR;;
2018 010722 063737 001374 001372 RANDY: ADD RNB,RNA
2019 010730 063737 001376 001372 ADD RNC,RNA
2020 010736 005537 001372 ADC RNA
2021 010742 063737 001372 001374 ADD RNA,RNB
2022 010750 063737 001376 001374 ADD RNC,RNB
2023 010756 005537 001374 ADC RNB
2024 010762 063737 001372 001376 ADD RNA,RNC
2025 010770 063737 001374 001376 ADD RNB,RNC
2026 010776 005537 001376 ADC RNC
2027 011002 000207 RTS PC

```

```

2028      ::ROUTINE TO AVERAGE 8 CONVERSIONS;;
2029      CONVRT: MOV      (R5)+,R0      ;GET CHANNEL VALUE
2030      ADD      BASECH,R0
2031      MOV      R0,CHANL
2032      SWAB     R0
2033      CLR      TEMP
2034      MOV      R0,@STREG      ;LOAD CHANNEL INTO MIX BITS
2035      MOV      #10000,R0
2036      2$:     DEC      R0
2037      BNE     2$
2038      MOV      #RETURN,@VECTOR    ;LOAD VECTOR
2039      MOV      #10,R0             ;SET UP COUNTER
2040      1$:     BISB     #101,@STREG ;SET INTRPT. EN., START CONV.
2041      WAIT
2042      ADD      @ADBUFF,TEMP      ;READ BUFFER
2043      DEC      R0
2044      BNE     1$                 ;DO 8 TIMES
2045      ASR      TEMP              ;AVERAGE VALUE
2046      ASR      TEMP
2047      ASR      TEMP
2048      ADC      TEMP
2049      RTS      R5                ;RETURN
2050
2051      :COMPARE $GDDAT AND $BDDAT::
2052      COMPAR: MOV      (R5)+,$GDDAT ;GET GOOD DATA
2053      MOV      @R5+,SPREAD ;GET SPREAD
2054      MOV      TEMP,$BDDAT ;GET BAD(ACTUAL) DATA
2055      MOV      $BDDAT,R1
2056      MOV      $GDDAT,R0
2057      SUB      R1,R0              ;GET DIFFERENCE
2058      BPL     7$
2059      NEG      R0
2060      7$:     CMP      R0,SPREAD    ;COMPARE IT TO SPREAD
2061      BGT     10$                ;GO TO ERROR PRINTOUT
2062      TST     (R5)+              ;BUMP RETURN POINTER AROUND ERROR CALL
2063      10$:    RTS

```

```

2064                                     ;;SUBROUTINE TO TYPE INTRPT. TST MSG.;;
2065 011164 005737 001202                DUMW: TST $PASS
2066 011170 001021                       BNE 20$
2067 011172 012737 011234 001110        MOV #20$, $LPERR
2068 011200 012737 011234 001106        MOV #20$, $LPADR
2069 011206 104400 013633                TYPE MET$T
2070 011212 010046                       MOV R0, -(SP)
                                           ;;TYPE ASCIZ STRING
                                           ;;SAVE R0 FOR TYPEOUT
2071                                     ;;TYPE TEST NO.
2072 011214 104402                       TYPOS
                                           ;;GO TYPE--OCTAL ASCII
2073 011216 002                          .BYTE 2
                                           ;;TYPE 2 DIGIT(S)
2074 011217 000                          .BYTE 0
                                           ;;SUPPRESS LEADING ZEROS
2075 011220 104400 012723                TYPE ONAD
2076 011224 013746 001316                MOV $STREG, -(SP)
                                           ;;SAVE STREG FOR TYPEOUT
2077                                     ;;TYPE BUS ADDRESS
2078 011230 104402                       TYPOS
                                           ;;GO TYPE--OCTAL ASCII
2079 011232 006                          .BYTE 6
                                           ;;TYPE 6 DIGITS
2080 011233 001                          .BYTE 1
                                           ;;TYPE LEADING ZEROS
2081 011234 000207                       20$: RTS PC
2082
2083 011236 005737 001202                DUMC: TST $PASS
2084 011242 001010                       BNE 30$
2085 011244 012737 011264 001110        MOV #30$, $LPERR
2086 011252 012737 011264 001106        MOV #30$, $LPADR
2087 011260 104400 012320                TYPE DONE
2088 011264 000207                       30$: RTS PC

```

```

2089          ;SUBROUTINE TO RESET & SET INTRPT. EN.;
2090 011266 000005 RST: RESET
2091 011270 052777 000100 167646 BIS #100, @STKS
2092 011276 005037 177776 CLR PSW
2093 011302 000207 RTS PC
2094
2095          ;SUBROUTINE LOADY:
2096 011304 005702 LOADY: TST R2          ;ROUTINE TO LOAD VLAUE INTO R2
2097 011306 100001 BPL PLUSR2      ;AS A VT55 Y-VALUE
2098 011310 005002 CLR R2
2099 011312 020227 000353 PLUSR2: CMP R2, #235.
2100 011316 002402 BLT LESS
2101 011320 012702 000353 MOV #235., R2
2102 011324 010203 LESS: MOV R2, R3
2103 011326 042702 177740 BIC #177740, R2
2104 011332 052702 000040 BIS #40, R2
2105 011336 105777 167606 B10: TSTB @STPS      ;PRINT CHARACTER
2106 011342 100375 BPL B10
2107 011344 110277 167602 MOVB R2, @STPB
2108 011350 006203 ASR R3
2109 011352 006203 ASR R3
2110 011354 006203 ASR R3
2111 011356 006203 ASR R3
2112 011360 006203 ASR R3
2113 011362 042703 177770 BIC #177770, R3
2114 011366 052703 000040 BIS #40, R3
2115 011372 105777 167552 B11: TSTB @STPS      ;PRINT CHARACTER
2116 011376 100375 BPL B11
2117 011400 110377 167546 MOVB R3, @STPB
2118 011404 000207 RTS PC
2119
2120

```

```

011406 005702
011410 100003
011412 104400 012135
011414 005402
011416 020227 001747
011418 003402
011420 012702 001747
011422 105037 014467
011424 105037 014466
011426 105037 014464
011428 005702
011430 001424
011432 005302
011434 105237 014467
011436 123727 014467 000012
011438 001367
011440 105037 014467
011442 105237 014466
011444 123727 014466 000012
011446 001357
011448 105037 014466
011450 105237 014464
011452 000752
011454 152737 000060 014464
011456 152737 000060 014466
011458 152737 000060 014467
011460 104400 014464
011462 000207
011552 012701 011644
011554 005737 001336
011556 001403
011558 012702 011676
011560 000410
011562 005737 001422
011564 001003
011566 012702 011656
011568 000402
011570 012702 011666
011572 012221
011574 005711
011576 100375
011578 000207

```

```

::SUBROUTINE TO TYPE DECIMAL VALUE;;
::IN R2 AS X.XX;;
DECTYP: TST R2 ;TEST VALUE TO BE TYPED
        BPL POS ;TYPE MINUS SIGN
        TYPE MINUS
        NEG R2 ;>999. REPLACE IT WITH 999.
        POS: CMP R2,#999.
        BLE OKAYD
        MOV #999.,R2
        OKAYD: CLRB ONES ;CLEAR ONES
              CLRB TENS ;CLEAR TENS
              CLRB HUNS ;CLEAR HUNS
        TESTR2: TST R2 ;CONVERT VALUE TO A DECIMAL VALUE
              BEQ TYP0UT
              DEC R2
              INCB ONES
              CMPB ONES,#10.
              BNE TESTR2
              CLRB ONES
              INCB TENS
              CMPB TENS,#10.
              BNE TESTR2
              CLRB TENS
              INCB HUNS
              BR TESTR2
        TYP0UT: BLSB #60,HUNS ;PREPARE FOR TYP0UT
              BLSB #60,TENS
              BLSB #60,ONES
              TYPE HUNS
              RTS PC ;TYPE VALUE

WFADJ: MOV #VNR,R1 ;SUBROUTINE TO SET UP LIMITS
        TST BASECH ;TESTING AN AM11K?
        BEQ 1$ ;;
        MOV #VARLT3,R2 ;BASECH NOT ZERO, USE AM11K LIMITS
        BR 3$ ;;
        1$: TST WFTEST
           BNE 2$
           MOV #VARLT1,R2 ;WFTEST=0,USE NORMAL LIMITS
           BR 3$
        2$: MOV #VARLT2,R2 ;WFTEST=1,USE OPTION AREA LIMITS
        3$: MOV (R2)+,(R1)+
           TST (R1)
           BPL 3$
           RTS PC

```

; TOLERANCE VALUES FOR FUNCTIONAL TESTS

2166	011622	000001
2167	011624	000002
2168	011626	000010
2169	011630	000050
2170	011632	000144
2171	011634	000115
2172	011636	000240
2173	011640	000005
2174	011642	000062
2175		
2176	011644	000000
2177	011646	000000
2178	011650	000000
2179	011652	000000
2180	011654	100000
2181		
2182	011656	000031
2183	011660	000310
2184	011662	000144
2185	011664	000144
2186		
2187	011666	000027
2188	011670	000226
2189	011672	000132
2190	011674	000132
2191		
2192	011676	000062
2193	011700	000310
2194	011702	000226
2195	011704	000226
2196		
2197	011706	052777
2198	011714	000177
2199	011720	001620

V1:	1
V2:	2
V10:	10
V50:	50
V144:	144
V115:	115
V240:	240
V5:	5
V500:	50.
VNR:	0
VNP:	0
VSET:	0
VLIN:	0
	BIT15
VARLT1:	25.
	200.
	100.
	100.
VARLT2:	23.
	150.
	90.
	90.
VARLT3:	50.
	200.
	150.
	150.
AGATST:	BIS
	JMP
AGTST:	BEGIN

; RMS NOISE LIMIT
; PEAK NOISE LIMIT
; INTER-CHANNEL SETTLING LIMIT
; RELATIVE ACCURACY ERROR LIMIT

; 25 LSB, NORMAL LIMITS FOR SYSTEM
; 2. LSB, INTEGRATION AND FIELD USE ON SPEC TESTS
; 1 LSB
; 1 LSB

; 23 LSB, TIGHTER LIMITS FOR OPTION
; 1.5 LSB, AREA USE ON SPEC TESTS
; .9 LSB
; .9 LSB

; 5 LSB, LIMITS FOR AMI1K TESTING
; 2. LSB
; 1.5 LSB
; 1.5 LSB

000100 167230 #100,0\$TKS
000000 @AGTST

```

2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210 011722
2211 011722 000240
2212 011724 005037 001102
2213 011730 005037 001160
2214 011734 005237 001202
2215 011740 042737 100000 001202
2216 011746 005327
2217 011750 000001
2218 011752 003015
2219 011754 012737
2220 011756 000001
2221 011760 011750
2222 011762 104400 012015
2223 011766 013700 000042
2224 011772 001405
2225 011774 000005
2226 011776 004710
2227 012000 000240
2228 012002 000240
2229 012004 000240
2230 012006
2231 012006 000137
2232 012010 011706
2233 012012 377 377 000
2234 012015 015 042412 042116
2235 012022 050040 051501 000123
2236

```

.SBTTL END OF PASS ROUTINE

```

*****
*INCREMENT THE PASS NUMBER ($PASS)
*TYPE "END PASS"
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO AGATST
*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
*SENDMG CAN BE CHANGED TO 7.

```

SEOP:

```

NOP
CLR $STNM ;; ZERO THE TEST NUMBER
CLR $TIMES ;; ZERO THE NUMBER OF ITERATIONS
INC $PASS ;; INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;; LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,2(PC)+ ;; RESTORE COUNTER
SENDCT: .WORD 1
SEOPCT
TYPE $SENDMG ;; TYPE "END PASS"
$GET42: MOV #42,R0 ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
RESET ;; CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11
$DOAGN: JMP 2(PC)+ ;; RETURN
SRTNAD: .WORD AGATST
SENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
SENDMG: .ASCIZ <15><12>/END PASS/

```


MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 56
ASCII MESSAGES

2237				
2238	012030	005015	047516	051511
2239	012036	020105	042524	052123
2240	012044	026455	000040	
2241	012050	005015	042523	052124
2242	012056	044514	043516	052040
2243	012064	051505	026524	020055
2244	012072	054524	042520	042040
2245	012100	051505	051111	042105
2246	012106	023440	051106	046517
2247	012114	020047	044103	047101
2248	012122	042516	020114	020046
2249	012130	051103	020072	000
2250	012137	055	000	
2251	012141	077	000	
2252	012144	136	101	040
2253	012144	040	000	
2254	012146	136	103	040
2255	012151	040	000	
2256	012153	136	107	015
2257	012156	012	123	127
2258	012161	122	105	107
2259	012164	072	000	
2260	012166	046040	041123	005015
2261	012174	000		
2262	012175	055	020055	000
2263	012201	123	040524	042524
2264	012206	026455	053440	042111
2265	012214	044124	005015	000
2266	012221	103	000110	
2267	012224	020040	020040	000
2268	012231	040	051514	020102
2269	012236	047117	041440	000110
2270	012244	051440	052105	046124
2271	012252	047111	020107	051106
2272	012260	046517	041440	050110
2273	012266	040440	020124	000
2274	012273	116	044517	042523
2275	012300	020072	000	
2276	012303	040	047117	041440
2277	012310	040510	047116	046105
2278	012316	000040		
2279	012320	020040	020040	047504
2280	012326	042516	005015	000
2281	012333	057	000	
2282	012335	124	050131	020105
2283	012342	042504	044523	042522
2284	012350	020104	052047	023517
2285	012356	041440	040510	047116
2286	012364	046105	023040	041440
2287	012372	035122	000040	
2288	012376	020040	020040	045517
2289	012404	005015	000	

```

.SBTTL      ASCII MESSAGES
NOIMSG: .ASCIZ  <15><12>/NOISE TEST-- /

SETMSG: .ASCIZ  <15><12>/SETTLING TEST-- TYPE DESIRED 'FROM' CHANNEL & CR: /

MINUS:  .BYTE   55,0
QUEST:  .BYTE   77,0
AMSG:   .BYTE  136,101,40,40,0
CMMSG:  .BYTE  136,103,40,40,0
GMSG:   .BYTE  136,107,15,12,123,127,122,105,107,72.0

LSBMSG:  .ASCIZ  / LSB/<15><12>

DASH:    .ASCIZ  /-- /
STATE:   .ASCIZ  /STATE-- WIDTH/<15><12>

CH:      .ASCIZ  /CH/
SPACE:   .ASCIZ  / /
LSB:     .ASCIZ  / LSB ON CH/

SETCH:   .ASCIZ  / SETTLING FROM CH/

ATMSG:   .ASCIZ  / AT /
NOI:     .ASCIZ  /NOISE: /

CHAN:    .ASCIZ  / ON CHANNEL /

DONE:    .ASCIZ  /  DONE/<15><12>

SLASH:   .ASCIZ  #/#
TOMSG:   .ASCIZ  /TYPE DESIRED 'TO' CHANNEL & CR: /

OKMSG:   .ASCIZ  /  OK/<15><12>

```

2290	012407	040	025052	051105	ERMSG: .ASCIZ / **ERROR**/<15><12>
2291	012414	047522	025122	006452	
2292	012422	000012			
2293	012424	051440	044513	050120	SKPMSG: .ASCIZ / SKIPPED STATE(S)/
2294	012432	042105	051440	040524	
2295	012440	042524	051450	000051	
2296	012446	047040	051101	047522	NARMSG: .ASCIZ * NARROW (< 1/2 LSB) STATE(S)*<15><12>
2297	012454	020127	036050	030440	
2298	012462	031057	046040	041123	
2299	012470	020051	052123	052101	
2300	012476	024105	024523	005015	
2301	012504	000			
2302	012505	040	044527	042504	WIDMSG: .ASCIZ * WIDE (> 1 1/2 LSB) STATE(S)*<15><12>
2303	012512	024040	020076	020061	
2304	012520	027461	020062	051514	
2305	012526	024502	051440	040524	
2306	012534	042524	051450	006451	
2307	012542	000012			
2308	012544	051440	040524	042524	OUTMSG: .ASCIZ / STATE(S) WIDER THAN 2 LSB/
2309	012552	051450	020051	044527	
2310	012560	042504	020122	044124	
2311	012566	047101	031040	046040	
2312	012574	041123	000		
2313	012577	040	052123	052101	HAFMSG: .ASCIZ * STATE-WIDTH(S) OUTSIDE + OR - 1/2 LSB*
2314	012604	026505	044527	052104	
2315	012612	024110	024523	047440	
2316	012620	052125	044523	042504	
2317	012626	025440	047440	020122	
2318	012634	020055	027461	020062	
2319	012642	051514	000102		
2320	012646	046040	041123	051040	MESR: .ASCIZ / LSB RMS, /
2321	012654	051515	020054	000	
2322	012661	040	051514	020102	MESP: .ASCIZ / LSB PEAK AT /
2323	012666	042520	045501	040440	
2324	012674	020124	000		
2325	012677	015	042412	042116	MEND: .ASCII <15><12>/END OF LOGIC TESTS/
2326	012704	047440	020106	047514	
2327	012712	044507	020103	042524	
2328	012720	052123	123		
2329	012723	040	047117	040440	ONAD: .ASCIZ / ON AD11K AT /
2330	012730	030504	045461	040440	
2331	012736	020124	000		
2332	012741	040	042101	030461	MSG50: .ASCIZ / AD11K'S FOUND/<15><12>
2333	012746	023513	020123	047506	
2334	012754	047125	006504	000012	
2335	012762	005012	025412	027461	MSG18: .ASCII <12><12><12>*+1/2 LSB*<15><12><12><12><12><12><12><12><12><12><12><12><1
2336	012770	020062	051514	006502	
2337	012776	005012	005012	005012	
2338	013004	005012	005012	005012	
2339	013012	030455	031057	051514	.ASCIZ \-1/2LSB\
2340	013020	000102			
2341					

2342				
2343	013022	044504	043106	051105
2344	013030	047105	044524	046101
2345	013036	046040	047111	040505
2346	013044	044522	054524	006472
2347	013052	000012		
2348	013054	020040	020040	020040
2349	013062	020040	020040	020040
2350	013070	020040	020040	020040
2351	013076	020040	052123	052101
2352	013104	026505	044527	052104
2353	013112	020110	044504	052123
2354	013120	044522	052502	044524
2355	013126	047117	005015	005012
2356	013134	020040	020043	043117
2357	013142	051440	040524	042524
2358	013150	005123	005012	005012
2359	013156	005012	005012	005012
2360	013164	005012	005012	005012
2361	013172	005012		
2362	013174	020040	020040	020040
2363	013202	020040	020040	020040
2364	013210	020040	020040	020040
2365	013216	020040	020040	020040
2366	013224	020040	020040	020040
2367	013232	020040	020040	020040
2368	013240	020040	020040	020040
2369	013246	020040	020040	020040
2370	013254	051440	040524	042524
2371	013262	053440	042111	044124
2372	013270	024040	051514	024502
2373	013276	005015		
2374	013300	030040	020040	020040
2375	013306	020040	020040	020040
2376	013314	020040	020040	027461
2377	013322	020062	020040	020040
2378	013330	020040	020040	020040
2379	013336	020040	020061	020040
2380	013344	020040	020040	020040
2381	013352	020040	030440	030440
2382	013360	031057	020040	020040
2383	013366	020040	020040	020040
2384	013374	020040	031040	000
2385	013401	015	052012	050131
2386	013406	020105	042514	052124
2387	013414	051105	023040	041440
2388	013422	020122	047506	020122
2389	013430	042504	044523	042522
2390	013436	020104	042524	052123
2391	013444	020072	000	
2392	013447	122	046105	052101
2393	013454	053111	020105	041501
2394	013462	052503	040522	054503
2395	013470	006472	000012	

MSG20: .EVEN .ASCIZ /DIFFERENTIAL LINEARITY:/(15)<12>

MSG16: .ASCII / STATE-WIDTH DISTRIBUTION/(15)<12><12><12>

.ASCII / # OF STATES/(12)<12><12><12><12><12><12><12><12><12><12><12><12><12><

.ASCII / STATE WIDTH (LSB)/(15)

.ASCIZ # 0 1/2 1 1 1/2 2#

MSG71: .ASCIZ (15)<12>/TYPE LETTER & CR FOR DESIRED TEST: /

MSG21: .ASCIZ /RELATIVE ACCURACY:/(15)<12>

H05

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665)
ASCII MESSAGES

2-DEC-76 16:29 PAGE 59

2396	013474	046040	041123	046440	LINEA: .ASCIZ / LSB MAXIMUM AT /
2397	013502	054101	046511	046525	
2398	013510	040440	020124	000	
2399	013515	015	041412	046101	HEADS: .ASCII <15><12>/CALIBRATION--/
2400	013522	041111	040522	044524	
2401	013530	047117	026455		
2402	013534	051440	052105	041440	ASKCH: .ASCIZ / SET CHANNEL IN SWR LOW BYTE/<15><12>
2403	013542	040510	047116	046105	
2404	013550	044440	020116	053523	
2405	013556	020122	047514	020127	
2406	013564	054502	042524	005015	
2407	013572	000			
2408	013573	033	000132		CO: .ASCIZ <33><132>
2409	013576	000055			C1: .ASCIZ <55>
2410	013600	031033	000		C2: .ASCIZ <33><62>
2411	013603	112	000		C3: .ASCIZ <112>
2412	013605	015	047412	043106	MOFSET: .ASCIZ <15><12>/OFFSET =/
2413	013612	042523	020124	000075	
2414	013620	046040	041123	000040	MLSB: .ASCIZ / LSB /
2415	013626	040440	020124	000	MAT: .ASCIZ / AT /
2416	013633	015	020012	047105	METST: .ASCIZ <15><12>/ ENTERING TEST /
2417	013640	042524	044522	043516	
2418	013646	052040	051505	020124	
2419	013654	000			
2420	013655	033	061	101	BUFF1: .BYTE 33,61,101,61,111,62,114,41,60,45,63,51,66,55,71,61,74,110,41,40,112,0
2421	013660	061	111	062	
2422	013663	114	041	060	
2423	013666	045	063	051	
2424	013671	066	055	071	
2425	013674	061	074	110	
2426	013677	041	040	112	
2427	013702	000			
2428	013703	033	061	101	BUFF2: .BYTE 33,61,101,47,111,61,104,50,65,44,62,110,40,40,102,0
2429	013706	047	111	061	
2430	013711	104	050	065	
2431	013714	044	062	110	
2432	013717	040	040	102	
2433	013722	000			
2434	013723	033	110	033	VTINIT: .BYTE 33,110,33,112,33,61,101,40,33,62,0
2435	013726	112	033	061	
2436	013731	101	040	033	
2437	013734	062	000		
2438	013736	005015	046412	026504	HEAD1: .ASCII <15><12><12>/MD-11-DZADL-B AD11K DIAGNOSTIC/<15><12>
2439	013744	030461	042055	040532	
2440	013752	046104	041055	020040	
2441	013760	020040	042101	030461	
2442	013766	020113	044504	043501	
2443	013774	047516	052123	041511	
2444	014002	005015			
2445	014004	040412	020072	052501	.ASCII <12>/A: AUTO TEST/
2446	014012	047524	052040	051505	
2447	014020	124			
2448	014021	015	041412	020072	.ASCII <15><12>/C: CALIBRATION/
2449	014026	040503	044514	051102	

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 60
ASCII MESSAGES

2450	014034	052101	047511	116
2451	014041	015	046012	020072
2452	014046	047514	044507	020103
2453	014054	042524	052123	
2454	014060	005015	035116	047040
2455	014066	044517	042523	052040
2456	014074	051505	124	

.ASCII <15><12>/L: LOGIC TEST/

.ASCII <15><12>/N: NOISE TEST/

2457	014077	015	051412	020072		.ASCII <15><12>/S: SETTLE TEST/
2458	014104	042523	052124	042514		
2459	014112	052040	051505	124		
2460	014117	015	053412	020072		.ASCIZ <15><12>/W: WRAPAROUND TEST/<15><12>
2461	014124	051127	050101	051101		
2462	014132	052517	042116	052040		
2463	014140	051505	006524	000012		
2464	014146	005015	052123	052101	EM1:	.ASCIZ <15><12>/STATUS REG. ERROR/<15><12>
2465	014154	051525	051040	043505		
2466	014162	020056	051105	047522		
2467	014170	006522	000012			
2468	014174	005015	040506	046111	EM2:	.ASCIZ <15><12>/FAILED TO INTERRUPT/<15><12>
2469	014202	042105	052040	020117		
2470	014210	047111	042524	051122		
2471	014216	050125	006524	000012		
2472	014224	005015	047125	054105	EM3:	.ASCIZ <15><12>/UNEXPECTED INTERRUPT/<15><12>
2473	014232	042520	052103	042105		
2474	014240	044440	052116	051105		
2475	014246	052522	052120	005015		
2476	014254	000				
2477	014255	015	042412	051122	EM4:	.ASCIZ <15><12>#ERROR ON A/D CHANNEL#<15><12>
2478	014262	051117	047440	020116		
2479	014270	027501	020104	044103		
2480	014276	047101	042516	006514		
2481	014304	000012				
2482	014306	051105	050122	020103	DH1:	.ASCIZ /ERRPC STREG EXPECTED ACTUAL/<15><12>
2483	014314	052123	042522	020107		
2484	014322	054105	042520	052103		
2485	014330	042105	040440	052103		
2486	014336	040525	006514	000012		
2487	014344	051105	050122	020103	DH2:	.ASCIZ /ERRPC STREG CHANNEL NOMINAL TOLERANCE ACTUAL/
2488	014352	051440	051124	043505		
2489	014360	020040	041440	040510		
2490	014366	047116	046105	020040		
2491	014374	047516	044515	040516		
2492	014402	020114	052040	046117		
2493	014410	051105	047101	042503		
2494	014416	020040	041501	052524		
2495	014424	046101	000			
2496	014427	105	051122	041520	DH3:	.ASCIZ /ERRPC STREG ACTUAL/<15><12>
2497	014434	020040	020040	020040		
2498	014442	052123	042522	020107		
2499	014450	020040	040440	052103		
2500	014456	040525	006514	000012		

K05

MAINDEC-11-DZADL-B
DZADLB.P11

MACY11 27(665) 2-DEC-76 16:29 PAGE 62
ASCII MESSAGES

2501	014464	000			HUNS:	.BYTE	0
2502	014465	056			DECPNT:	.BYTE	56
2503	014466	000			TENS:	.BYTE	0
2504	014467	000	000		ONES:	.BYTE	0,0
2505		014472			.EVEN		
2506							
2507	014472	001116	001316	001124	DT1:	\$ERRPC, STREG, \$GDDAT, \$BDDAT, 0	
2508	014500	001126	000000				
2509	014504	001116	001316	001366	DT2:	\$ERRPC, STREG, CHANL, \$GDDAT, SPREAD, \$BDDAT, 0	
2510	014512	001124	001406	001126			
2511	014520	000000					
2512	014522	001116	001316	001126	DT3:	\$ERRPC, STREG, \$BDDAT, 0	
2513	014530	000000					
2514							
2515	014532	000000			DF1:	0	
2516							
2517							
2518							
2519							

```

2520 .SBTTL TTY INPUT ROUTINE
2521
2522 ::*****
2523 .ENABL LSB
2524
2525 .DSABL LSB
2526
2527
2528 ::*****
2529 ::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2530 ::CALL:
2531 ::
2532 * RDCHR :: INPUT A SINGLE CHARACTER FROM THE TTY
2533 * RETURN HERE :: CHARACTER IS ON THE STACK
2534 * :: WITH PARITY BIT STRIPPED OFF
2535
2536 014534 011646 $RDCHR: MOV (SP),-(SP) :: PUSH DOWN THE PC
2537 014536 016666 000004 000002 MOV 4(SP),2(SP) :: SAVE THE PS
2538 014544 105777 164374 1$ TSTB @STKS :: WAIT FOR
2539 014550 100375 BPL 1$ :: A CHARACTER
2540 014552 117766 164370 000004 MOVB @STKB,4(SP) :: READ THE TTY
2541 014560 042766 177600 000004 BIC #C<177>,4(SP) :: GET RID OF JUNK IF ANY
2542 014566 026627 000004 000023 CMP 4(SP),#23 :: IS IT A CONTROL-S?
2543 014574 001013 BNE 3$ :: BRANCH IF NO
2544 014576 105777 164342 2$ TSTB @STKS :: WAIT FOR A CHARACTER
2545 014602 100375 BPL 2$ :: LOOP UNTIL ITS THERE
2546 014604 117746 164336 MOVB @STKB,-(SP) :: GET CHARACTER
2547 014610 042716 177600 BIC #C177,(SP) :: MAKE IT 7-BIT ASCII
2548 014614 022627 000021 CMP (SP)+,#21 :: IS IT A CONTROL-Q?
2549 014620 001366 BNE 2$ :: IF NOT DISCARD IT
2550 014622 000750 BR 1$ :: YES, RESUME
2551 014624 026627 000004 000140 3$ CMP 4(SP),#140 :: IS IT UPPER CASE?
2552 014632 002407 BLT 4$ :: BRANCH IF YES
2553 014634 026627 000004 000175 CMP 4(SP),#175 :: IS IT A SPECIAL CHAR?
2554 014642 003003 BGT 4$ :: BRANCH IF YES
2555 014644 042766 000040 000004 BIC #40,4(SP) :: MAKE IT UPPER CASE
2556 014652 000002 4$ RTI :: GO BACK TO USER
2557
2558 ::*****
2559 ::THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2560 ::CALL:
2561 * RDLIN :: INPUT A STRING FROM THE TTY
2562 * RETURN HERE :: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2563 * :: TERMINATOR WILL BE A BYTE OF ALL 0'S
2564 014654 010346 $RDLIN: MOV R3,-(SP) :: SAVE R3
2565 014656 012703 014762 1$ MOV #TTYIN,R3 :: GET ADDRESS
2566 014662 022703 014772 2$ CMP #TTYIN+8.,R3 :: BUFFER FULL?
2567 014666 101405 BLOS 4$ :: BR IF YES
2568 014670 104404 RDCHR :: GO READ ONE CHARACTER FROM THE TTY
2569 014672 112613 MOVB (SP)+,(R3) :: GET CHARACTER
2570 014674 122713 000177 10$ CMPB #177,(R3) :: IS IT A RUBOUT
2571 014700 001003 BNE 3$ :: SKIP IF NOT
2572 014702 104400 001170 4$ TYPE $QUES :: TYPE A '?'
2573 014706 000763 BR 1$ :: CLEAR THE BUFFER AND LOOP
    
```


2574	014710	111337	014760	3\$:	MOVB	(R3),9\$::ECHO THE CHARACTER
2575	014714	104400	014760		TYPE	,9\$	
2576	014720	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
2577	014724	001356			BNE	2\$::LOOP IF NOT RETURN
2578	014726	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
2579	014732	104400	001172		TYPE	,\$LF	::TYPE A LINE FEED
2580	014736	012603			MOV	(SP)+,R3	::RESTORE R3
2581	014740	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
2582	014742	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
2583	014750	012766	014762	000004	MOV	#\$TTYIN,4(SP)	
2584	014756	000002			RTI		::RETURN
2585	014760	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
2586	014761	000			.BYTE	0	::TERMINATOR
2587	014762	000010		\$TTYIN:	.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
2588	014772	052536	005015	000	\$CNTLU:	.ASCIZ /↑U/<15><12>	::CONTROL "U"
2589	014777	136	006507	000012	\$CNTLG:	.ASCIZ /↑G/<15><12>	::CONTROL "G"
2590	015004	005015	053523	020122	\$MSWR:	.ASCIZ <15><12>/SWR = /	
2591	015012	020075	000				
2592	015015	040	047040	053505	\$MNEW:	.ASCIZ / NEW = /	
2593	015022	036440	000040				

```

2594 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2595
2596 ;:*****
2597 ;:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2598 ;:CHANGE IT TO BINARY.
2599 ;:CALL:
2600 ;: * RDOCT ;:READ AN OCTAL NUMBER
2601 ;: * RETURN HERE ;:LOW ORDER BITS ARE ON TOP OF THE STACK
2602 ;: * ;:HIGH ORDER BITS ARE IN $HIOCT
2603
2604 015026 011646 000004 000002 $RDOCT: MOV (SP),-(SP) ;:PROVIDE SPACE FOR THE
2605 015030 016666 000004 000002 MOV 4(SP),2(SP) ;:INPUT NUMBER
2606 015036 010046 MOV RO,-(SP) ;:PUSH RO ON STACK
2607 015040 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
2608 015042 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
2609 015044 104405 1$: RDLIN ;:READ AN ASCII LINE
2610 015046 012600 MOV (SP)+,RO ;:GET ADDRESS OF 1ST CHARACTER
2611 015050 005001 CLR R1 ;:CLEAR DATA WORD
2612 015052 005002 CLR R2
2613 015054 112046 2$: MOV (RO)+,-(SP) ;:PICKUP THIS CHARACTER
2614 015056 001412 BEQ 3$ ;:IF ZERO GET OUT
2615 015060 006301 ASL R1 ;:*2
2616 015062 006102 ROL R2
2617 015064 006301 ASL R1 ;:*4
2618 015066 006102 ROL R2
2619 015070 006301 ASL R1 ;:*8
2620 015072 006102 ROL R2
2621 015074 042716 177770 BIC #1C7,(SP) ;:STRIP THE ASCII JUNK
2622 015100 062601 ADD (SP)+,R1 ;:ADD IN THIS DIGIT
2623 015102 000764 BR 2$ ;:LOOP
2624 015104 005726 3$: TST (SP)+ ;:CLEAN TERMINATOR FROM STACK
2625 015106 010166 000012 MOV R1,12(SP) ;:SAVE THE RESULT
2626 015112 010237 015126 MOV R2,$HIOCT
2627 015116 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
2628 015120 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
2629 015122 012600 MOV (SP)+,RO ;:POP STACK INTO RO
2630 015124 000002 RTI ;:RETURN
2631 015126 000000 $HIOCT: .WORD 0 ;:HIGH ORDER BITS GO HERE
    
```

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($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
    
```

2633										
2634										
2635										
2636										
2637										
2638										
2639										
2640										
2641										
2642										
2643										
2644										
2645										
2646	015130				\$SCOPE:					
2647	015130	032777	040000	164002	1\$: BIT	#BIT14,\$SWR		::LOOP ON PRESENT TEST?		
2648	015136	001114			BNE	\$OVER		::YES IF SW14=1		
2649					;	*****START OF CODE FOR THE XOR TESTER*****				
2650	015140	000416			\$XTSTR: BR	6\$::IF RUNNING ON THE "XOR" TESTER CHANGE		
2651								::THIS INSTRUCTION TO A "NOP" (NOP=240)		
2652	015142	013746	000004		MOV	\$ERRVEC,-(SP)		::SAVE THE CONTENTS OF THE ERROR VECTOR		
2653	015146	012737	015166	000004	MOV	\$S,\$ERRVEC		::SET FOR TIMEOUT		
2654	015154	005737	177060		TST	\$177060		::TIME OUT ON XOR?		
2655	015160	012637	000004		MOV	(SP)+,\$ERRVEC		::RESTORE THE ERROR VECTOR		
2656	015164	000463			BR	\$SVLAD		::GO TO THE NEXT TEST		
2657	015166	022626			5\$: CMP	(SP)+,(SP)+		::CLEAR THE STACK AFTER A TIME OUT		
2658	015170	012637	000004		MOV	(SP)+,\$ERRVEC		::RESTORE THE ERROR VECTOR		
2659	015174	000423			BR	7\$::LOOP ON THE PRESENT TEST		
2660	015176				6\$:;	*****END OF CODE FOR THE XOR TESTER*****				
2661	015176	032777	000400	163734	BIT	#BIT08,\$SWR		::LOOP ON SPEC. TEST?		
2662	015204	001404			BEQ	2\$::BR IF NO		
2663	015206	127737	163726	001102	CMPB	\$SWR,\$TSTNM		::ON THE RIGHT TEST? SWR<7:0>		
2664	015214	001465			BEQ	\$OVER		::BR IF YES		
2665	015216	105737	001103		2\$: TSTB	\$ERFLG		::HAS AN ERROR OCCURRED?		
2666	015222	001421			BEQ	3\$::BR IF NO		
2667	015224	123737	001115	001103	CMPB	\$ERMAX,\$ERFLG		::MAX. ERRORS FOR THIS TEST OCCURRED?		
2668	015232	101015			BHI	3\$::BR IF NO		
2669	015234	032777	001000	163676	BIT	#BIT09,\$SWR		::LOOP ON ERROR?		
2670	015242	001404			BEQ	4\$::BR IF NO		
2671	015244	013737	001110	001106	7\$: MOV	\$LPERR,\$LPADR		::SET LOOP ADDRESS TO LAST SCOPE		
2672	015252	000446			BR	\$OVER				
2673	015254	105037	001103		4\$: CLRB	\$ERFLG		::ZERO THE ERROR FLAG		
2674	015260	005037	001160		CLR	\$TIMES		::CLEAR THE NUMBER OF ITERATIONS TO MAKE		
2675	015264	000415			BR	1\$::ESCAPE TO THE NEXT TEST		
2676	015266	032777	004000	163644	3\$: BIT	#BIT11,\$SWR		::INHIBIT ITERATIONS?		
2677	015274	001011			BNE	1\$::BR IF YES		
2678	015276	005737	001202		TST	\$PASS		::IF FIRST PASS OF PROGRAM		
2679	015302	001406			BEQ	1\$::INHIBIT ITERATIONS		
2680	015304	005237	001104		INC	\$ICNT		::INCREMENT ITERATION COUNT		
2681	015310	023737	001160	001104	CMP	\$TIMES,\$ICNT		::CHECK THE NUMBER OF ITERATIONS MADE		
2682	015316	002024			BGE	\$OVER		::BR IF MORE ITERATION REQUIRED		
2683	015320	012737	000001	001104	1\$: MOV	\$1,\$ICNT		::REINITIALIZE THE ITERATION COUNTER		
2684	015326	013737	015404	001160	MOV	\$MXCNT,\$TIMES		::SET NUMBER OF ITERATIONS TO DO		
2685	015334	105237	001102		\$SVLAD: INCB	\$TSTNM		::COUNT TEST NUMBERS		

```

2696 015340 113737 001102 001200
2697 015346 011637 001106
2698 015352 011637 001110
2699 015356 005037 001162
2700 015362 112737 000001 001115
2701 015370 013777 001102 163544
2702 015376 013716 001106
2703 015402 000002
2704 015404 003720
2705
2706
2707
2708
2709
2710 015406 105237 001103
2711 015406 001775
2712 015412 013777 001102 163520
2713 015422 032777 002000 163510
2714 015430 001402
2715 015432 104400 001164
2716 015436 005237 001112
2717 015442 011637 001116
2718 015446 162737 000002 001116
2719 015454 117737 163436 001114
2720 015462 032777 020000 163450
2721 015470 001004
2722 015472 004737 015602
2723 015476 104400 001171
2724 015502
2725 015502 122737 000001 001214
2726 015510 001007
2727 015512 113737 001114 015524
2728 015520 004737 016236
2729 015524 000
2730 015526 000
2731 015526 000777
2732 015530 005777 163404
2733 015534 100001
2734 015536 000000
2735 015540 032777 001000 163372
2736 015546 001402
2737 015550 013716 001110
2738 015554 005737 001162
2739 015560 001402

```

```

MOV B $STNM,$STN :: SET TEST NUMBER IN APT MAILBOX
MOV (SP), $LPADR :: SAVE SCOPE LOOP ADDRESS
MOV (SP), $LPERR :: SAVE ERROR LOOP ADDRESS
CLR $ESCAPE :: CLEAR THE ESCAPE FROM ERROR ADDRESS
MOV B #1,$ERMAX :: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
$OVER: MOV $STNM,$DISPLAY :: DISPLAY TEST NUMBER
MOV $LPADR,(SP) :: FUDGE RETURN ADDRESS
RTI :: FIXES PS
$MXCNT: 2000 :: MAX. NUMBER OF ITERATIONS
.SBTTL ERROR HANDLER ROUTINE

```

```

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

```

```

$ERROR:
7$: INCB $ERFLG :: SET THE ERROR FLAG
BEQ 7$ :: DON'T LET THE FLAG GO TO ZERO
MOV $STNM,$DISPLAY :: DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR :: BELL ON ERROR?
BEQ 1$ :: NO - SKIP
TYPE $BELL :: RING BELL
1$: INC $ERTTL :: COUNT THE NUMBER OF ERRORS
MOV (SP), $ERRPC :: GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOV B,$ERRPC,$ITEMB :: STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR :: SKIP TYPEOUT IF SET
BNE 20$ :: SKIP TYPEOUTS
JSR PC,$ERRTYP :: GO TO USER ERROR ROUTINE
TYPE $CRLF
20$: CMPB #APTENV,$ENV :: RUNNING IN APT MODE
BNE 2$ :: NO SKIP APT ERROR REPORT
MOV B,$ITEMB,21$ :: SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$ATY4 :: REPORT FATAL ERROR TO APT
21$: .BYTE 0
22$: .BYTE 0
23$: BR 22$ :: APT ERROR LOOP
2$: TST $SWR :: HALT ON ERROR
BPL 3$ :: SKIP IF CONTINUE
HALT :: HALT ON ERROR!
3$: BIT #BIT09,$SWR :: LOOP ON ERROR SWITCH SET?
BEQ 4$ :: BR IF NO
MOV $LPERR,(SP) :: FUDGE RETURN FOR LOOPING
TST $ESCAPE :: CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ :: BR IF NONE
5$:

```

```

2740 015562 013716 001162          MOV      $ESCAPE, (SP)      ;; FUDGE RETURN ADDRESS FOR ESCAPE
2741 015566                                     5$:
2742 015566 022737 011776 000042      CMP      #SENDAD, @#42     ;; ACT-11 AUTO-ACCEPT?
2743 015574 001001                                     BNE     6$                ;; BRANCH IF NO
2744 015576 000000                                     HALT                                     ;; YES
2745 015600                                     6$:
2746 015600 000002          RTI                          ;; RETURN
2747                                     .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761 015602
2762 015602 104400 001171          TYPE     $CRLF              ;; "CARRIAGE RETURN" & "LINE FEED"
2763 015606 010046          MOV      RO, -(SP)         ;; SAVE RO
2764 015610 005000          CLR     RO                 ;; PICKUP THE ITEM INDEX
2765 015612 153700 001114          BISB   @#$ITEMB, RO
2766 015616 001004          BNE     1$                ;; IF ITEM NUMBER IS ZERO, JUST
2767                                     ;; TYPE THE PC OF THE ERROR
2768
2769 015620 013746 001116          MOV      $ERRPC, -(SP)    ;; SAVE $ERRPC FOR TYPEOUT
2770                                     ;; ERROR ADDRESS
2771 015624 104401          TYP0C                                     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2772 015626 000426          BR      6$                ;; GET OUT
2773 015630 005300          1$:  DEC     RO              ;; ADJUST THE INDEX SO THAT IT WILL
2774 015632 006300          ASL     RO                 ;; WORK FOR THE ERROR TABLE
2775 015634 006300          ASL     RO
2776 015636 006300          ASL     RO
2777 015640 062700 001256          ADD     #SERRTB, RO      ;; FORM TABLE POINTER
2778 015644 012037 015654          MOV     (RO)+, 2$        ;; PICKUP "ERROR MESSAGE" POINTER
2779 015650 001404          BEQ    3$                ;; SKIP TYPEOUT IF NO POINTER
2780 015652 104400          TYPE   $CRLF              ;; TYPE THE "ERROR MESSAGE"
2781 015654 000000          2$:  .WORD  0              ;; "ERROR MESSAGE" POINTER GOES HERE
2782 015656 104400 001171          TYPE   $CRLF              ;; "CARRIAGE RETURN" & "LINE FEED"
2783 015662 012037 015672          3$:  MOV     (RO)+, 4$        ;; PICKUP "DATA HEADER" POINTER
2784 015666 001404          BEQ    5$                ;; SKIP TYPEOUT IF 0
2785 015670 104400          TYPE   $CRLF              ;; TYPE THE "DATA HEADER"
2786 015672 000000          4$:  .WORD  0              ;; "DATA HEADER" POINTER GOES HERE
2787 015674 104400 001171          TYPE   $CRLF              ;; "CARRIAGE RETURN" & "LINE FEED"
2788 015700 011000          5$:  MOV     (RO), RO         ;; PICKUP "DATA TABLE" POINTER
2789 015702 001004          BNE    7$                ;; GO TYPE THE DATA
2790 015704 012600          6$:  MOV     (SP)+, RO      ;; RESTORE RO
2791 015706 104400 001171          TYPE   $CRLF              ;; "CARRIAGE RETURN" & "LINE FEED"
2792 015712 000207          RTS     PC                 ;; RETURN
2793 015714                                     7$:
2794 015714 013046          MOV     @ (RO)+, -(SP)    ;; SAVE @ (RO)+ FOR TYPEOUT
2795 015716 104401          TYP0C                                     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2796 015720 005710          TST    (RO)              ;; IS THERE ANOTHER NUMBER?
2797 015722 001770          BEQ    6$                ;; BR IF NO
2798 015724 104400 015732          TYPE   $CRLF              ;; TYPE TWO(2) SPACES
2799 015730 000771          BR     7$                ;; LOOP
2800 015732 020040          8$:  .ASCIZ  / /              ;; TWO(2) SPACES
2801 015736 .EVEN

```

.SBTTL TYPE ROUTINE

2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847

015736 105737 001157
015742 100002
015744 000000
015746 000430
015750 010046
015752 017600 000002
015756 122737 000001 001214
015764 001011
015766 132737 000100 001215
015774 001405
015776 010037 016006
016002 004737 016226
016006 000000
016010 132737 000040 001215
016016 001003
016020 112046
016022 001005
016024 005726
016026 012600
016030 062716 000002
016034 000002
016036 122716 000011
016042 001430
016044 122716 000200
016050 001006
016052 005726
016054 104400
016056 001171
016060 105037 016214
016064 000755
016066 004737 016150
016072 123726 001156
016076 001350
016100 013746 001154
016104 105366 000001
016110 002770

```
*****
*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 $STPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV RO, -(SP) ;; SAVE RO
MOV @2(SP), RO ;; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
BITB #APTSPool, $ENVM ;; SPOOL MESSAGE TO APT
BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
61$: .WORD 0 ;; MESSAGE ADDRESS
62$: BITB #APTCSUP, $ENVM ;; APT CONSOLE SUPPRESSED
BNE 60$ ;; YES, SKIP TYPE OUT
2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+, RO ;; RESTORE RO
3$: ADD #2, (SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
BEQ 8$
CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
BR 2$ ;; GET NEXT CHARACTER
5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER
6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;; IF NO GO GET NEXT CHAR.
MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
```

```

2848 016112 004737 016150 JSR PC,$STYPEC ;;GO TYPE A NULL
2849 016116 105337 016214 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
2850 016122 000770 BR 7$ ;;LOOP
2851
2852 ;HORIZONTAL TAB PROCESSOR
2853
2854 016124 112716 000040 8$: MOVB #' (SP) ;;REPLACE TAB WITH SPACE
2855 016130 004737 016150 9$: JSR PC,$STYPEC ;;TYPE A SPACE
2856 016134 132737 000007 016214 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
2857 016142 001372 BNE 9$ ;;TAB STOP
2858 016144 005726 TST (SP)+ ;;POP SPACE OFF STACK
2859 016146 000724 BR 2$ ;;GET NEXT CHARACTER
2860 016150 105777 162774 $STYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
2861 016154 100375 BPL $STYPEC
2862 016156 116677 000002 162766 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2863 016164 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
2864 016172 001003 BNE 1$ ;;BRANCH IF NO
2865 016174 105037 016214 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
2866 016200 000406 BR $STYPEC ;;EXIT
2867 016202 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
2868 016210 001402 BEQ $STYPEC ;;BRANCH IF YES
2869 016212 105227 INCB (PC)+ ;;COUNT THE CHARACTER
2870 016214 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
2871 016216 000207 $STYPEC: RTS PC
2872
2873 .SBTTL APT COMMUNICATIONS ROUTINE
2874
2875 ;*****
2876 016220 112737 000001 016464 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
2877 016226 112737 000001 016462 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
2878 016234 000403 BR $ATYC
2879 016236 112737 000001 016464 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
2880 016244 $ATYC:
2881 016244 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
2882 016246 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
2883 016250 105737 016462 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
2884 016254 001450 BEQ 5$ ;;IF NOT: BR
2885 016256 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
2886 016264 001031 BNE 3$ ;;IF NOT: BR
2887 016266 132737 000100 001215 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
2888 016274 001425 BEQ 3$ ;;IF NOT: BR
2889 016276 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
2890 016302 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
2891 016310 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
2892 016314 001375 BNE 1$ ;;IF NOT: WAIT
2893 016316 010037 001210 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
2894 016322 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
2895 016324 001376 BNE 2$
2896 016326 163700 001210 SUB $MSGAD,RO ;;SUB START OF MESSAGE
2897 016332 006200 ASR RO ;;GET MESSAGE LNGTH IN WORDS
2898 016334 010037 001212 MOV RO,$MSGGLT ;;PUT LENGTH IN MAILBOX
2899 016340 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
2900 016346 000413 BR 5$
2901 016350 017637 000004 016374 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE

```

G06

2902	016356	062766	000002	000004		ADD	#2,4(SP)	::BUMP RETURN ADDRESS
2903	016364	013746	177776			MOV	177776,-(SP)	::PUSH 177776 ON STACK
2904	016370	004737	015736			JSR	PC,\$TYPE	::CALL TYPE MACRO
2905	016374	000000			4\$:	.WORD	0	
2906	016376				5\$:			
2907	016376	105737	016464		10\$:	TSTB	\$FFLG	::SHOULD REPORT FATAL ERROR?
2908	016402	001416				BEQ	12\$::IF NOT: BR
2909	016404	005737	001214			TST	\$ENV	::RUNNING UNDER APT?
2910	016410	001413				BEQ	12\$::IF NOT: BR
2911	016412	005737	001174		11\$:	TST	\$MSGTYPE	::FINISHED LAST MESSAGE?
2912	016416	001375				BNE	11\$::IF NOT: WAIT
2913	016420	017637	000004	001176		MOV	24(SP),\$FATAL	::GET ERROR #
2914	016426	062766	000002	000004		ADD	#2,4(SP)	::BUMP RETURN ADDR.
2915	016434	005237	001174			INC	\$MSGTYPE	::TELL APT TO TAKE ERROR
2916	016440	105037	016464		12\$:	CLRB	\$FFLG	::CLEAR FATAL FLAG
2917	016444	105037	016463			CLRB	\$LFLG	::CLEAR LOG FLAG
2918	016450	105037	016462			CLRB	\$MFLG	::CLEAR MESSAGE FLAG
2919	016454	012601				MOV	(SP)+,R1	::POP STACK INTO R1
2920	016456	012600				MOV	(SP)+,R0	::POP STACK INTO R0
2921	016460	000207				RTS	PC	::RETURN
2922	016462	000			\$MFLG:	.BYTE	0	::MESSG. FLAG
2923	016463	000			\$LFLG:	.BYTE	0	::LOG FLAG
2924	016464	000			\$FFLG:	.BYTE	0	::FATAL FLAG
2925		016466				.EVEN		
2926		000200			APTSIZE=	200		
2927		000001			APTENV=	001		
2928		000100			APTSPool=	100		
2929		000040			APTCSUP=	040		

H06

2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983

```
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   ;;CALL FOR TYPEOUT

016466 017646 000000          $TYPOS: MOV     2(SP),-(SP)      ;; PICKUP THE MODE
016472 116637 000001 016711  MOVB   1(SP),SOFILL    ;; LOAD ZERO FILL SWITCH
016500 112637 016713          MOVB   (SP)+,SOMODE+1  ;; NUMBER OF DIGITS TO TYPE
016504 062716 000002          ADD    #2,(SP)        ;; ADJUST RETURN ADDRESS
016510 000406          BR     $TYPON
016512 112737 000001 016711  $TYPOC: MOVB   #1,SOFILL    ;; SET THE ZERO FILL SWITCH
016520 112737 000006 016713  MOVB   #6,SOMODE+1    ;; SET FOR SIX(6) DIGITS
016526 112737 000005 016710  $TYPON: MOVB   #5,SOCNT    ;; SET THE ITERATION COUNT
016534 010346          MOV    R3,-(SP)      ;; SAVE R3
016536 010446          MOV    R4,-(SP)      ;; SAVE R4
016540 010546          MOV    R5,-(SP)      ;; SAVE R5
016542 113704 016713          MOVB   SOMODE+1,R4   ;; GET THE NUMBER OF DIGITS TO TYPE
016546 005404          NEG    R4
016550 062704 000006          ADD    #6,R4         ;; SUBTRACT IT FOR MAX. ALLOWED
016554 110437 016712          MOVB   R4,SOMODE     ;; SAVE IT FOR USE
016560 113704 016711          MOVB   SOFILL,R4     ;; GET THE ZERO FILL SWITCH
016564 016605 000012          MOV    12(SP),R5    ;; PICKUP THE INPUT NUMBER
016570 005003          CLR    R3           ;; CLEAR THE OUTPUT WORD
016572 006105          1$:   ROL    R5      ;; ROTATE MSB INTO "C"
016574 000404          BR     3$          ;; GO DO MSB
016576 006105          2$:   ROL    R5      ;; FORM THIS DIGIT
016600 006105          ROL    R5
016602 006105          ROL    R5
016604 010503          MOV    R5,R3
016606 006103          3$:   ROL    R3      ;; GET LSB OF THIS DIGIT
016610 105337 016712          DECB   SOMODE        ;; TYPE THIS DIGIT?
016614 100016          BPL   7$           ;; BR IF NO
016616 042703 177770          BIC   #177770,R3    ;; GET RID OF JUNK
016622 001002          BNE   4$           ;; TEST FOR 0
016711          4$:
```

2984	016624	005704		TST	R4	:: SUPPRESS THIS 0?
2985	016626	001403		BEQ	5\$:: BR IF YES
2986	016630	005204	4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
2987	016632	052703	000060	BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
2988	016636	052703	000040	5\$:	#',R3	:: MAKE ASCII IF NOT ALREADY
2989	016642	110337	016706	MOV	R3,8\$:: SAVE FOR TYPING
2990	016646	104400	016706	TYPE	8\$:: GO TYPE THIS DIGIT
2991	016652	105337	016710	7\$:	\$OCNT	:: COUNT BY 1
2992	016656	003347		BGT	2\$:: BR IF MORE TO DO
2993	016660	002402		BLT	6\$:: BR IF DONE
2994	016662	005204		INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
2995	016664	000744		BR	2\$:: GO DO THE LAST DIGIT
2996	016666	012605	6\$:	MOV	(SP)+,R5	:: RESTORE R5
2997	016670	012604		MOV	(SP)+,R4	:: RESTORE R4
2998	016672	012603		MOV	(SP)+,R3	:: RESTORE R3
2999	016674	016666	000002 000004	MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
3000	016702	012616		MOV	(SP)+,(SP)	
3001	016704	000002		RTI		:: RETURN
3002	016706	000	8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
3003	016707	000		.BYTE	00	:: TERMINATOR FOR TYPE ROUTINE
3004	016710	000	\$OCNT:	.BYTE	00	:: OCTAL DIGIT COUNTER
3005	016711	000	\$OFILL:	.BYTE	00	:: ZERO FILL SWITCH
3006	016712	000000	\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

3007
 3008
 3009
 3010
 3011
 3012
 3013
 3014
 3015
 3016
 3017
 3018
 3019
 3020
 3021
 3022
 3023
 3024
 3025
 3026
 3027
 3028
 3029
 3030
 3031
 3032
 3033
 3034
 3035
 3036
 3037
 3038
 3039

.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.
    
```

016714 010046
 016716 016600 000002
 016722 005740
 016724 111000
 016726 006300
 016730 016000 016736
 016734 000200

```

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

.SBTTL TRAP TABLE

```

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

: ROUTINE
 :-----

016736
 016736 015736
 016740 016512
 016742 016466
 016744 016526
 016746 014534
 016750 014654
 016752 015026

```

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

        $RDCHR ;;CALL=RDCHR    TRAP+4(104404)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+5(104405)  TTY TYPEIN STRING ROUTINE
        $RDOCT ;;CALL=RDOCT    TRAP+6(104406)  READ AN OCTAL NUMBER FROM TTY
    
```

```

3040          .SBTTL POWER DOWN AND UP ROUTINES
3041
3042          ;:*****
3043          :POWER DOWN ROUTINE
3044 016754 012737 017120 000024 $PWRDN: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST UP
3045 016762 012737 000340 000026      MOV #340,@#PWRVEC+2 ;;PRIO:7
3046 016770 010046      MOV RO,-(SP) ;;PUSH RO ON STACK
3047 016772 010146      MOV R1,-(SP) ;;PUSH R1 ON STACK
3048 016774 010246      MOV R2,-(SP) ;;PUSH R2 ON STACK
3049 016776 010346      MOV R3,-(SP) ;;PUSH R3 ON STACK
3050 017000 010446      MOV R4,-(SP) ;;PUSH R4 ON STACK
3051 017002 010546      MOV R5,-(SP) ;;PUSH R5 ON STACK
3052 017004 017746 162130      MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
3053 017010 010637 017124      MOV SP,$SAVR6 ;;SAVE SP
3054 017014 012737 017026 000024      MOV $PWRUP,@#PWRVEC ;;SET UP VECTOR
3055 017022 000000      HALT
3056 017024 000776      BR -.2 ;;HANG UP
3057
3058          ;:*****
3059          :POWER UP ROUTINE
3060 017026 012737 017120 000024 $PWRUP: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
3061 017034 013706 017124      MOV $SAVR6,SP ;;GET SP
3062 017040 005037 017124      CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
3063 017044 005237 017124      IS: INC $SAVR6 ;;WAIT FOR THE INC
3064 017050 001375      BNE IS ;;OF WORD
3065 017052 012677 162062      MOV (SP)+,@SWR ;;POP STACK INTO @SWR
3066 017056 012605      MOV (SP)+,R5 ;;POP STACK INTO R5
3067 017060 012604      MOV (SP)+,R4 ;;POP STACK INTO R4
3068 017062 012603      MOV (SP)+,R3 ;;POP STACK INTO R3
3069 017064 012602      MOV (SP)+,R2 ;;POP STACK INTO R2
3070 017066 012601      MOV (SP)+,R1 ;;POP STACK INTO R1
3071 017070 012600      MOV (SP)+,R0 ;;POP STACK INTO R0
3072 017072 012737 016754 000024      MOV $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3073 017100 012737 000340 000026      MOV #340,@#PWRVEC+2 ;;PRIO:7
3074 017106 104400      TYPE ;;REPORT THE POWER FAILURE
3075 017110 017126      SPWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
3076 017112 012716      MOV (PC)+,(SP) ;;RESTART AT BEG2
3077 017114 002256      SPWRAD: .WORD BEG2 ;;RESTART ADDRESS
3078 017116 000002      RTI
3079 017120 000000      $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
3080 017122 000776      BR -.2 ;;BEFORE THE POWER DOWN WAS COMPLETE
3081 017124 000000      $SAVR6: 0 ;;PUT THE SP HERE
3082 017126 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
3083 017134 000122
3084          .EVEN
3085          .EVEN
3086 017136 000310      DIST: .BLKW 200. ;;STATE-WIDTH DISTRIBUTION
3087 017756 010000      BUFFER: .BLKW 4096. ;;BUFFER AREA
3088
3089          .END

```


PR3	= 000140	435#												
PR4	= 000200	436#												
PR5	= 000240	437#												
PR6	= 000300	438#												
PR7	= 000340	439#												
PS	= 177776	412#	413											
PSW	= 177776	413#	873*	1034*	1057*	1058*	1068*	1074*	1388*	2092*				
PWAVEC	= 000024	504#	807*	808*	3044*	3045*	3054*	3060*	3072*	3073*				
QUEST	012137	786	894	2251#										
RANDY	010722	1706	2018#											
RBEG	001634	790	792#											
ROCHR	= 104404	2568	3037#											
ROLIN	= 104405	871	2609	3038#										
RODOCT	= 104406	782	1514	1517	3039#									
READ	007210	1737#	1775											
RELACC	007650	1813	1830#											
REST1	002206	838	856#											
RESVEC	= 000010	499#												
RET	006512	1630	1638#											
RETERR	003710	1143	1145#											
RETURN	001610	785#	1399	1610	1659	1704	2038							
RMS	001400	735#	1951	1953*	1957	1970	1979*	1983*	1986*	1999*	2007*	2008*		
RNA	001372	732#	861*	1707	2018*	2019*	2020*	2021	2024					
RNB	001374	733#	862*	2018	2021*	2022*	2023*	2025						
RNC	001376	734#	863*	2019	2022	2024*	2025*	2026*						
RST	011266	763	769	2090#										
RO	=%000000	420#	756	757*	758*	759	765	771	784*	843	846	849	851	858*
		859*	860*	874*	875*	876	879	882	885	888	891	916*	919*	923*
		924*	990*	993*	998*	1021*	1032*	1055*	1075*	1110*	1111*	1116*	1122*	1125*
		1133*	1174*	1178*	1182*	1330*	1331*	1332	1380*	1381*	1389*	1390*	1394	1400*
		1401*	1402	1406*	1409	1412	1485*	1489	1491*	1492	1493	1496	1497	1601*
		1602*	1603*	1604	1605*	1606*	1609*	1614*	1651*	1670*	1673	1682*	1684*	1687*
		1695*	1698*	1699*	1700*	1701*	1702	1703*	1733*	1735*	1774*	1803*	1807	1817*
		1819	1833*	1834	1837*	1844	1847	1875*	1877	1883*	1886*	1891	1894	1910*
		1913*	1919	1920	1925*	1927*	2029*	2030*	2031	2032*	2034	2035*	2036*	2039*
		2043*	2056*	2057*	2059*	2060	2070	2223*	2226	2606	2610*	2613	2629*	2756
		2757*	2758*	2765*	2766*	2767*	2768*	2769*	2770	2775	2780*	2782*	2786	2788
		2815	2816*	2821	2826	2829*	2881	2889*	2893	2894	2896*	2897*	2898	2920*
		3015	3016*	3017	3018*	3019*	3020*	3021*	3046	3071*				
R1	=%000001	421#	1417*	1418*	1486*	1487	1489*	1490*	1491	1496	1497	1657*	1671*	1683*
		1685*	1688*	1696*	1705*	1725*	1728*	1736*	1737	1763	1818*	1823*	1830*	1836*
		1837	1838	1876*	1884*	1889*	1906*	1918*	1922*	1926*	1929*	2055*	2057	2152*
		2162*	2163	2607	2611*	2615*	2617*	2619*	2622*	2625	2628*	2882	2919*	3047
		3070*												
R2	=%000002	422#	1260*	1261	1270	1271	1281*	1282	1334*	1403*	1420*	1522*	1526*	1527
		1570*	1574*	1582*	1589	1591*	1592	1707*	1708*	1710*	1714*	1716	1718*	1719*
		1721*	1723	1737*	1738*	1739*	1740*	1741*	1742	1746*	1747*	1748*	1749	1752
		1756	1777*	1785*	1788*	1789*	1792*	1800*	1801*	1802*	1803	1819*	1821*	1834*
		1835*	1836	1853*	1877*	1878*	1879*	1880*	1881*	1882*	1883	1890*	1901*	1919*
		1957*	1960*	2096	2098*	2099	2101*	2102	2103*	2104*	2107	2123	2126*	2127
		2129*	2133	2135*	2155*	2159*	2161*	2162	2608	2612*	2616*	2618*	2620*	2626
		2627*	3048	3069*										
R2POS	006320	1590	1592#											
R3	=%000003	423#	1274*	1276	1583*	1586*	1623*	1624	1631*	1633	1689*	1726*	1763*	1764*

M07

.SAPT8	1#	390#	616#
.SAPTH	1#	390#	550
.SAPTY	1#	390#	2873
.SASTA	1#		
.SCATC	1#	390#	526
.SCMTA	1#	390#	572
.SDB2D	1#		
.SDB20	1#		
.SDIV	1#		
.SEOP	1#	390#	2200
.SEFR0	1#	390#	2695
.SEFR1	1#	390#	2747
.SMULT	1#		
.SPARM	390#		
.SPOWE	1#	390#	3040
.SRAND	1#	390#	
.SRDDE	1#		
.SRDOC	1#	390#	2594
.SREAD	1#	390#	2520
.SR2AZ	1#		
.SSAVE	1#	390#	
.SSB2D	1#		
.SSB20	1#		
.SSCOP	1#	390#	2632
.SSIZE	1#		
.SSPAC	390#		
.SSUPR	1#		
.SSWDO	390#		
.STRAP	1#	390#	3007
.STYP8	1#		
.STYPD	1#	390#	
.STYPE	1#	390#	2794
.STYPO	1#	390#	2930
.S40CA	1#		
.1170	1#		

INC	751 1726 3063	853 1744	901 1747	991 1751	1006 1754	1269 1758	1281 1761	1404 1862	1474 2003	1585 2214	1611 2680	1670 2716	1712 2915	1719 2986	1725 2994
INCB IOT JMP	2136 405 536	2140	2144	2685	2710	2869									
JSR	2198 763	2231 769	538 842	764 845	770 848	869 857	878 864	881 933	884 945	887 953	890 961	893 968	1431 996	1448 1010	1458 1022
	1033 1216 1300 1428 1550 1822 1987	1042 1227 1309 1436 1568 1826 1990	1045 1229 1311 1437 1572 1854 1998	1056 1239 1323 1444 1581 1911 2002	1076 1241 1326 1445 1584 1914 2012	1096 1251 1335 1453 1594 1916 2226	1113 1252 1337 1454 1706 1921 2722	1156 1255 1351 1455 1772 1939 2728	1158 1262 1353 1455 1778 1940 2822	1180 1263 1355 1521 1786 1942 2841	1191 1277 1363 1523 1790 1958 2848	1193 1286 1367 1524 1793 1961 2855	1202 1287 1377 1529 1804 1963 2904	1204 1289 1426 1531 1814 1981	1214 1298 1427 1539 1820 1984
MOV	750 805 834 923 998	756 806 858 930	757 807 859 931	762 808 860 932	768 809 861 943	776 813 862 944	783 814 863 952	784 817 866 960	791 818 874 967	795 819 896 974	799 820 897 976	801 825 898 977	802 827 905 982	803 828 912 990	804 829 913 992
	1068 1116 1176 1276 1385 1451 1511 1578 1654 1707 1818 1886 1918 2024 2034 2101 2161 2276 2361 2498 2690 2969	1069 1121 1176 1297 1389 1457 1515 1583 1657 1714 1819 1887 1919 2035 2035 2103 2163 2278 2363 2499 2691 2970	1018 1072 1122 1182 1308 1394 1471 1518 1592 1659 1721 1833 1888 1926 2038 2038 2105 2165 2280 2365 2499 2693 2975	1021 1075 1123 1190 1319 1399 1472 1519 1593 1662 1735 1834 1889 1937 2039 2039 2102 2162 2277 2362 2499 2694 2977	1032 1086 1127 1201 1320 1402 1473 1520 1600 1663 1736 1837 1890 1938 2052 2052 2115 2175 2290 2375 2499 2695 2978	1035 1087 1133 1213 1321 1403 1476 1527 1604 1666 1737 1838 1891 1941 2053 2053 2116 2176 2291 2376 2499 2696 2979	1036 1088 1140 1226 1322 1406 1480 1528 1605 1668 1763 1843 1892 1957 2054 2054 2119 2179 2294 2379 2499 2697 2980	1039 1089 1141 1238 1325 1409 1483 1533 1609 1682 1766 1844 1893 1960 2055 2055 2122 2182 2297 2382 2499 2698 2981	1040 1092 1152 1249 1330 1412 1485 1541 1610 1687 1777 1853 1894 1965 2056 2056 2121 2181 2296 2381 2499 2699 2982	1055 1098 1153 1254 1334 1417 1486 1547 1623 1688 1785 1856 1897 1995 2067 2067 2126 2186 2301 2386 2499 2699 2983	1057 1099 1154 1260 1349 1425 1489 1549 1624 1698 1788 1863 1900 1997 2068 2068 2127 2187 2302 2387 2499 2700 2984	1059 1105 1155 1261 1350 1430 1491 1552 1625 1699 1792 1875 1903 1997 2001 2070 2070 2137 2197 2312 2397 2499 2701 2985	1060 1108 1170 1270 1362 1434 1502 1567 1644 1703 1800 1876 1904 1904 2011 2076 2076 2138 2198 2313 2400 2499 2702 2986	1063 1109 1173 1273 1366 1439 1505 1571 1649 1704 1803 1877 1910 1910 2029 2085 2085 2140 2200 2314 2401 2499 2703 2987	1067 1110 1174 1274 1374 1447 1507 1577 1653 1705 1817 1883 1913 1913 2031 2086 2086 2141 2201 2315 2402 2499 2704 2988
MOV8	812 2690 2969	1124 2719 2970	1478 2727 2989	1601 2826 3018	1658 2854	1776 2862	1805 2876	2107 2877	2117 2879	2540 2956	2546 2957	2569 2960	2574 2961	2613 2962	2686 2966
NEG NOP RESET	1526 1151	1591 2211	1840 2227	2059 2228	2126 2229	2967									
ROL RTT RTS	935 754 914	2616 785 925	2618 826 1136	2620 2556 1145	2973 2584 1378	2975 2630 1993	2090 2976 2693	2225 2977 2746	2225 2979 2831	3001 1575 2063	3078 1595 2081	1620 2088	1638 2093	1678 2118	1917 2150
SUB	1924 2165 2718	1935 2784 2896	1943 2871 1329	1975 2921 1332	1977 3021 1548	1993 3021 1574	2013 3021 1588	2027 3021 1675	2049 3021 1764	2063 3021 1835	2081 3021 1845	2088 3021 1986	2093 3021 1992	2118 3021 2006	2150 3021 2057

	960	963	965	967	970	972	974	975	983	986	988	990	1000	1002	1004
	1005	1014	1016	1018	1024	1026	1028	1043	1047	1049	1051	1077	1082	1084	1086
	1087	1094	1101	1103	1105	1117	1119	1121	1130	1143	1148	1150	1152	1153	1166
	1168	1170	1171	1172	1186	1189	1190	1191	1197	1199	1201	1202	1209	1211	1213
	1214	1222	1224	1226	1227	1234	1236	1238	1239	1245	1247	1249	1250	1293	1295
	1297	1298	1304	1306	1308	1309	1315	1317	1319	1320	1343	1345	1347	1349	1350
	1358	1360	1362	1363	1370	1372	1374	1375	1392	1397	1398	1415	1416	1536	1537
	1544	1545	1555	1556	1627	1628	1636	1637	1769	1770	1859	1860	1866	1867	1932
	1968	1969	2073	2074	2079	2080	2154	2156	2202	2203	2204	2205	2206	2207	2211
	2217	2220	2222	2223	2225	2231	2233	2234	2236	2522	2524	2525	2528	2529	2557
	2565	2566	2570	2571	2587	2588	2594	2596	2599	2611	2634	2637	2642	2647	2659
	2661	2662	2663	2665	2666	2667	2676	2678	2686	2688	2693	2694	2695	2697	2700
	2710	2713	2720	2722	2723	2725	2732	2735	2742	2746	2747	2749	2764	2780	2796
	2817	2875	2877	2880	2907	2922	2932	3009	3015	3019	3023	3032	3033	3034	3035
	3036	3037	3038	3039	3040	3042	3052	3053	3058	3065	3066	3074	3076	3078	3082
.IFF	402	517	520	521	522	542	546	548	553	555	562	575	578	581	609
	616	619	750	779	781	799	877	880	883	886	889	892	903	909	918
	920	922	927	928	929	930	931	937	940	941	942	943	949	950	951
	952	957	958	959	960	964	965	966	967	971	972	973	974	984	987
	988	989	990	1001	1002	1003	1004	1015	1016	1017	1018	1025	1026	1027	1028
	1044	1048	1049	1050	1051	1078	1083	1084	1085	1086	1095	1102	1103	1104	1105
	1118	1119	1120	1121	1130	1143	1148	1149	1150	1151	1152	1167	1168	1169	1170
	1173	1187	1188	1189	1190	1198	1199	1200	1201	1210	1211	1212	1213	1223	1224
	1225	1226	1235	1236	1237	1238	1246	1247	1248	1249	1250	1294	1295	1296	1297
	1305	1306	1307	1308	1316	1317	1318	1319	1320	1344	1346	1347	1348	1349	1350
	1359	1360	1361	1362	1363	1371	1372	1373	1374	1375	1392	1398	1416	1417	1537
	1545	1556	1557	1628	1629	1637	1638	1770	1771	1860	1861	1867	1868	1932	1969
	2074	2079	2081	2154	2156	2203	2206	2211	2218	2221	2233	2523	2525	2529	2531
	2536	2557	2558	2567	2571	2588	2597	2635	2660	2663	2664	2667	2694	2695	2698
	2700	2713	2742	2747	2750	2765	2794	2797	2876	2933	3010	3016	3043	3059	3076
.IFT	2531	2536	2615	2631	2632	2675	2723	2722							
.IFTF	2525	2529	2532	2611	2615	2631	2673	2722							
.IIF	390	395	400	514	515	516	518	521	522	532	615	619	777	800	803
	809	810	811	813	814	906	1395	1413	1534	1542	1553	1625	1634	1767	1857
	1864	1966	2071	2077	2204	2205	2212	2213	2233	2236	2523	2580	2588	2594	2638
	2639	2640	2641	2642	2643	2647	2674	2675	2691	2694	2695	2701	2702	2703	2704
.IRP	2705	2710	2735	2742	2747	2762	2787	2873	3031	3032	3033	3034	3037	3038	3039
	680	927	939	948	956	963	970	986	1000	1014	1024	1047	1082	1101	1117
	1148	1166	1186	1197	1209	1222	1234	1245	1293	1304	1315	1345	1358	1370	2211
	2606	2627	2881	2882	2903	2919	2920	3046	3052	3065	3066				
.LIST	1	390	510	521	532	609	616	619	680	815	927	931	939	943	948
	952	956	960	963	967	970	974	986	990	1000	1004	1014	1018	1024	1028
	1047	1051	1082	1086	1101	1105	1117	1121	1148	1152	1166	1170	1186	1190	1197
	1201	1209	1213	1222	1226	1234	1238	1245	1249	1293	1297	1304	1308	1315	1319
	1345	1349	1358	1362	1370	1374	2212	2225	2557	2642	2742	3023	3031	3032	3033
	3034	3035	3037	3038	3039	3040									
.MACRO	1	522	572	683	831	3023									
.MCALL	390	510	616	815											
.MEXIT	664														
.NLIST	1	390	510	521	532	609	616	619	680	815	927	931	939	943	948
	952	956	960	963	967	970	974	986	990	1000	1004	1014	1018	1024	1028
	1047	1051	1082	1086	1101	1105	1117	1121	1148	1152	1166	1170	1186	1190	1197
	1201	1209	1213	1222	1226	1234	1238	1245	1249	1293	1297	1304	1308	1315	1319
	1345	1349	1358	1362	1370	1374	2212	2225	2557	2642	2742	3023	3031	3032	3033

	3034	3035	3037	3038	3039	3040									
.PAGE	572	665													
.REM	532														
.REPT	400	510	526	535	539	550	572	616	665	709	755	788	793	839	854
.SBTTL	927	939	948	956	963	970	986	1000	1014	1024	1047	1082	1101	1117	1146
	1148	1166	1186	1197	1209	1222	1234	1245	1293	1304	1315	1345	1358	1370	1384
	1424	1433	1450	1459	1501	1511	2200	2237	2520	2594	2632	2695	2747	2794	2873
	2930	3007	3023	3040											
.TITLE	390														
.WORD	532	533	534	547	566	567	568	569	570	571	580	583	584	585	586
	589	590	591	592	593	594	595	598	599	600	621	622	623	624	625
	626	627	628	632	633	634	647	651	654	657	658	659	660	661	662
	2217	2220	2232	2631	2773	2778	2823	2870	2905	3006	3075	3077			

ERRORS DETECTED: 0

*.DZADLB.SEG/SOL/CRF/NL:TOC=DSKZ:SYSMAC.SML,DZADLB.P11
RUN-TIME: 85 62 8 SECONDS
CORE USED: 33K

10			...B1	2130	011432	105037	...B5
63			...C1	2175	011642	000062	...C5
116			...D1	2209			...D5
			...F1	2246	012106	023440	...F5
172			...F1	2299	012470	020051	...F5
213			...G1	2351	013076	020040	...G5
241			...H1	2405	013556	020122	...H5
			...I1				...I5
297			...J1	2466	014162	020056	...J5
			...K1	2510	014512	001124	...K5
352			...L1	2529			...L5
399			...M1	2583	014750	012766	...M5
453		000020	...N1	2603			...N5
			...B2	2641			...B6
507		000060	...C2	2695			...C6
548		000214	...D2	2749			...D6
581	001102	000	...E2	2803			...E6
635			...F2	2857	016142	001372	...F6
674			...G2	2911	016412	005737	...G6
718	001336	000000	...H2	2939			...H6
764	001510	000137	...I2	2993	016660	002402	...I6
797	001644	022706	...J2	3016	016716	016600	...J6
845	002144	004737	...K2	3049	016776	010346	...K6
863	002244	012737	...L2	ADDW11=	000000		...L6
913	002522	012637	...M2	BASECH	001336		...M6
935	002612	006137	...N2	CLEAR1	006744		...N6
979	002760	001375					
			...B3	FIXADR	005500		...B7
1033	003152	004737	...C3	NOITST	010312		...C7
1087	003420	012737	...D3	RANDY	010722		...D7
1126	003614	001376	...E3				...E7
1155	003736	012737	...F3	SW10 =	002000		...F7
1206	004126	011626	...G3	TST34	005014		...G7
1254	004270	013704	...H3	\$ATYC	016244		...H7
1302	004512	011632	...I3	\$INTAG	001135		...I7
1324	004604	000062	...J3	\$STUP =	177777		...J7
1354	004746	000007	...K3				...K7
1393	005112	104400	...L3	NEWTST	1#	510#	...L7
1433			...M3	.SEOP	1#	390#	...M7
1468	005472	005337	...N3		2714	2736	...N7
1520	005770	013737					
			...B4		1216	1227	...B8
1570	006204	063702	...C4	.ASCIZ	611	614	...C8
1605	006362	012700	...D4		2217	2220	...D8
1649	006542	012737	...E4	.TITLE	390	3007	...E8
1688	006756	012701	...F4	**END**	USER	DAVIES, TOM	...F8
1742	007222	020227	...G4				
1794	007466	104400	...H4				
1837	007674	010120	...I4				
1886	010056	012700	...J4				
1934	010304	104400	...K4				
1987	010544	004537	...L4				
2037	011036	001376	...M4				
2073	011216	002	...N4				
2098	011310	005002					