

AA11-K

DIAGNOSTIC
MD-11-DZAAC-B

EP DZAAC B DL B
COPYRIGHT 1977
FICHE 1 OF 1

MAR 1977
digital
MADE IN USA

This microfiche card contains a grid of frames, each displaying diagnostic data for an MD-11 aircraft. The data is organized into columns and rows, with some frames containing text and others containing numerical data or status indicators. The frames are arranged in a grid that is approximately 10 columns wide and 15 rows high. The data appears to be organized into sections, possibly corresponding to different aircraft systems or components. The text is small and difficult to read, but it includes various alphanumeric codes and numbers. The overall layout is dense and structured, typical of a diagnostic manual or data sheet.

EDF:DPBAJ0501

00010000

770224

IDENTIFICATION

HDRIDZACBSEQ

00010000

770224
SEQ 000

PRODUCT CODE: MAINDEC-11-DZAAC-B
PRODUCT NAME: AA11-K DIAGNOSTIC TEST
DATE: JANUARY 1977
MAINTAINER: DIAGNOSTIC GROUP

FIRST PRINTING, 1976

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1976, 1977 BY DIGITAL EQUIPMENT CORPORATION

1. ABSTRACT

THIS DIAGNOSTIC EXERCISES THE "AA11-K" ANALOG CIRCUITRY. THE PROGRAM WHEN STARTED WILL TYPE OUT THE PROGRAM TITLE. A MESSAGE IS THEN PRINTED GIVING THE TWO LETTER DESIGNATOR TO BE TYPED TO RUN ANY ONE OF THE SIX (6) SEPERATE TESTS OF WHICH THIS PROGRAM IS COMPRISED. THE PROGRAM THEN TYPES A 'CR.' AND THEN WAITS IN A KEYBOARD MONITOR MODE FOR TWO LETTER'S TO BE TYPED. ALTHOUGH THESE TESTS MAY BE RUN IN ANY ORDER IT IS IMPERATIVE THAT TEST "AL" IS RUN FIRST AND VERIFY THAT THE AA11-K IS FULLY OPERATIONAL. THE PROGRAM IS SET UP TO GIVE THE OPERATOR AS MUCH CONTROL OVER THE PROGRAM AS POSSIBLE VIA THE TELETYPE. TYPING A '↑C' (OBTAINED VIA TYPING THE 'CNTR' AND 'C' KEYS SIMULTANEOUSLY) WHILE RUNNING ANY TEST WILL ENABLE THE PROGRAM TO RETURN TO THE KEYBOARD MONITOR AND AWAIT A NEW LETTER DESIGNATOR TO BE TYPED. TYPING A '↑G' WHILE RUNNING WILL ENABLE THE SOFTWARE SWITCH REGISTER VALUE TO BE CHANGED.

2. REQUIREMENTS (EQUIPMENT)

- A. PDP-11 COMPUTER WITH 8K OF MEMORY AND A CONSOLE I/O TERMINAL
- B. AA11-K QUAD OPTION MODULE INSTALLED
- C. <OPTIONAL> VR14 OR STORAGE SCOPE
- D. <OPTIONAL> UNIQUE HARDWARE TO PERFORM THE AUTO CALIBRATION TEST.

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING BINARY TAPES.

4. STARTING PROCEDURE

THE PROGRAM STARTING ADDRESS IS '200'.
THE RESTART ADDRESS IS '204'.

5. CONSOLE SWITCH SETTINGS

- THIS PROGRAM HAS BEEN MODIFIED TO RUN WITH OR WITHOUT A HARDWARE SWITCH REGISTER.
- A. ALL SWITCHES SHOULD BE DOWN (0) WHEN THE PROGRAM IS STARTED.
 - B. REFER TO THE INDIVIDUAL TEST DESCRIPTIONS FOR APPLICABLE CONSOLE SWITCH SETTINGS
 - C. ALL SWITCHES SET TO A 1 WILL SELECT SOFTWARE SWITCH CONTROL.

6. ERRORS

ALL ERRORS ARE ACCOMPANIED WITH A ENGLISH DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED FROM THE COMMENT AT THE ERROR PC OR THE TEST ITSELF.

7.0 TEST PROCEDURE

7.1 AUTO LOGIC TEST

A. THIS TEST IS DESIGNED TO VERIFY THE DATA PATH THAT IS ADDRESSABLE FROM THE CPU. THIS ALSO INCLUDES ALL CONTROL, INTERRUPT AND INITILIZE SIGNALS. THE LOGIC TEST ALSO INCLUDES PROVISIONS FOR TESTING MULTIPLE RA11-K'S.

B. STARTING SEQUENCE

1. TYPE 'AL' TO RUN THE AUTO LOGIC TEST.
2. THE PROGRAM WILL THEN EXECUTE A LOGIC TEST ON ALL AVAILABLE UNITS

C. CONTROL SWITCHES

1. TYPING ↑C WILL ENABLE THE PROGRAM TO EXIT AND RETURN TO THE KEYBOARD MONITOR.
2. TYPING ↑G WHEN SOFTWARE SWITCH REGISTER IS ENABLED WILL REPORT LAST VALUE AND WAIT FOR NEW VALUE.

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON CURRENT TEST
SW13=1	020000	INHIBIT ERROR TYPEOUT
SW12=1	010000	STORAGE SCOPE CONNECTED
SW11=1	004000	INHIBIT TEST INTERACTIONS
SW10=1	002000	EXTURNAL DELAY SIGNAL CONNECTED
SW09=1	001000	TWO'S COMPLIMENT MODE
SW08=1	0004XX	LOOP ON TEST IN SWR 7:0

D. ERRORS

REF. TO 6.

E. RESTRICTIONS

IF A STORAGE SCOPE IS CONNECTED, POWER MUST BE APPLIED TO IT.

F. EXECUTION TIME

IT TAKES APPROXIMATELY 5 SECONDS WITH OUT TEST INTERACTIONS.
IT TAKES APPROXIMATELY 30 SECONDS WITH TEST INTERACTIONS

7.2 AUTO DISPLAY TEST

A. THIS TEST IS DESIGNED TO AID IN THE ADJUSTING AND ALIGNMENT OF THE VR14 OR STORAGE SCOPE SCOPE ON THE AA11-K DISPLAY CONTROL.

B. TYPE 'AD' TO RUN THE AUTO VISUAL DISPLAY TEST. THE PROGRAM WILL THEN EXECUTE THE VISUAL DISPLAY TEST.

C. CONTROL SWITCHES

1. TYPING 1C AT ANY TIME WILL ENABLE THE PROGRAM TO EXIT AND RETURN TO THE MONITOR.
2. TYPING 1G WHEN SOFTWARE SWITCH REGISTER IS ENABLED WILL ENABLE THE PROGRAM TO CHANGE THE SOFTWARE SWITCH REGISTER VALUE.

CONSOLE SWITCHES	FUNCTION
CONSOLE SW10=1	SELECTE EXTERNAL DELAY MODE
CONSOLE SW09=1	SELECTS TWO'S COMPLEMENT MODE
CONSOLE SW08=0	CYCLE THRU ALL EIGHT DISPLAY PATTERNS
CONSOLE SW08=1	SELECT PATTERNS IN SW 00-02
CONSOLE SW07=0	HORIZONTAL SETTling TEST <SETTLING TEST>
CONSOLE SW07=1	VERTICAL SETTling TEST <SETTLING TEST>
CONSOLE SW05=0	STORAGE SCOPE NOT CONNECTED <PHOSPHOR TEST>
CONSOLE SW05=1	STORAGE SCOPE CONNECTED <PHOSPHOR TEST>
CONSOLE SW03=0	SELECT DAC 0 AND DAC 1
CONSOLE SW03=1	SELECT DAC 2 AND DAC 3
CONSOLE SW00-02=0	DISPLAY A HORIZONTAL LINE
CONSOLE SW00-02=1	DISPLAY A VERTICAL LINE
CONSOLE SW00-02=2	DISPLAY A SQUARE
CONSOLE SW00-02=3	DISPLAY AN "X"
CONSOLE SW00-02=4	DISPLAY SETTling TEST
CONSOLE SW00-02=5	DISPLAY CHARACTER TEST
CONSOLE SW00-02=6	DISPLAY CHANNEL TEST <VR14>
CONSOLE SW00-02=7	DISPLAY ERASE AND PHOSPHOR <STORAGE SCOPE>

D. ERRORS

THE ONLY ERRORS IN THIS TEST ARE DETECTED VISUALLY.

E. RESTRICTIONS

IF VR14, CHANNEL SWITCH MUST BE SET TO "1 & 2" POSITION.
 IF STORAGE SCOPE, POWER MUST BE APPLIED.
 THE "AUTO-CALIBRATION (AC)" MUST HAVE BEEN RUN PRIOR TO RUNNING THIS SECTION WHEN ON THE "AUTO HARDWARE TESTER".

F. EXECUTION TIME

IT TAKES APPROXIMATELY 60 SECONDS TO THIS TEST.

7.3 AUTO CALIBTARION

A. THIS TESTS REQUIRES UNIQUE HARDWARE. THE PROGRAM CONTROLS
A KNOWN GOOD AA11-K AND A PROGRAMABLE VOLTAGE STANDARD.
THE OPERATOR IS INSTRUCTED TO ADJUST THE POT'S.

B. STARTING SEQUENCE

1. TYPE 'AC' TO RUN THE AUTO CALIBRATION TEST.
2. THE PROGRAM WILL NOW INFORM THE OPERATOR WHAT TO DO.

C. CONTROL SWITCHES

<u>SWITCH</u>	<u>OCTAL</u>	<u>FUNCTION</u>
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON CURRNET OPERATION
SW13=1	020000	INHIBIT ERROR TYPEOUT
SW08=1	0004XX	LOOP ON SELECTED FUNCTION TEST

D. ERRORS

REF. TO 6.

E. RESTRICTIONS

TEST REQUIRES UNIQUE HARDWARE TO OPERATE.

F. EXECUTION TIME

OPERATOR DEPENDANT

MANUAL LOGIC LOOP

A. THIS LOOP IS PROVIDED TO ENABLE THE OPERATOR A SIMPLE PROGRAM LOOP TO AID IN REPAIR OF THE AA11-K. SWITCH REGISTER BITS 15:13 SELECT THE REGISTER TO BE LOADED AND BITS 12:00 CONTAINS THE DATA TO BE LOADED.

B. STARTING SEQUENCE

1. TYPE 'ML' TO RUN THE MANUAL LOGIC LOOP.
2. THE PROGRAM WILL NOW LOOP AND LOAD THE VALUE OF THE SWITCH REGISTER BITS 12:00 INTO THE SELECTED AA11-K REGISTER.

C. CONTROL SWITCHES

<u>SW15:13</u>	<u>REGISTER SELECTED</u>
000	DAC #0
001	DAC #1
010	DAC #2
011	DAC #3
1XX	STATUS REGISTER

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

E. RESTRICTIONS

NONE

F. EXECUTION TIME

THIS IS A NON-ENDING PROGRAM LOOP THAT CAN BE EXITED BY TYPING ↑C.

7.5 MANUAL DISPLAY LOOP

A. THIS LOOP IS PROVIDED FOR THE OPERATOR TO VERIFY COPERATION OF THE D/A CONVERTER AND MULTIPLEXER.

B. STARTING SEQUENCE

1. TYPE 'MD' TO RUN MANUAL DISPLAY LOOP
2. THE PROGRAM WILL NOW LOAD A "RAMP PATTERN" INTO EACH D/A CONVERTER.

C. CONTROL SWITCHES

1. TYPING ↑C WILL RETURN CONTROL TO THE KEYBOARD MONITOR.
2. SW10=1 WILL SELECT EXTURNAL DELAY MODE.

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

E. RESTRICTIONS

NONE

7.6 MANUAL CALIBRATION LOOP

A. THIS LOOP IS PROVIDED TO EANBLE THE OPERATOR TO ADJUST THE D/A CONVERTER.

B. STARTING SEQUENCE

1. TYPE 'MC' TO RUN THE MANUAL CALIBRATION LOOP.
2. THE PROGRAM WILL NOW LOAD THE CONTENTS OF THE SWITCH REGISTER INTO EACH D/A REGISTER AND AFTER A DELAY CLEAR THE D/A REGISTER.

C. CONTROL SWITCHES

1. TYPING ↑C WILL EXIT THE LOOP AND RETURN TO THE KEYBOARD MONITOR.
2. TYPING ↑G WHEN SOFTWARE SWITCH REGISTER IS ENABLED TO CHANGE THE SWITCH REGISTER VALUE
3. SWITCH REGISTER BITS 11:0 ARE LOADED IN TO ALL DAC'S.

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS.

E. RESTRICTIONS

NONE

8. AUTO DISPLAY TEST PATTERN DESCRIPTIONS

DISPLAY HORIZONTAL LINE

A HORIZONTAL LINE IS DISPLAYED ON THE SCOPE BY INITIALLY SETTING THE X AND Y DAC'S TO ZERO AND THEN INCREMENTING THE X VALUE WHILE HOLDING THE Y VALUE AT 4000. THE POINTS ARE DISPLAYED USING THE DISPLAY INTERRUPT ENABLED.

DISPLAY VERTICAL LINE

A VERTICAL LINE IS DISPLAYED ON THE SCOPE IN THE SAME MANNER AS FOR A HORIZONTAL LINE EXCEPT NOW THE Y VALUE IS INCREMENTED WHILE HOLDING THE X VALUE AT 1000.

DISPLAY SQUARE

A SQUARE IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE, THEN X IS INCREMENTED TO POSITIVE FULL SCALE (BOTTOM LINE) THEN Y IS INCREMENTED TO POSITIVE FULL SCALE (RIGHT LINE) THEN X IS DECREMENTED TO NEGATIVE FULL SCALE (TOP LINE) AND FINALLY Y IS DECREMENTED TO NEGATIVE FULL SCALE (LEFT LINE). MODE 01 (INTENSIFY ON LOADING X) AND MODE 10 (INTENSIFY ON LOADING Y) ARE USED.

DISPLAY X

AN X IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE AND THEN INCREMENTING BOTH TO POSITIVE FULL SCALE (LOWER LEFT TO UPPER RIGHT DIAGONAL) THEN X IS RESET TO NEGATIVE FULL SCALE, Y REMAINS AT POSITIVE FULL SCALE AND THEN X IS INCREMENTED WHILE Y IS DECREMENTED UNTIL BOTH REACH FULL SCALE AGAIN (UPPER LEFT TO LOWER RIGHT DIAGONAL). MODE 01 (INTENSIFY ON LOADING X) IS USED.

DISPLAY SETTLING TEST

A TWO CYCLE SQUARE WAVE WILL BE DISPLAYED TO TEST THE SETTLING DELAY. IF A SETTLING PROBLEM EXISTS, THE LEADING EDGES WILL APPEAR TO BE ROUNDED. THE PROGRAM PLOTS A LINE AT THE MINIMUM AXIS VALUE. UPON COMPLETION, ANOTHER LINE IS PLOTTED AT THE MAXIMUM VALUE. THIS IS REPEATED WITH THE RESULT BEING A SQUARE WAVE. SWITCH BIT 7 DETERMINES HORIZ., OR VERT. SETTLING PATTERN.

DISPLAY ALPHA-NUMERIC CHARACTER SET

THE ALPHABET AND NUMBERS 0 THRU 9 ARE DISPLAYED.
THE FIRST ROW CONSISTS OF THE LETTERS 'A' THRU 'M'. THE SECOND CONTAINS
THE LETTERS 'N' THRU 'Z'. THE LAST LINE CONTAINS THE NUMBERS
'0' THRU '9'.

DISPLAY CHANNEL 1 AND CHANNEL 2 <VR14>

THE TEXT "CHANNEL 1" IS DISPLAYED ON CHANNEL 1 SWITCH POSITION.
THE TEXT "CHANNEL 2" IS DISPLAYED ON CHANNEL 2 SWITCH POSITION.
THE COMBINED MESSAGE WILL APPEAR IF THE CHANNEL SELECTOR SWITCH IS
IN THE 1 & 2 POSITION.

PHOSPHOR AND ERASE TEST

THIS TEST PROVIDES A METHOD OF CHECKING FOR PHOSPHOR BURNS ON THE
SCREEN. THIS ROUTINE WILL FIRST ERASE THE STORAGE SCOPE SCREEN.
A DESCENDING HORIZONTAL LINE IS DISPLAYED IN 'STORE' MODE. THE RESULT
IS AN INTENSIFICATION OF THE ENTIRE SCREEN. IF EXECUTED USING A VR14,
OR DISPLAY SCOPE THE RESULT WILL BE ONLY A DESCENDING LINE.

9. MISCELLANEOUS

9.1 AA11-K BUS & VECTOR ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' IF BASE ADDRESS IS NOT 170416.
 MODIFY LOCATION '\$VECT1' IF THE VECTOR AND PRIORITY IS NOT 100360.

*NOTE IF EITHER VALUE IS CHANGED, THE PROGRAM MUST BE RESTARTED AT 200.

9.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP.
 THIS DIAGNOSTIC HAS THE "APT" HOOKS BUT HAS NOT BEEN TESTED.

9.3 POWER FAIL

A POWER FAIL WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED.

9.4 MULTIPLE AA11-K INTERFACE TESTING

THE PROGRAM WILL "AUTO-SIZE" THE NUMBER OF AA11-K'S. THE PROGRAM WILL AND REPORT THE NUMBER TO THE OPERATOR WHEN THE "AL" TEST IS SELECTED THE FIRST TIME.
 IF THE OPERATOR WISHES TO INHIBIT "AUTO-SIZE" BIT 15 OF LOCATION "\$ENV" MUST BE SET.

9.5 RESTRICTION

POWER MUST BE APPLIED TO A STORAGE SCOPE IF CONNECTED.

9.6 EXECUTION TIME

1. EXECUTION TIME OF THE AUTO LOGIC TEST IS:
 5 SECONDS WITHOUT INTERACTIONS.
 30 SECONDS WITH INTERACTIONS.
2. EXECUTION TIME OF THE AUTO DISPLAY TEST IS:
 60 SECONDS
3. EXECUTION TIME OF AUTO CALIBRATION IS:
 OPERATOR DEPENDANT
4. EXECUTION TIME OF THE MANUAL LOGIC LOOP IS:
 OPERATOR DEPENDANT
5. EXECUTION TIME OF THE MANUAL DISPLAY LOOP IS:
 OPERATOR DEPENDANT
6. EXECUTION TIME OF THE MANUAL CALIBRATION LOOP IS:
 OPERATOR DEPENDANT

10. TABLE OF CONTENTS

 ATTACHED

14 BASIC DEFINITIONS
21 OPERATIONAL SWITCH SETTINGS
23 TRAP CATCHER
(1) STARTING ADDRESS(ES)
27 ACT11 HOOKS
29 APT PARAMETER BLOCK
30 COMMON TAGS
(2) APT MAILBOX-ETABLE
(1) ERROR POINTER TABLE
136 INITIALIZE THE COMMON TAGS
140 SUBROUTINE TO LOAD A TRAP CATCHER
149 LOAD DEVICE ADDRESSES LOCATIONS

TEST # DESCRIPTION

171
172
173
177 INITIAL HEADER TYPEOUT AND WAIT FOR OPERATOR
227 DETERMINE THE NUMBER OF AA11-K ON THIS SYSTEM
272 T1 TEST THAT THE AA11-K RESPONDS TO THE CPU
285 T2 TEST THAT THE DAC0 REGISTER CAN BE CLEARED
293 T3 TEST THAT THE DAC0 REGISTER CAN BE LOADED WITH #7777
301 T4 TEST THAT THE DAC0 REGISTER CAN HOLD A FLOATING 1 PATTERN
311 T5 TEST THAT THE DAC #1 REGISTER CAN BE CLEARED
319 T6 TEST THAT THE Y REGISTER CAN BE LOADED WITH #7777
327 T7 TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN
338 T10 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
346 T11 TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
354 T12 TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
366 T13 TEST THAT THE DAC #3 REGISTER CAN BE CLEARED
374 T14 TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777
382 T15 TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN
392 T16 TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA
422 T17 TEST THAT RESET SETS READY BIT
430 T20 TEST THAT FAST INTENSIFY CAN BE SET AND CLEARED
444 T21 TEST THAT MODE BIT 2 CAN BE SET AND CLEARED
460 T22 TEST THAT MODE BIT 3 CAN BE SET
475 T23 TEST THAT EXT. DELAY (BIT 4) CAN BE SET AND CLEARED
489 T24 TEST THAT INTERRUPT ENABLE (BIT 6) CAN BE SET
497 T25 TEST THAT CHANNEL (BIT 9) CAN BE SET
505 T26 TEST THAT STORE (BIT 10) CAN BE SET
513 T27 TEST THAT WRITE THRU (BIT 11) CAN BE SET
522 T30 TEST THAT THE LOW BYTE OF THE STATUS REGISTER CAN BE CLEARED
532 T31 TEST THAT THE HIGH BYTE OF THE STATUS REGISTER CAN BE CLEARED
543 T32 TEST THAT WHEN INTENSIFY BIT IS SET THAT THE READY BIT CLEARS
561 T33 TEST THAT MODE 1 (INTENSIFY ON DAC0) CLEARS THE READY FLAG
592 T34 TEST THAT MODE 2 (INTENSIFY ON DAC1) CLEARS THE READY FLAG
618 T35 TEST WHEN ERASE IS SET, READY BIT CLEARS AND SET AFTER DELAY (SW12=1)
639 T36 TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED -1
654 T37 TEST THAT THE DISPLAY DOES NOT INTERRUPT AT LEVEL INDICATED
675 T40 TEST THAT RESET CLEARS MODE, EXT. DELAY AND FAST INTENSIFY BITS
683 T41 TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
691 T42 TEST THAT RESET CLEARS DAC0 REGISTER (SW09=0)
700 T43 TEST THAT RESET CLEARS DAC1 REGISTER (SW09=0)
710 T44 TEST THAT RESET CLEARS DAC #2 REGISTER (SW09=0)

TABLE OF CONTENTS

718 T45 TEST THAT RESET CLEARS DAC #3 REGISTER (SW09=0)
 738 T46 TEST THAT RESET SET DAC0 TO 4000 (SW09=1)
 739 T47 TEST THAT RESET SET DAC1 TO 4000 (SW09=1)
 740 T50 TEST THAT RESET SET DAC2 TO 4000 (SW09=1)
 741 T51 TEST THAT RESET SET DAC3 TO 4000 (SW09=1)
 742 T52 TEST THAT EXTERNAL DELAY DOES NOT SET DISPLAY READY SW10=0
 759 T53 TEST THE EXTERNAL DELAY DOES SET DISPLAY READY (SW10=1)
 775 T54 DETERMINE IF MORE AA11-K'S REMAIN TO BE TESTED
 791 END OF PASS ROUTINE

794
795
796
797
798
799
800

VISUAL TEST PATTERNS

801
802
803
818
832
868
925
975
1040
1166
1204
1216
1217
1218
1219

 DISPLAY HORIZONTAL LINE
 DISPLAY A VERTICAL LINE
 PINCUSHION TEST (DISPLAY SQUARE)
 PLOT AN X
 SCOPE SETTling TIME TEST
 PLOT CHARACTER SET
 CHANNEL 1 AND CHANNEL 2 TEST
 PHOSPHOR TEST

DAC ADJUSTMENT ROUTINES

1221
1248
1306
1316
1326
1336
1348
1357
1366
1375
1391
1402
1415
1433
1477

 MANUAL DISPLAY ROUTINE
 AUTO CALIBRATION
 T55 TEST THAT CH 3 IS AT +2.5 BIPOLAR
 T56 TEST THAT CH 4 IS AT -2.5 BIPOLAR
 T57 TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
 T60 TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
 T61 TEST THAT WRITE-THRU (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
 T62 TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
 T63 TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
 T64 TEST THAT DELAY RETURN SETS READY
 T65 TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
 T66 TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
 T67 TEST THAT "ERASE" AND DONE CAN BE CLEARED AND SET
 T70 ADJUSTMENT ROUTINES FOR DAC 0 - 1
 T71 ADJUSTMENT ROUTINES FOR DAC 2 - 3

1540
1610
1623
1645
1646
1647
1651
1678

MANUAL LOGIC TEST
 MANUAL DAC CALIBRATION
 DYNAMIC DAC CALIBRATION

 MISC. SUB-ROUTINES, ASCII MESSAGES AND SOFTWARE HANDLERS

 SUBROUTINE TO ERASE STORAGE SCOPE SCREEN
 SUBROUTINE TO DRAW A HORIZONTAL LINE

1753	TIMER ROUTINE FOR VISUAL TEST PATTERNS
1876	ASCII MESSAGES
1999	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2000	SCOPE HANDLER ROUTINE
2001	ERROR HANDLER ROUTINE
2002	ERROR MESSAG TYPEOUT ROUTINE
2003	POWER DOWN AND JP ROUTINES
2007	BINARY TO OCTAL (ASCII) AND TYPE
2008	TYPE ROUTINE
2009	TTY INPUT ROUTINE
2010	READ AN OCTAL NUMBER FROM THE TTY
2011	APT COMMUNICATIONS ROUTINE
2013	TRAP DECODER
(3)	TRAP TABLE


```

(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

```

```

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

```

```

(1) 100000
(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

```

```

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

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES

(1)

(1) 000004
 (1) 000010
 (1) 000014
 (1) 000014
 (1) 000014
 (1) 000020
 (1) 000024
 (1) 000030
 (1) 000034
 (1) 000060
 (1) 000064
 (1) 000240
 15
 16 170416
 17 100360
 18 000200

ERRVEC= 4
 RESVEC= 10
 TBITVEC=14
 TRTVEC= 14
 BPTVEC= 14
 IOTVEC= 20
 PWRVEC= 24
 EMTVEC= 30
 TRAPVEC=34
 TKVEC= 60
 TPVEC= 64
 PIRQVEC=240

ABASE=170416
 AVECT1=100360
 APRIOR=200

:: TIME OUT AND OTHER ERRORS
 :: RESERVED AND ILLEGAL INSTRUCTIONS
 :: "T" BIT
 :: TRACE TRAP
 :: BREAKPOINT TRAP (BPT)
 :: INPUT/OUTPUT TRAP (IOT) **SCOPE**
 :: POWER FAIL
 :: EMULATOR TRAP (EMT) **ERROR**
 :: "TRAP" TRAP
 :: TTY KEYBOARD VECTOR
 :: TTY PRINTER VECTOR
 :: PROGRAM INTERRUPT REQUEST VECTOR

20
21
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
22
23
(1)
(1)
(1)
(1)
(1)
(1)
(1)
24
25

000000

000174 000174
000174 000000
000176 000000

000200 000137 001462
000204 000137 001500
000210 000137 013434

```
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:* SWITCH USE
:* -----
:* 15 HALT ON ERROR
:* 14 LOOP ON TEST
:* 13 INHIBIT ERROR TIMEOUTS
:* 12 STORAGE SCOPE CONNECTED
:* 11 INHIBIT ITERATIONS
:* 10 EXTERNAL DELAY SIGNAL CONNECTED
:* 8 LOOP ON TEST IN SWR<7:0>
:* 9 TWO'S COMPLEMENT MODE
.SBTTL TRAP CATCHER
.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
JMP BEGIN1 ;JUMP TO RESTART ADDRESS
JMP DYNCAL ;JUMP TO DYNAMIC DAC CALIBRATION
```


(2)	001202	000000	\$PASS:	.WORD	APASS	::	PASS COUNT
(2)	001204	000000	\$DEVCT:	.WORD	ADEVCT	::	DEVICE COUNT
(2)	001206	000000	\$UNIT:	.WORD	AUNIT	::	I/O UNIT NUMBER
(2)	001210	000000	\$MSGAD:	.WORD	AMSGAD	::	MESSAGE ADDRESS
(2)	001212	000000	\$MSGLG:	.WORD	AMSGLG	::	MESSAGE LENGTH
(2)	001214		\$ETABLE:			::	APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV:	.BYTE	AENV	::	ENVIRONMENT BYTE
(2)	001215	000	\$ENVM:	.BYTE	AENVM	::	ENVIRONMENT MODE BITS
(2)	001216	000000	\$SWREG:	.WORD	ASWREG	::	APT SWITCH REGISTER
(2)	001220	000000	\$USWR:	.WORD	AUSWR	::	USER SWITCHES
(2)	001222	000000	\$CPUOP:	.WORD	ACPUOP	::	CPU TYPE, OPTIONS
(2)			*			::	BITS 15-11=CPU TYPE
(2)			*			::	11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
(2)			*			::	11/70=06, PDQ=07, Q=10
(2)			*			::	BIT 10=REAL TIME CLOCK
(2)			*			::	BIT 9=FLOATING POINT PROCESSOR
(2)			*			::	BIT 8=MEMORY MANAGEMENT
(2)	001224	000	\$MAMS1:	.BYTE	AMAMS1	::	HIGH ADDRESS, M.S. BYTE
(2)	001225	000	\$MTYP1:	.BYTE	AMTYP1	::	MEM. TYPE, BLK#1
(2)			*			::	MEM. TYPE BYTE -- (HIGH BYTE)
(2)			*			::	900 NSEC CORE=001
(2)			*			::	300 NSEC BIPOLAR=002
(2)			*			::	500 NSEC MOS=003
(2)	001226	000000	\$MADR1:	.WORD	AMADR1	::	HIGH ADDRESS, BLK#1
(2)			*			::	MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
(2)	001230	000	\$MAMS2:	.BYTE	AMAMS2	::	HIGH ADDRESS, M.S. BYTE
(2)	001231	000	\$MTYP2:	.BYTE	AMTYP2	::	MEM. TYPE, BLK#2
(2)	001232	000000	\$MADR2:	.WORD	AMADR2	::	MEM. LAST ADDRESS, BLK#2
(2)	001234	000	\$MAMS3:	.BYTE	AMAMS3	::	HIGH ADDRESS, M.S. BYTE
(2)	001235	000	\$MTYP3:	.BYTE	AMTYP3	::	MEM. TYPE, BLK#3
(2)	001236	000000	\$MADR3:	.WORD	AMADR3	::	MEM. LAST ADDRESS, BLK#3
(2)	001240	000	\$MAMS4:	.BYTE	AMAMS4	::	HIGH ADDRESS, M.S. BYTE
(2)	001241	000	\$MTYP4:	.BYTE	AMTYP4	::	MEM. TYPE, BLK#4
(2)	001242	000000	\$MADR4:	.WORD	AMADR4	::	MEM. LAST ADDRESS, BLK#4
(2)	001244	100360	\$VECT1:	.WORD	AVECT1	::	INTERRUPT VECTOR#1, BUS PRIORITY#1
(2)	001246	000000	\$VECT2:	.WORD	AVECT2	::	INTERRUPT VECTOR#2, BUS PRIORITY#2
(2)	001250	170416	\$BASE:	.WORD	ABASE	::	BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001252	000000	\$DEVN:	.WORD	ADEVN	::	DEVICE MAP
(2)	001254	000000	\$CDW1:	.WORD	ACDW1	::	CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:			::	
(2)			.MEXIT				

71			; ITEM	7					
72	001336	016170		EM7					:AA11-K INTERRUPTED IN ERROR
73	001340	016625		DH6					:ERRPC IBASE
74	001342	020320		DT6					:SERRPC STAT
75	001344	020364		DF0					
76									
77			; ITEM	10					
78	001346	016224		EM10					:BUS TIME-OUT ON AA11-K ADDRESSES
79	001350	016625		DH6					:ERRPC IBASE
80	001352	020320		DT6					:SERRPC STAT
81	001354	020364		DF0					
82									
83			; ITEM	11					
84	001356	016275		EM11					:AA11-K READY BIT ERROR
85	001360	016571		DH1					:ERRPC IBASE GOOD BAD
86	001362	020236		DT1					:SERRPC STAT \$GDDAT \$BDDAT
87	001364	020364		DF0					
88									
89			; ITEM	12					
90	001366	016324		EM12					:POWER SUPPLY VOLTAGE WAS INCORRECT
91	001370	016673		DH14					:ERRPC BASE GOOD BAD
92	001372	020340		DT14					:SERRPC \$BASE \$GDDAT \$BDDAT
93	001374	020364		DF0					
94									
95			; ITEM	13					
96	001376	016367		EM13					:A PREVIOUSLY EXISTING AA11-K DOES NOT RESPOND NOW
97	001400	016641		DH13					:ERRPC BASE FIRST# #NOW
98	001402	020326		DT13					:SERRPC \$BASE EVER \$UNIT
99	001404	020364		DF0					
100									
101			; ITEM	14					
102	001406	016410		EM14					:AA11-K LOGIC SIGNAL HIGH OUTPUT TO LOW
103	001410	016673		DH14					:ERRPC BASE GOOD BAD
104	001412	020340		DT14					:SERRPC \$BASE \$GDDAT \$BDDAT SPREAD
105	001414	020364		DF0					
106									
107			; ITEM	15					
108	001416	016457		EM15					:AA11-K LOGIC LOW OUTPUT TO HIGH
109	001420	016673		DH14					:ERRPC BASE GOOD BAD
110	001422	020340		DT14					:SERRPC \$BASE \$GDDAT \$BDDAT SPREAD
111	001424	020364		DF0					
112									
113			; ITEM	16					
114	001426	016526		EM16					:SELECTED DAC HAS A LINEARITY ERROR
115	001430	016720		DH16					:ERRPC DACADR GOOD BAD
116	001432	020352		DT16					:SERRPC XDACUT \$GDDAT \$BDDAT
117	001434	020364		DF0					
118	001436	000040	VADDR:	40					:ADJUSTMENT TO NEXT AA11-K'S ADDRESS
119	001440	000040	VVECT:	40					:ADJUSTMENT TO NEXT AA11-K'S VECTOR
120	001442	000012	DELAY:	10.					:DELAY COUNTER FOR DELAY LOOPS
121	001444	170416	STAT:	170416					
122	001446	170420	DACO:	170420					
123	001450	170422	DAC1:	170422					
124	001452	170424	DAC2:	170424					

```

125 001454 170426 DAC3: 170426
126 001456 000350 IV: 350
127 001460 000352 IVS: 352
128
129 001462 005037 020416 BEGIN: CLR TEMP
130 001466 005037 020426 CLR EVER ;RESET POINTER
131 001472 005037 020430 CLR EVER1
132 001476 000403 BR BEG
133 001500 012737 000001 020416 BEGIN1: MOV #1,TEMP
134 001506 000005 BEG: RESET
136 .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001510 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 001514 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 001516 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 001522 001374 BNE -6 ;;LOOP BACK IF NO
(1) 001524 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 001530 012737 020730 000020 MOV #SSCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001536 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
(1) 001544 012737 021144 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001552 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
(1) 001560 012737 023504 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001566 012737 000340 000036 MOV #340,@TRAPVEC+2 ;;LEVEL 7
(1) 001574 012737 021430 000024 MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
(1) 001602 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
(1) 001610 005037 001166 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001614 012737 001614 001106 MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001622 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 001626 012737 001662 000004 MOV #64$,@ERRVEC ;;SET UP ERROR VECTOR
(2) 001634 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001642 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001650 022777 177777 177262 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 001656 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001660 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 001662 012716 001670 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
(2) 001666 000002 RTI
(2) 001670 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
(2) 001676 012737 000174 001142 MOV #DISPREG,DISPLAY
(2) 001704 012637 000004 66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 001710 005037 001202 CLR $PASS ;;CLEAR PASS COUNT
(2) 001714 132737 000200 001215 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 001722 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
(2) 001724 012737 001216 001140 MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 001732
137 001732 005037 177776 67$: CLR @PS
138 001736 000137 002126 JMP INIT1

```



```

140          .SBTTL SUBROUTINE TO LOAD A TRAP CATCHER
141 001742 012702 000242 LDTRAP: MOV #242,R2 ;LOAD R2
142 001746 012701 000240          MOV #240,R1 ;LOAD R1
143 001752 010221          5$: MOV R2,(R1)+ ;LOAD .+2
144 001754 005021          CLR (R1)+ ;LOAD HALT
145 001756 010102          MOV R1,R2 ;LOAD R2
146 001760 005722          TST (R2)+ ;BUMP R2
147 001762 020227 001002          CMP R2,#1002 ;TEST FOR LAST
148 001766 001371          BNE 5$ ;BR UNTIL DONE
149          .SBTTL LOAD DEVICE ADDRESSES LOCATIONS
150 001770 012700 001444          MOV #STAT,RO ;LOAD POINTER
151 001774 013720 001250          10$: MOV $BASE,(RO)+ ;LOAD BASE ADDRESS
152 002000 022700 001456          CMP #IV,RO ;TEST FOR DONE
153 002004 001373          BNE 10$ ;BR
154 002006 012700 001446          MOV #DACD,RO ;LOAD 2ND ADDRESS
155 002012 012701 000002          MOV #2,R1 ;LOAD R1
156 002016 060120          12$: ADD R1,(RO)+ ;UPDATE REAL DEVICE WORD
157 002020 005721          TST (R1)+ ;BUMP R1
158 002022 022701 000012          CMP #12,R1 ;TEST FOR DONE
159 002026 001373          BNE 12$ ;BR
160 002030 013737 001244 001456          MOV $VECT1,IV ;LOAD INTR. VECTOR ADDRESS
161 002036 042737 160000 001456          BIC #160000,IV
162 002044 013737 001456 001460          MOV IV,IVS
163 002052 062737 000002 001460          ADD #2,IVS
164 002060 113737 001245 020422          MOVB $VECT1+1,BRLEV1 ;LOAD BR LEVEL
165 002066 042737 177437 020422          BIC #177437,BRLEV1
166 002074 013737 020422 020424          MOV BRLEV1,BRLEV2
167 002102 162737 000040 020424          SUB #40,BRLEV2
168 002110 000207          RTS ;EXIT
175

```

```

177          .SBTTL INITIAL HEADER TYPEOUT AND WAIT FOR OPERATOR
178 002112 046101 AL: .ASCII /AL/
179 002114 042101 AD: .ASCII /AD/
180 002116 041501 AC: .ASCII /AC/
181 002120 046115 ML: .ASCII /ML/
182 002122 042115 MD: .ASCII /MD/
183 002124 041515 MC: .ASCII /MC/
184
185 002126 004737 001742 INIT1: JSR PC_LDTRAP
186 002132 005737 020416 TST TEMP ;TEST IF START OR RESTART
187 002136 001065 BNE CTRLC ;RESTART
188 002140 005737 000042 TST J#42 ;TEST IF MONITOR
189 002144 001067 BNE MTEST ;BR IF NOT
190 002146 104401 TYPE ;CALL MESSAGE PRINTER VIA 'EMT'
191 002150 014606 TITLE ;TYPE PROGRAM HEADER.
192 002152 012706 001100 WAITIN: MOV #STACK,SP ;RESET STACK
193 002156 104407 CKSWR ;TEST FOR CTRL G
194 002160 104411 RDLIN ;READ TWO CHARACTERS
195 002162 013637 020432 MOV J(SP)+,OPRIN ;READ THE CHARACTER
196 002166 042737 000000 020432 BIC #0,OPRIN ;MASK OTHER CHARACTERS
197
198 002174 023737 002112 020432 CMP AL,OPRIN ;TEST FOR "AL"
199 002202 001427 BEQ 10$ ;BR IF YES
200 002204 023737 002114 020432 CMP AD,OPRIN ;TEST FOR "AD"
201 002212 001425 BEQ 11$ ;BR IF YES
202 002214 023737 002116 020432 CMP AC,OPRIN ;TEST FOR "AC"
203 002222 001423 BEQ 12$ ;BR IF YES
204 002224 023737 002120 020432 CMP ML,OPRIN ;TEST FOR "ML"
205 002232 001421 BEQ 13$ ;BR FI YES
206 002234 023737 002122 020432 CMP MD,OPRIN ;TEST FOR "MD"
207 002242 001417 BEQ 14$ ;BR IF YES
208 002244 023737 002124 020432 CMP MC,OPRIN ;TEST FOR "MC"
209 002252 001415 BEQ 15$ ;BR IF YES
210 002254 104401 TYPE
211 002256 016746 QMARK
212 002260 000734 BR WAITIN ;WAIT FOR OPERATOR AGAIN
213
214 002262 000137 002324 10$: JMP MTEST ;RUN THE LOGIC TEST
215 002266 000137 006622 11$: JMP VISUAL ;RUN THE VISUAL PATTERN
216 002272 000137 011506 12$: JMP AUTCAL ;RUN THE AUTO CALIBRATION (FACTORY ONLY)
217 002276 000137 013104 13$: JMP MANUL ;RUN THE MANUAL LOGIC TEST
218 002302 000137 011372 14$: JMP FULRMP ;RUN THE RAMP PATTERN ON FOUR DAC'S
219 002306 000137 013370 15$: JMP CALDAC ;RUN THE MANUAL CAL OF THE DAC
220
221 002312 104401 CTRLC: TYPE
222 002314 016754 CONTC
223 002316 005037 001202 CLR $PASS
224 002322 000713 BR WAITIN
225

```

```

227 .SBTTL DETERMINE THE NUMBER OF AA11-K ON THIS SYSTEM
228
229 002324 013737 001250 001126 MTEST: MOV $BASE,$BDDAT ;GET THE BASE ADDRESS
230 002332 005037 001206 CLR $UNIT ;CLEAR UNIT #
231 002336 012737 002364 000004 MOV #2,$ERRVEC ;LOAD TRAP RETURN
232 002344 005777 176556 1S: TST @BDDAT ;TEST IF ADDR EXISTS
233 002350 063737 001436 001126 ADD VADDR,$BDDAT ;UPDATE THE BUS ADDRESS
234 002356 005237 001206 INC $UNIT ;UPDATE UNIT COUNT
235 002362 000406 BR 3$
236 002364 022626 001206 2$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
237 002366 005737 TST $UNIT ;TEST IF ANY EXIST
238 002372 001002 BNE 3$ ;BR IF SOME ARE THERE
239 002374 104010 ERROR 10 ;BASE ADDRESS CAUSED AN BUS TRAP
240 002376 000434 BR TST1
241 002400 005737 020426 3$: TST EVER ;TEST IF # HAS BEEN REPORTED
242 002404 100417 BMI 4$ ;BR IF IT HAS
243 002406 104401 TYPE
244 002410 016762 FOUND1 ;TELL OPERATOR THE # OF AA11-K'S
245 002412 013746 001206 MOV $UNIT,-(SP)
246 002416 104403 TYPOS
247 002420 002 .BYTE 2
248 002421 000 .BYTE 0
249 002422 104401 TYPE
250 002424 017006 FOUND2
251 002426 013737 001206 020426 MOV $UNIT,EVER ;SAVE THE # OF AA11-K'S FOR LATER
252 002434 052737 100000 020426 BIS #BIT15,EVER ;SET "REPORTED # FLAG"
253 002442 000405 BR 5$
254 002444 123737 020426 001206 4$: CMPB EVER,$UNIT ;TEST IF ANY HAVE GONE AWAY
255 002452 001401 BEQ 5$ ;BR IF ALL ARE STILL HERE
256 002454 104013 ERROR 13 ;EXISTING UNIT FAILED TO RESPOND NOW
257 002456 005037 001206 5$: CLR $UNIT ;RESET UNIT POINTER
258 002462 004737 001742 JSR PC,LDTRAP ;LOAD TRAP CATCHER AND BUS ADDRESSES
259
260 002466 000240 LTEST: NOP
272 ;*****
(3) ;*TEST 1 TEST THAT THE AA11-K RESPONDS TO THE CPU
(3) ;*****
(2) 002470 000004 TST1: SCOPE
273 002472 012737 002506 001106 MOV #3,$LPADR
274 002500 012737 002534 000004 MOV #1,$ERRVEC ;LOAD BUS TRAP RETURN
275 002506 005777 176732 3$: TST @STAT ;TEST THE STATUS REGISTER
276 002512 005777 176730 TST @DAC0 ;TEST THE DAC0
277 002516 005777 176726 TST @DAC1 ;TEST THE DAC1
278 002522 005777 176724 TST @DAC2 ;TEST DAC #2
279 002526 005777 176722 TST @DAC3 ;TEST DAC #3
280 002532 000402 BR 2$ ;BR AND RESTORE LOC. 4
281 002534 022626 1S: CMP (SP)+,(SP)+ ;CLEAN THE STACK
282 002536 104010 ERROR 10 ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE AA11-K
283 002540 012737 000006 000004 2$: MOV #6,$ERRVEC ;LOAD RETURN

```

```

285      ;:*****
(3)      ;*TEST 2      TEST THAT THE DACO REGISTER CAN BE CLEARED
(3)      ;:*****
(2)      002546 000004      †ST2:  SCOPE
286      002550 005037 001124      CLR      $GDDAT      ;LOAD EXPECTED
287      002554 013777 001124 176664      MOV      $GDDAT,‡DACO      ;LOAD REG
288      002562 017737 176660 001126      MOV      ‡DACO,$BDDAT      ;READ REG
289      002570 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
290      002576 001401      BEQ      TST3      ;;BR IF EQUAL
291      002600 104002      ERROR    2      ;ERROR, DACO REGISTER NOT = 0
292
293      ;:*****
(3)      ;*TEST 3      TEST THAT THE DACO REGISTER CAN BE LOADED WITH #7777
(3)      ;:*****
(2)      002602 000004      †ST3:  SCOPE
294      002604 012737 007777 001124      MOV      #7777,$GDDAT      ;LOAD EXPECTED
295      002612 013777 001124 176626      MOV      $GDDAT,‡DACO      ;LOAD REG
296      002620 017737 176622 001126      MOV      ‡DACO,$BDDAT      ;READ REG
297      002626 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
298      002634 001401      BEQ      TST4      ;;BR IF EQUAL
299      002636 104002      ERROR    2      ;ERROR, DACO REGISTER NOT = 7777
300
301      ;:*****
(3)      ;*TEST 4      TEST THAT THE DACO REGISTER CAN HOLD A FLOATING 1 PATTERN
(3)      ;:*****
(2)      002640 000004      †ST4:  SCOPE
(1)      002642 012737 000100 001166      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
302      002650 012737 004000 001124      MOV      #BIT11,$GDDAT      ;LOAD EXPECTED
303      002656 013777 001124 176562 1$:  MOV      $GDDAT,‡DACO      ;LOAD DACO REGISTER
304      002664 017737 176556 001126      MOV      ‡DACO,$BDDAT      ;READ THE REGISTER
305      002672 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE THE DATA
306      002700 001401      BEQ      2$      ;;BR IF SAME
307      002702 104002      ERROR    2      ;ERROR, DACO REGISTER FAILED TO HOLD A FLOATING
308      002704 006237 001124 2$:  ASR      $GDDAT      ;CHANGE THE DATA
309      002710 001362      BNE     1$      ;BR AND TEST MORE DATA

```

```

311      ;:*****
(3)      ;*TEST 5      TEST THAT THE DAC #1 REGISTER CAN BE CLEARED
(3)      ;:*****
(2)      002712  000004      †ST5:  SCOPE
312      002714  005037      CLR      $GDDAT      ;LOAD EXPECTED
313      002720  013777      001124  176522  MOV      $GDDAT, @DAC1 ;LOAD Y REG
314      002726  017737      176516  001126  MOV      @DAC1, $BDDAT ;READ REG
315      002734  023737      001124  001126  CMP      $GDDAT, $BDDAT ;COMPARE
316      002742  001401      BEQ      TST6      ;;BR IF EQUAL
317      002744  104003      ERROR    3          ;ERROR, DAC1 REGISTER NOT = 0
318
319      ;:*****
(3)      ;*TEST 6      TEST THAT THE Y REGISTER CAN BE LOADED WITH #7777
(3)      ;:*****
(2)      002746  000004      †ST6:  SCOPE
320      002750  012737      007777  001124  MOV      #7777, $GDDAT ;LOAD EXPECTED
321      002756  013777      001124  176464  MOV      $GDDAT, @DAC1 ;LOAD Y REG
322      002764  017737      176460  001126  MOV      @DAC1, $BDDAT ;READ REG
323      002772  023737      001124  001126  CMP      $GDDAT, $BDDAT ;COMPARE
324      003000  001401      BEQ      TST7      ;;BR IF EQUAL
325      003002  104003      ERROR    3          ;ERROR, DAC1 REGISTER NOT = 7777
326
327      ;:*****
(3)      ;*TEST 7      TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN
(3)      ;:*****
(2)      003004  000004      †ST7:  SCOPE
(1)      003006  012737      000100  001166  MOV      #100, $TIMES ;;DO 100 ITERATIONS
328      003014  012737      004000  001124  MOV      #BIT11, $GDDAT ;LOAD EXPECTED
329      003022  013777      001124  176420  1$:  MOV      $GDDAT, @DAC1 ;LOAD THE REGISTER
330      003030  017737      176414  001126  MOV      @DAC1, $BDDAT ;READ THE REGISTER
331      003036  023737      001124  001126  CMP      $GDDAT, $BDDAT ;COMPARE THE DATA
332      003044  001401      BEQ      2$      ;;BR IF DATA IS SAME
333      003046  104003      ERROR    3          ;ERROR, DAC1 REGISTER FAILED TO HOLD A FLOATING
334      003050  006237      001124  2$:  ASR      $GDDAT ;CHANGE THE DATA
335      003054  001362      BNE     1$      ;BR AND TEST MORE DATA

```

337
338
(3)
(3)
(2)
339
340
341
342
343
344
345
346
(3)
(3)
(2)
347
348
349
350
351
352
353
354
(3)
(3)
(2)
(1)
355
356
357
358
359
360
361
362

003056 000004
003060 005037 001124
003064 013777 001124 176360
003072 017737 176354 001126
003100 023737 001124 001126
003106 001401
003110 104004

003112 000004
003114 012737 007777 001124
003122 013777 001124 176322
003130 017737 176316 001126
003136 023737 001124 001126
003144 001401
003146 104004

003150 000004
003152 012737 000100 001166
003160 012737 004000 001124
003166 013777 001124 176256
003174 017737 176252 001126
003202 023737 001124 001126
003210 001401
003212 104004
003214 006237 001124
003220 001362

*TEST 10 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED

†ST10: SCOPE
CLR \$GDDAT ;LOAD EXPECTED
MOV \$GDDAT, @DAC2 ;LOAD REG
MOV @DAC2, \$BDDAT ;READ REG
CMP \$GDDAT, \$BDDAT ;COMPARE
BEQ TST11 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 0

*TEST 11 TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777

†ST11: SCOPE
MOV #7777, \$GDDAT ;LOAD EXPECTED
MOV \$GDDAT, @DAC2 ;LOAD REG
MOV @DAC2, \$BDDAT ;READ REG
CMP \$GDDAT, \$BDDAT ;COMPARE
BEQ TST12 ;;BR IF EQUAL
ERROR 4 ;ERROR, DAC #2 REGISTER NOT = 7777

*TEST 12 TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN

†ST12: SCOPE
MOV #100, \$TIMES ;;DO 100 ITERATIONS
MOV #BIT11, \$GDDAT ;LOAD EXPECTED
1\$: MOV \$GDDAT, @DAC2 ;LOAD X REGISTER
MOV @DAC2, \$BDDAT ;READ THE REGISTER
CMP \$GDDAT, \$BDDAT ;COMPARE THE DATA
BEQ 2\$;;BR IF SAME
ERROR 4 ;ERROR, DAC #2 REGISTER FAILED TO HOLD A FLOATIN
2\$: ASR \$GDDAT ;CHANGE THE DATA
BNE 1\$;BR AND TEST MORE DATA

364
365
366
(3)
(3)
(2)
367
368
369
370
371
372
373
374
(3)
(3)
(2)
375
376
377
378
379
380
381
382
(3)
(3)
(2)
(1)
383
384
385
386
387
388
389
390

003222 000004
003224 005037 001124
003230 013777 001124 176216
003236 017737 176212 001126
003244 023737 001124 001126
003252 001401
003254 104005

003256 000004
003260 012737 007777 001124
003266 013777 001124 176160
003274 017737 176154 001126
003302 023737 001124 001126
003310 001401
003312 104005

003314 000004
003316 012737 000100 001166
003324 012737 004000 001124
003332 013777 001124 176114
003340 017737 176110 001126
003346 023737 001124 001126
003354 001401
003356 104005
003360 006237 001124
003364 001362

*TEST 13 TEST THAT THE DAC #3 REGISTER CAN BE CLEARED

TST13: SCOPE
CLR \$GDDAT ;LOAD EXPECTED
MOV \$GDDAT, @DAC3 ;LOAD REG
MOV @DAC3, \$BDDAT ;READ REG
CMP \$GDDAT, \$BDDAT ;COMPARE
BEQ TST14 ;;BR IF EQUAL
ERROR 5 ;ERROR, DAC #3 REGISTER NOT = 0

*TEST 14 TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777

TST14: SCOPE
MOV #7777, \$GDDAT ;LOAD EXPECTED
MOV \$GDDAT, @DAC3 ;LOAD REG
MOV @DAC3, \$BDDAT ;READ REG
CMP \$GDDAT, \$BDDAT ;COMPARE
BEQ TST15 ;;BR IF EQUAL
ERROR 5 ;ERROR, DAC #3 REGISTER NOT = 7777

*TEST 15 TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN

TST15: SCOPE
MOV #100, \$TIMES ;;DO 100 ITERATIONS
MOV #BIT11, \$GDDAT ;LOAD EXPECTED
1\$: MOV \$GDDAT, @DAC3 ;LOAD DAC #3 REGISTER
MOV @DAC3, \$BDDAT ;READ THE REGISTER
CMP \$GDDAT, \$BDDAT ;COMPARE THE DATA
BEQ 2\$;;BR IF SAME
2\$: ERROR 5 ;ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN
ASR \$GDDAT ;CHANGE THE DATA
BNE 1\$;BR AND TEST MORE DATA

```

392 (3)
393 (3)
394 (2) 003366 000004
395 003370 012777 001111 176050
396 003376 012777 002222 176044
397 003404 012777 004444 176040
398 003412 012777 007777 176034
399 003420 005077 176020
400 003424 012737 001111 001124
401 003432 017737 176010 001126
402 003440 023737 001124 001126
403 003446 001401
404 003450 104002
405 003452 012737 002222 001124 1$: MOV #2222,$GDDAT ;LOAD EXPECTED
406 003460 017737 175764 001126 MOV @DAC1,$BDDAT ;READ REG
407 003466 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
408 003474 001401 BEQ 2$ ;;BR IF EQUAL
409 003476 104003 ERROR 2 ;ERROR, SELECTED DAC0 IN ERROR
410 003500 012737 004444 001124 2$: MOV #4444,$GDDAT ;LOAD EXPECTED
411 003506 017737 175740 001126 MOV @DAC2,$BDDAT ;READ REG
412 003514 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
413 003522 001401 BEQ 3$ ;;BR IF SAME
414 003524 104004 ERROR 4 ;ERROR, SELECTED DAC #2 IN ERROR
415
416 003526 012737 007777 001124 3$: MOV #7777,$GDDAT ;LOAD EXPECTED
417 003534 017737 175714 001126 MOV @DAC3,$BDDAT ;READ REG
418 003542 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
419 003550 001401 BEQ TST17 ;;BR IF SAME
420 003552 104005 ERROR 5 ;ERROR, SELECTED DAC #3 IN ERROR

```



```

422      ;:*****
(3)      ;*TEST 17      TEST THAT RESET SETS READY BIT
(3)      ;:*****
(2)      003554 000004      †ST17: SCOPE
(1)      003556 012737 000010 001166      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
423      003564 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
424      003572 000005      RESET
425      003574 017737 175644 001126      MOV      @STAT,$BDDAT      ;READ REGISTER
426      003602 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
427      003610 001401      BEQ      TST20      ;;BR IF SET
428      003612 104001      ERROR    1      ;RESET FAILED TO SET READY
429
430      ;:*****
(3)      ;*TEST 20      TEST THAT FAST INTENSIFY CAN BE SET AND CLEARED
(3)      ;:*****
(2)      003614 000004      †ST20: SCOPE
431      003616 012777 000002 175620      MOV      #BIT1,@STAT      ;LOAD BIT 1
432      003624 012737 000202 001124      MOV      #BIT7!BIT1,$GDDAT      ;LOAD EXPECTED
433      003632 017737 175606 001126      MOV      @STAT,$BDDAT      ;READ THE REGISTER
434      003640 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMAPRE
435      003646 001401      BEQ      1$      ;;BR IF SAME
436      003650 104001      ERROR    1      ;ERROR, STATUS NOT = 202
437      003652 005077 175566      1$: CLR      @STAT      ;CLEAR STATUS REGISTER
438      003656 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
439      003664 017737 175554 001126      MOV      @STAT,$BDDAT      ;READ THE REGISTER
440      003672 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
441      003700 001401      BEQ      TST21      ;;BR IF SAME
442      003702 104001      ERROR    1      ;ERROR, STATUS NOT = 200
443
444      ;:*****
(3)      ;*TEST 21      TEST THAT MODE BIT 2 CAN BE SET AND CLEARED
(3)      ;:*****
(2)      003704 000004      †ST21: SCOPE
445      003706 012777 000004 175530      MOV      #BIT2,@STAT      ;LOAD DISPLAY STATUS
446      003714 012737 000204 001124      MOV      #BIT7!BIT2,$GDDAT      ;LOAD EXPECTED
447      003722 017737 175516 001126      MOV      @STAT,$BDDAT      ;READ REG
448      003730 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
449      003736 001401      BEQ      1$      ;;BR IF EQUAL
450      003740 104001      ERROR    1      ;ERROR, STATUS NOT = 204
451      003742 005077 175476      1$: CLR      @STAT      ;CLEAR STATUS
452      003746 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
453      003754 017737 175464 001126      MOV      @STAT,$BDDAT      ;READ REG
454      003762 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
455      003770 001401      BEQ      TST22      ;;BR IF CLEARED
456      003772 104001      ERROR    1      ;MODE FAILED TO CLEAR
457

```

459
460
(3)
(3)
(2)
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
(3)
(3)
(2)
476
477
478
479
480
481
482
483
484
485
486
487

003774 000004
003776 012777 000010 175440
004004 012737 000210 001124
004012 017737 175426 001126
004020 023737 001124 001126
004026 001401
004030 104001

004032 005077 175406
004036 012737 000200 001124
004044 017737 175374 001126
004052 023737 001124 001126
004060 001401
004062 104001

004064 000004
004066 012777 000020 175350
004074 012737 000220 001124
004102 017737 175336 001126
004110 023737 001124 001126
004116 001401
004120 104001
004122 005077 175316
004126 012737 000200 001124
004134 017737 175304 001126
004142 023737 001124 001126
004150 001401
004152 104001

*TEST 22 TEST THAT MODE BIT 3 CAN BE SET

TST22: SCOPE
MOV #BIT3,STAT ;LOAD
MOV #BIT7,STAT,SGDDAT ;LOAD EXPECTED
MOV STAT,SBDDAT ;READ REG
CMP SGDDAT,SBDDAT ;COMPARE
BEQ 1\$;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 210

1\$: CLR STAT ;CLEAR REG.
MOV #BIT7,SGDDAT ;LOAD EXPECTED
MOV STAT,SBDDAT ;READ REG.
CMP SGDDAT,SBDDAT ;COMPARE
BEQ TST23 ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 200

*TEST 23 TEST THAT EXT. DELAY (BIT 4) CAN BE SET AND CLEARED

TST23: SCOPE
MOV #BIT4,STAT ;LOAD STATUS
MOV #BIT7,STAT,SGDDAT ;LOAD EXPECTED
MOV STAT,SBDDAT ;READ REGISTER
CMP SGDDAT,SBDDAT ;COMPARE
BEQ 1\$;;BR IF SAME
ERROR 1 ;ERROR, EXT. DELAY BIT FAILED TO SET

1\$: CLR STAT ;CLEAR BIT 4
MOV #BIT7,SGDDAT ;LOAD EXPECTED
MOV STAT,SBDDAT ;READ REG.
CMP SGDDAT,SBDDAT ;COMPARE
BEQ TST24 ;;BR IF SAME
ERROR 1 ;ERROR, EXT. DELAY BIT FAILED TO CLEAR

```

489
(3)
(3)
(2) 004154 000004
490 004156 012777 000100 175260
491 004164 012737 000300 001124
492 004172 017737 175246 001126
493 004200 023737 001124 001126
494 004206 001401
495 004210 104001
496
497
(3)
(3)
(2) 004212 000004
498 004214 012777 001000 175222
499 004222 012737 001200 001124
500 004230 017737 175210 001126
501 004236 023737 001124 001126
502 004244 001401
503 004246 104001
504
505
(3)
(3)
(2) 004250 000004
506 004252 012777 002000 175164
507 004260 012737 002200 001124
508 004266 017737 175152 001126
509 004274 023737 001124 001126
510 004302 001401
511 004304 104001
512
513
(3)
(3)
(2) 004306 000004
514 004310 012777 004000 175126
515 004316 012737 004200 001124
516 004324 017737 175114 001126
517 004332 023737 001124 001126
518 004340 001401
519 004342 104001
520

```

```

*****
;*TEST 24 TEST THAT INTERRUPT ENABLE (BIT 6) CAN BE SET
*****
†ST24: SCOPE
MOV #BIT6,ASTAT ;LOAD DISPLAY STATUS
MOV #BIT7!BIT6,$GDDAT ;LOAD EXPECTED
MOV ASTAT,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST25 ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 300

*****
;*TEST 25 TEST THAT CHANNEL (BIT 9) CAN BE SET
*****
†ST25: SCOPE
MOV #BIT9,ASTAT ;LOAD DISPLAY STATUS
MOV #BIT9!BIT7,$GDDAT ;LOAD EXPECTED
MOV ASTAT,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST26 ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 1200

*****
;*TEST 26 TEST THAT STORE (BIT 10) CAN BE SET
*****
†ST26: SCOPE
MOV #BIT10,ASTAT ;LOAD DISPLAY STATUS
MOV #BIT10!BIT7,$GDDAT ;LOAD EXPECTED
MOV ASTAT,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST27 ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 2200

*****
;*TEST 27 TEST THAT WRITE THRU (BIT 11) CAN BE SET
*****
†ST27: SCOPE
MOV #BIT11,ASTAT ;LOAD DISPLAY STATUS
MOV #BIT11!BIT7,$GDDAT ;LOAD EXPECTED
MOV ASTAT,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST30 ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 4200

```

```

522 (3) *****
523 (3) *TEST 30 TEST THAT THE LOW BYTE OF THE STATUS REGISTER CAN BE CLEARED
524 (2) 004344 000004 *****
525 004346 012777 005014 175070 †TST30: SCOPE
526 004354 012737 005200 001124 MOV #BIT11!BIT9!BIT3!BIT2,@STAT ;LOAD THE STATUS REGISTER
527 004362 105077 175056 MOV #BIT11!BIT9!BIT7,$GDDAT ;LOAD THE EXPECTED RESULT
528 004366 017737 175052 001126 CLR @STAT ;CLEAR LOW STATUS BYTE
529 004374 023737 001124 001126 MOV @STAT,$BDDAT ;READ THE REGISTER
530 004402 001401 BEQ TST31 ;;BR IF EXPECTED
531 004404 104001 ERROR 1 ;FAILED TO ADDRESS ONLY THE LOW BYTE OF STATUS R

```

```

532 (3) *****
533 (3) *TEST 31 TEST THAT THE HIGH BYTE OF THE STATUS REGISTER CAN BE CLEARED
534 (2) 004406 000004 *****
535 004410 012777 005014 175026 †TST31: SCOPE
536 004416 012737 000214 001124 MOV #BIT11!BIT9!BIT3!BIT2,@STAT ;LOAD THE STATUS REGISTER
537 004424 013737 001444 010520 MOV #BIT7!BIT3!BIT2,$GDDAT ;LOAD EXPECTED RESULTS
538 004432 005237 010520 MOV STAT,CHRCOL ;MAKE THE HIGH BYTE ADDRESS
539 004436 105077 004056 INC CHRCOL ;MAKE IT ODD
540 004442 017737 174776 001126 CLR @CHRCOL ;CLEAR THE HIGH BYTE
541 004450 023737 001124 001126 MOV @STAT,$BDDAT ;RWEAD THE REGISTER
542 004456 001401 BEQ TST32 ;;BR IF EXPECTED
543 004460 104001 ERROR 1 ;FAILED TO CLEAR ONLY THE HIGH BYTE OF STATUS RE

```

```

543      ;*****
(3)      ;*TEST 32      TEST THAT WHEN INTENSIFY BIT IS SET THAT THE READY BIT CLEARS
(3)      ;*****
(2) 004462 000004
544      TST32: SCOPE
; AND THEN SETS AFTER A DELAY
545 004464 012700 001000      MOV      #1000,RO      ;LOAD EXPECTED
546 004470 005037 001124      CLR      $GDDAT      ;INTENSIFY
547 004474 012777 000001 174742      MOV      #BIT0,$STAT      ;READ REG
548 004502 017737 174736 001126      MOV      $STAT,$BDDAT      ;TEST READY
549 004510 105737 001126      TSTB    $BDDAT      ;BR IF NOT SET
550 004514 100002      BPL     1$      ;BR TO SCOPE
551 004516 104011      ERROR   11      ;READY FAILED TO CLEAR
552 004520 000414      BR      TST33
553 004522 105777 174716 1$: TSTB    $STAT      ;NEXT TEST
554 004526 100411      BMI     TST33
555 004530 005300      DEC     RO      ;DELAY
556 004532 001373      BNE     1$
557 004534 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
558 004542 017737 174676 001126      MOV      $STAT,$BDDAT      ;READ REG
559 004550 104011      ERROR   11      ;READY FAILED TO SET AFTER A DELAY
560      ; IS STORAGE SCOPE CONNECTED BUT NOT TURNED ON ??
561      ;*****
(3)      ;*TEST 33      TEST THAT MODE 1 (INTENSIFY ON DAC0) CLEARS THE READY FLAG
(3)      ;*****
(2) 004552 000004
562      TST33: SCOPE
; AND THEN SETS IT
563 004554 012700 001000      MOV      #1000,RO      ;SET UP DELAY
564 004560 012737 000204 001124      MOV      #BIT7:BIT2,$GDDAT      ;LOAD EXPECTED
565 004566 012777 000004 174650      MOV      #BIT2,$STAT      ;LOAD MODE 1
566 004574 017737 174644 001126      MOV      $STAT,$BDDAT      ;READ REG
567 004602 105737 001126      TSTB    $BDDAT      ;TEST READY
568 004606 100402      BMI     2$      ;BR IF READY STILL SET
569 004610 104011      ERROR   11      ;ERROR, IN MODE 1 READY SHOULD NOT
570      ; CLEAR UNTIL DAC0 IS LOADED
571 004612 000432      BR      TST34      ;BR TO SCOPE
572
573 004614 012737 000004 001124 2$: MOV      #BIT2,$GDDAT      ;LOAD EXPECTED
574 004622 005077 174620      CLR     $DAC0      ;ADDRESS DAC 0
575 004626 017737 174612 001126      MOV      $STAT,$BDDAT      ;READ REG
576 004634 105737 001126      TSTB    $BDDAT      ;TEST READY
577 004640 000240      NOP
578 004642 100002      BPL     1$      ;BR IF CLEAR
579 004644 104011      ERROR   11      ;ERROR, MODE 1 LOAD DAC0 FAILED TO CLEAR READY F
580 004646 000414      BR      TST34      ;BR TO SCOPE
581
582 004650 105777 174570 1$: TSTB    $STAT      ;TEST READY
583 004654 100411      BMI     TST34      ;NEXT TEST
584 004656 005300      DEC     RO      ;DELAY
585 004660 001373      BNE     1$      ;TEST READY AGAIN
586 004662 012737 000204 001124      MOV      #BIT7:BIT2,$GDDAT      ;LOAD EXPECTED
587 004670 017737 174550 001126      MOV      $STAT,$BDDAT      ;READ REG
588 004676 104011      ERROR   11      ;ERROR, READY FAILED TO SET
589      ; AFTER MODE 1 OPERATION
590

```

```

592 (3) *****
593 (3) *TEST 34 TEST THAT MODE 2 (INTENSIFY ON DAC1) CLEARS THE READY FLAG
594 (2) 004700 000004 TST34: SCOPE *****
595 004702 012700 001000 ; AND THEN SETS IT
596 004706 012737 000210 001124 MOV #1000,RO ;SET UP DELAY
597 004714 012777 000010 174522 MOV #BIT7!BIT3,$GDDAT ;LOAD EXPECTED
598 004722 017737 174516 001126 MOV #BIT3,$STAT ;LOAD MODE 2
599 004730 105737 001126 MOV $STAT,$BDDAT ;READ REG
600 004734 100402 TSTB $BDDAT ;TEST READY
601 004736 104011 BMI 2$ ;;BR IF SET
602 004740 000431 ; AND THEN SETS IT ;BR TO SCOPE
603 004742 012737 000010 001124 2$: MOV #BIT3,$GDDAT ;LOAD EXPECTED
604 004750 005077 174474 CLR @DAC1 ;ADDRESS DAC1
605 004754 017737 174464 001126 MOV $STAT,$BDDAT ;READ REG
606 004762 105737 001126 TSTB $BDDAT ;TEST READY
607 004766 100002 BPL 1$ ;;BR IF CLEARED
608 004770 104011 ERROR 11 ;ERROR, MODE 2 LOAD DAC1 FAILED TO CLEAR READY F
609 004772 000414 BR TST35 ;BR TO SCOPE
610 004774 105777 174444 1$: TSTB $STAT ;TEST READY
611 005000 100411 BMI TST35 ;NEXT TEST
612 005002 005300 DEC RO ;DELAY
613 005004 001373 BNE 1$ ;;TEST READY AGAIN
614 005006 012737 000210 001124 MOV #BIT7!BIT3,$GDDAT ;LOAD EXPECTED
615 005014 017737 174424 001126 MOV $STAT,$BDDAT ;READ REG
616 005022 104011 ERROR 11 ;ERROR, READY FAILED TO SET
617 ; AFTER MODE 2 OPERATION
618 *****
619 (3) *TEST 35 TEST WHEN ERASE IS SET, READY BIT CLEARS AND SET AFTER DELAY (SW12=1)
620 (3) *****
621 (2) 005024 000004 TST35: SCOPE *****
622 (1) 005026 012737 000001 001166 MOV #1,$TIMES ;;DO 1 ITERATION
623 005034 032777 010000 174076 BIT #BIT12,$SWR ;TEST BIT 12
624 005042 001430 BEQ TST36 ;;BYPASS IF NO STORAGE SCOPE
625 005044 012700 000010 MOV #10,RO
626 005050 005037 020416 CLR TEMP ;CLEAR DELAY
627 005054 012777 002000 174362 MOV #BIT10,$STAT ;SET STORE MODE
628 005062 052777 010000 174354 BIS #BIT12,$STAT ;SET ERASE BIT
629 005070 105777 174350 TSTB $STAT ;TEST THAT READY CLEARS
630 005074 100002 BPL 1$ ;;BR IF CLEARED
631 005076 104011 ERROR 11 ;ERROR, READY FAILED TO RESET
632 005100 000411 BR TST36 ;BR TO SCOPE
633 005102 105777 174336 1$: TSTB $STAT ;TEST FOR READY
634 005106 100406 BMI TST36 ;BR IF SET
635 005110 005337 020416 DEC TEMP ;DELAY
636 005114 001372 BNE 1$ ;;BR IF NOT READY
637 005116 005300 DEC RO ;DECREMENT COUNTER
638 005120 001370 BNE 1$ ;;BR IF NOT DONE
639 005122 104011 ERROR 11 ;ERROR, ERASE CLEARED READY AND FAILED
;SWR 12 = 1 AND NO STORAGE SCOPE CONNECTED ??

```

```

639
(3)
(3)
(2) 005124 000004
(1) 005126 012737 000040 001166
640 005134 012737 000340 177776
641 005142 012777 005204 174306
642 005150 012700 000400
643 005154 013737 020424 177776
644 005162 012777 000101 174254
645 005170 005300
646 005172 001376
647 005174 005077 174244
648 005200 104006
649 005202 000401
650 005204 022626
651 005206 013777 001460 174242
652 005214 005077 174240
653
654
(3)
(3)
(2) 005220 000004
(1) 005222 012737 000040 001166
655 005230 012737 000340 177776
656 005236 012777 005272 174212
657 005244 012700 004000
658 005250 013737 020422 177776
659 005256 012777 000101 174160
660 005264 005300
661 005266 001376
662 005270 000404
663 005272 005077 174146
664 005276 104007
665 005300 000417
666 005302 012777 005324 174146
667 005310 005037 177776
668 005314 005077 174124
669 005320 104007
670 005322 000401
671 005324 022626
672 005326 013777 001460 174122
673 005334 005077 174120

```

```

*****
;*TEST 36 TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED -1
*****

```

```

TST36: SCOPE
MOV #40,STIMES ;;DO 40 ITERATIONS
MOV #340,PSW
MOV #1$,DIV ;SET INTERRUPT VECTOR
MOV #400,RO ;SET UP DELAY
MOV BRLEV2,PSW ;LOAD CPU PSW WITH DEVICE LEVEL -1
MOV #BIT6!BIT0,STAT ;START DISPLAY
DEC RO ;DELAY
BNE -2
CLR STAT ;DO NOT LET INTERRUPT ENABLE SET
ERROR 6 ;ERROR, FAILED TO INTERRUPT
BR 2$ ;;NEXT TEST
1$: CMP (SP)+,(SP)+ ;POP THE STACK
2$: MOV IVS,IV ;RESET VECTOR
CLR DIVS

```

```

*****
;*TEST 37 TEST THAT THE DISPLAY DOES NOT INTERRUPT AT LEVEL INDICATED
*****

```

```

TST37: SCOPE
MOV #40,STIMES ;;DO 40 ITERATIONS
MOV #340,PSW
MOV #1$,DIV ;SET INTERRUPT VECTOR
MOV #4000,RO ;SET UP DELAY
MOV BRLEV1,PSW ;LOAD CPU PSW WITH DEVICE LEVEL
MOV #BIT6!BIT0,STAT ;START DISPLAY
DEC RO ;DELAY
BNE -2
BR 2$ ;;
1$: CLR STAT ;DO NOT LET INTERRUPT ENABLE SET
ERROR 7 ;ERROR INTERRUPTED IN ERROR
BR TST40 ;;NEXT TEST
2$: MOV #3$,DIV ;LOAD RETURN VECTOR
CLR PSW ;LOWER PSW
CLR STAT ;CLEAR INT ENABLE
ERROR 7 ;LOWERING THE PRIORITY FAILED TO ALLOW INTERRUPT
BR 4$ ;;NEXT TEST
3$: CMP (SP)+,(SP)+
4$: MOV IVS,IV ;RESET VECTOR
CLR DIVS

```

```

675
(3)
(3)
(2) 005340 000004
(1) 005342 012737 000040 001166
676 005350 012777 000036 174066
677 005356 012737 000200 001124
678 005364 000005
679 005366 017737 174052 001126
680 005374 023737 001124 001126
681 005402 001401
682 005404 104001
683
(3)
(3)
(2) 005406 000004
(1) 005410 012737 000040 001166
684 005416 012777 007100 174020
685 005424 012737 000200 001124
686 005432 000005
687 005434 017737 174004 001126
688 005442 023737 001124 001126
689 005450 001401
690 005452 104001
691
(3)
(3)
(2) 005454 000004
(1) 005456 012737 000040 001166
692 005464 032777 001000 173446
(3) 005472 001016
693 005474 012777 177777 173744
694 005502 005037 001124
695 005506 000005
696 005510 017737 173732 001126
697 005516 023737 001124 001126
698 005524 001401
699 005526 104002
700
(3)
(3)
(2) 005530 000004
(1) 005532 012737 000040 001166
701 005540 032777 001000 173372
(3) 005546 001016
702 005550 012777 177777 173672
703 005556 005037 001124
704 005562 000005
705 005564 017737 173660 001126
706 005572 023737 001124 001126
707 005600 001401
708 005602 104003

```

```

*****
*TEST 40 TEST THAT RESET CLEARS MODE, EXT. DELAY AND FAST INTENSIFY BITS
*****
TST40: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #BIT4!BIT3!BIT2!BIT1,$STAT
MOV #BIT7,$GDDAT ;LOAD EXPECTED
RESET
MOV $STAT,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST41 ;;BR IF EQUAL
ERROR 1 ;ERROR, RESET FAILED TO CLEAR STATUS REG
*****
*TEST 41 TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
*****
TST41: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #BIT11!BIT10!BIT9!BIT6,$STAT
MOV #BIT7,$GDDAT ;LOAD EXPECTED
RESET
MOV $STAT,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST42 ;;BR IF EQUAL
ERROR 1 ;ERROR, RESET FAILED TO CLEAR STATUS
*****
*TEST 42 TEST THAT RESET CLEARS DAC0 REGISTER (SW09=0)
*****
TST42: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
BIT #SW09,$SWR ;TEST IF BIT 9 IS SET
BNE TST43 ;;BR IF SET
MOV #-1,$DAC0
CLR $GDDAT ;LOAD EXPECTED
RESET
MOV $DAC0,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST43 ;;BR IF EQUAL
ERROR 2 ;ERROR, RESET FAILED TO CLEAR DAC 0 REGISTER
*****
*TEST 43 TEST THAT RESET CLEARS DAC1 REGISTER (SI'79=0)
*****
TST43: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
BIT #SW09,$SWR ;TEST IF BIT 9 IS SET
BNE TST44 ;;BR IF SET
MOV #-1,$DAC1
CLR $GDDAT ;LOAD EXPECTED
RESET
MOV $DAC1,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST44 ;;BR IF EQUAL
ERROR 3 ;ERROR, RESET FAILED TO CLEAR DAC1 REGISTER

```



```

710      ;:*****
(3)      ;*TEST 44      TEST THAT RESET CLEARS DAC #2 REGISTER (SW09=0)
(3)      ;:*****
(2) 005604 000004      †ST44: SCOPE
(1) 005606 012737 000040 001166      MOV #40,$TIMES      ;;DO 40 ITERATIONS
711 005614 032777 001000 173316      BIT #SW09,$SWR      ;:TEST IF BIT 9 IS SET
(3) 005622 001013      BNE TST45      ;;BR IF SET
712 005624 012777 177777 173620      MOV #-1,$DAC2      ;:LOAD THE REGISTER
713 005632 005037 001124      CLR $GDDAT      ;:CLEAR EXPECTED
714 005636 000005      RESET
715 005640 017737 173606 001126      MOV $DAC2,$BDDAT      ;:READ THE REGISTER
716 005646 001401      BEQ TST45      ;;BR IF CLEARED
717 005650 104004      ERROR 4      ;:ERROR, RESET FAILED TO CLEAR DAC #2
718      ;:*****
(3)      ;*TEST 45      TEST THAT RESET CLEARS DAC #3 REGISTER (SW09=0)
(3)      ;:*****
(2) 005652 000004      †ST45: SCOPE
(1) 005654 012737 000040 001166      MOV #40,$TIMES      ;;DO 40 ITERATIONS
719 005662 032777 001000 173250      BIT #SW09,$SWR      ;:TEST IF BIT 9 IS SET
(3) 005670 001013      BNE TST46      ;;BR IF SET
720 005672 012777 177777 173554      MOV #-1,$DAC3      ;:LOAD THE REGISTER
721 005700 005037 001124      CLR $GDDAT      ;:CLEAR THE EXPECTED
722 005704 000005      RESET
723 005706 017737 173542 001126      MOV $DAC3,$BDDAT      ;:READ THE REGISTER
724 005714 001401      BEQ TST46      ;;BR IF CLEARED
725 005716 104005      ERROR 5      ;:ERROR, RESET FAILED TO CLEAR DAC #3
738      ;:*****
(4)      ;*TEST 46      TEST THAT RESET SET DAC0 TO 4000 (SW09=1)
(4)      ;:*****
(3) 005720 000004      †ST46: SCOPE
(2) 005722 012737 000040 001166      MOV #40,$TIMES      ;;DO 40 ITERATIONS
(1) 005730 032777 001000 173202      BIT #SW09,$SWR      ;:TEST BIT 9
(3) 005736 001417      BEQ TST47      ;;BR IF CLEARED
(1) 005740 012777 003777 173500      MOV #3777,$DAC0      ;:LOAD DAC0
(1) 005746 012737 004000 001124      MOV #4000,$GDDAT      ;:LOAD EXPTECTED
(1) 005754 000005      RESET
(1) 005756 017737 173464 001126      MOV $DAC0,$BDDAT      ;:READ THE DAC0 REGISTER
(1) 005764 023737 001124 001126      CMP $GDDAT,$BDDAT      ;:COMPARE
(3) 005772 001401      BEQ TST47      ;;BR IF SAME
(1) 005774 104002      ERROR 2      ;:RESET FAILED TO SET DAC0 TO 4000
739      ;:*****
(4)      ;*TEST 47      TEST THAT RESET SET DAC1 TO 4000 (SW09=1)
(4)      ;:*****
(3) 005776 000004      †ST47: SCOPE
(2) 006000 012737 000040 001166      MOV #40,$TIMES      ;;DO 40 ITERATIONS
(1) 006006 032777 001000 173124      BIT #SW09,$SWR      ;:TEST BIT 9
(3) 006014 001417      BEQ TST50      ;;BR IF CLEARED
(1) 006016 012777 003777 173424      MOV #3777,$DAC1      ;:LOAD DAC1
(1) 006024 012737 004000 001124      MOV #4000,$GDDAT      ;:LOAD EXPTECTED
(1) 006032 000005      RESET
(1) 006034 017737 173410 001126      MOV $DAC1,$BDDAT      ;:READ THE DAC1 REGISTER
(1) 006042 023737 001124 001126      CMP $GDDAT,$BDDAT      ;:COMPARE
(3) 006050 001401      BEQ TST50      ;;BR IF SAME
(1) 006052 104003      ERROR 3      ;:RESET FAILED TO SET DAC1 TO 4000

```

740
(4)
(4)
(3)
(2)
(1)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(3)
(1)
741
(4)
(4)
(3)
(2)
(1)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(3)
(1)
742
(3)
(3)
(2)
(1)
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757

006054 000004
006056 012737 000040 001166
006064 032777 001000 173046
006072 001417
006074 012777 003777 173350
006102 012737 004000 001124
006110 000005
006112 017737 173334 001126
006120 023737 001124 001126
006126 001401
006130 104004
006132 000004
006134 012737 000040 001166
006142 032777 001000 172770
006150 001417
006152 012777 003777 173274
006160 012737 004000 001124
006166 000005
006170 017737 173260 001126
006176 023737 001124 001126
006204 001401
006206 104005
006210 000004
006212 012737 000010 001166
006220 032777 002000 172712
006226 001024
006230 012777 000020 173206
006236 105277 173202
006242 012700 000000
006246 105777 173172
006252 100403
006254 005300
006256 001373
006260 000407
006262 012737 000020 001124
006270 017737 173150 001126
006276 104011

```
*****  
*TEST 50 TEST THAT RESET SET DAC2 TO 4000 (SW09=1)  
*****  
TST50: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
BIT #SW09,$SWR ;TEST BIT 9  
BEQ TST51 ;;BR IF CLEARED  
MOV #3777,$DAC2 ;LOAD DAC2  
MOV #4000,$GDDAT ;LOAD EXPTECTED  
RESET  
MOV $DAC2,$BDDAT ;READ THE DAC2 REGISTER  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST51 ;;BR IF SAME  
ERROR 4 ;RESET FAILED TO SET DAC2 TO 4000  
*****  
*TEST 51 TEST THAT RESET SET DAC3 TO 4000 (SW09=1)  
*****  
TST51: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
BIT #SW09,$SWR ;TEST BIT 9  
BEQ TST52 ;;BR IF CLEARED  
MOV #3777,$DAC3 ;LOAD DAC3  
MOV #4000,$GDDAT ;LOAD EXPTECTED  
RESET  
MOV $DAC3,$BDDAT ;READ THE DAC3 REGISTER  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST52 ;;BR IF SAME  
ERROR 5 ;RESET FAILED TO SET DAC3 TO 4000  
*****  
*TEST 52 TEST THAT EXTERNAL DELAY DOES NOT SET DISPLAY READY SW10=0  
*****  
TST52: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
BIT #SW10,$SWR ;TEST IF SW 10 IS SET  
BNE TST53 ;;BR IF EXT. DELAY SWITCH IS SET  
MOV #BIT4,$STAT ;LOAD EXT. DELAY ENABLE  
INCB $STAT ;START DISPLAY  
MOV #0,R0 ;LOAD DELAY  
TSTB $STAT ;WAIT FOR READY  
BMI 2$ ;BR IF SET  
DEC R0 ;DELAY  
BNE 1$ ;BR IF NOT FINISHED  
BR TST53 ;;  
MOV #BIT4,$GDDAT ;LOAD EXPECTED  
MOV $STAT,$BDDAT ;READ ACUTAL  
ERROR 11 ;EXT. DELAY CAUSED READY TO SET  
;WHEN THE OPERATOR (VIA SW10) SAID NO EXT. DELAY WAS CON
```

```

759
(3)
(3)
(2) 006300 000004
(1) 006302 012737 000010 001166
760 006310 032777 002000 172622
761 006316 001424
762 006320 012777 000020 173116
763 006325 012700 000000
764 006332 105277 173106
765 006336 105777 173102
766 006342 100411
767 006344 005300
768 006346 001373
769 006350 012737 000220 001124
770 006356 017737 173062 001126
771 006364 104011
772
773 006366 000005
774
775
(3)
(3)
(2) 006370 000004
(1) 006372 012737 000001 001166
776 006400 000005
777 006402 005237 001206
778 006406 123737 001206 020426
779 006414 001431
780 006416 063737 001436 001444
781 006424 063737 001436 001446
782 006432 063737 001436 001450
783 006440 063737 001436 001452
784 006446 063737 001436 001454
785 006454 063737 001440 001456
786 006462 063737 001440 001460
787 006470 005037 001102
788 006474 000137 002466

```

```

*****
*TEST 53 TEST THE EXTERNAL DELAY DOES SET DISPLAY READY (SW10=1)
*****
TST53: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
BIT #SW10,$SWR ;TEST SR BIT 10
BEQ TST54 ;;BR IF CLEARED
MOV #BIT4,$STAT ;ENABLE EXT. DELAY
MOV #0,RO ;LOAD COUNTER
INCB $STAT ;ENABLE DISPLAY
TSTB $STAT ;WAIT FOR READY
BMI 2$ ;BR IF DONE
DEC RO ;DELAY
BNE 1$ ;BR IF NOT DONE
MOV #BIT7:BIT4,$GDDAT ;LOAD EXPECTED VALUE
MOV $STAT,$BDDAT ;READ ACTUAL
ERROR 11 ;EXT. DELAY FAILED TO SET READY
;WHEN THE OPERATOR (VIA SW 10=1) SAID AN EXT. DELAY WAS
;ENSURE CLEARED AA11-K
2$: RESET

*****
*TEST 54 DETERMINE IF MORE AA11-K'S REMAIN TO BE TESTED
*****
TST54: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
RESET ;MORE UNITS?
INC $UNIT
CMPB $UNIT,EVER
BEQ $EOP ;;BR IF NONE
ADD VADDR,STAT ;UPDATE ADDRESS
ADD VADDR,DAC0
ADD VADDR,DAC1
ADD VADDR,DAC2
ADD VADDR,DAC3
ADD VVECT,IV ;UPDATE VECTOR
ADD VVECT,IVS
CLR $STNM
JMP LTEST ;TEST ANOTHER AA11-K

```



```

805 006622 012706 00110C VISUAL: MOV #STACK, SP ;LOAD SP
806 006626 005737 020430 TST EVER1 ;TEST IF ON AUTO-HARDWARE TESTER
807 006632 100003 BPL 1$ ;BR IF NOT
808 006634 004537 014062 JSR RS, CROSS ;CHANGE THE RELAYS
809 006640 120000 BIT15!BIT13
810 006642 105737 020430 1$: TSTB EVER1 ;TEST IF TYPED ONCE
811 006646 001006 BNE PICO ;BR IF YES
812 006650 104401 TYPE
813 006652 015403 MES6 ;HEADER ABOUT SCOPE ADJ.
814 006654 104401 TYPE
815 006656 015433 MES15 ;TEXT ABOUT SWR
816 006660 105237 020430 INCB ;SET FLAG
820 006664 013700 001446 PICO: MOV DAC0, R0
821 006670 013701 001450 MOV DAC1, R1
822 006674 032777 000010 172236 BIT #SW03, @SWR ;TEST SR BIT 3
823 006702 001404 BEQ 2$ ;BR IF DAC #0 AND #1
824 006704 013700 001452 MOV DAC2, R0 ;USE DAC #2 AND 3
825 006710 013701 001454 MOV DAC3, R1
826 006714 012737 000030 020414 2$: MOV #30, TICKS ;LOAD TIMER FOR CP TYPE
827 006722 004737 014240 JSR PC, CHTIME ;CHANGE TIMER FOR CP TYPE
828 006726 004737 007016 1$: JSR PC, PBB
829 006732 004737 014124 JSR PC, TIMER ;DONE ?
830 006736 000773 BR 1$
834 006740 013700 001450 PIC1: MOV DAC1, R0
835 006744 013701 001446 MOV DAC0, R1
836 006750 032777 000010 172162 BIT #SW03, @SWR ;TEST SR BIT 3
837 006756 001404 BEQ 2$ ;BR IF DAC #0 AND 1
838 006760 013700 001454 MOV DAC3, R0 ;USE DAC #2 AND 3
839 006764 013701 001452 MOV DAC2, R1
840 006770 012737 000030 020414 2$: MOV #30, TICKS ;LOAD TIME
841 006776 004737 014240 JSR PC, CHTIME ;CHANGE TIMER FOR CP TYPE
842 007002 004737 007016 1$: JSR PC, PBB
843 007006 004737 014124 JSR PC, TIMER ;DONE ?
844 007012 000773 BR 1$
845 007014 000440 BR PIC3
846 007016 005077 172422 PBB: CLR @STAT
847 007022 032777 002000 172110 BIT #SW10, @SWR ;TEST IS SW10 =1
848 007030 001403 BEQ 10$ ;BR IF NOT
849 007032 052777 000020 172404 BIS #BIT4, @STAT ;ENABLE EXT. DELAY
850 007040 013704 001444 10$: MOV STAT, R4
851 007044 012703 007777 MOV #7777, R3 ;SET HIGH LIMIT
852 007050 012702 000001 MOV #1, R2 ;INITIALIZE INCREMENTS BETWEEN POINTS
853 007054 032777 001000 172056 BIT #SW09, @SWR ;TEST SW 9 = 1
854 007062 001012 BNE 2$ ;BR IF YES
855 007064 012711 004000 MOV #4000, (R1)
856 007070 005010 CLR (R0)
857 007072 060210 1$: ADD R2, (R0) ;INCREMENT
858 007074 005214 INC (R4) ;INTENSIFY
859 007076 105714 TSTB (R4)
860 007100 100376 BPL -2
861 007102 021003 CMP (R0), R3 ;DONE ALL POINTS?
862 007104 001372 BNE 1$ ;NO
863 007106 000207 RTS PC
864 007110 005011 2$: CLR (R1) ;LOAD TWO'S COMP. ORIGIN

```

MAINDEC-11-DZAAC-B AA11-K DIAGNOSTIC
DZAACB.P11 DISPLAY A VERTICAL LINE

G04
MACY11 27(663) 19-DEC-76 08:39 PAGE 24-1

SEQ 0045

865 007112 005010
966 007114 000766

CLR (RD)
BR 15

```

      .SBTTL PINCUSHION TEST (DISPLAY SQUARE)
868
869
870 007116 005077 172322 PIC3: CLR  QSTAT
871 007122 012737 000140 020414 MOV  #140,TICKS
872 007130 004737 014240 JSR  PC,CHTIME
873 007134 013701 001446 MOV  DAC0,R1
874 007140 013702 001450 MOV  DAC1,R2
875 007144 032777 000010 171766 BIT  #SW03,QSWR ;TEST SR BIT 3
876 007152 001404 BEQ  1$ ;BR IF DAC #0 AND 1
877 007154 013701 001452 MOV  DAC2,R1 ;USE DAC #2 AND 3
878 007160 013702 001454 MOV  DAC3,R2
879 007164 013703 001444 1$: MOV  STAT,R3
880 007170 012704 000020 MOV  #20,R4
881 007174 032777 001000 171736 P3: BIT  #SW09,QSWR ;TEST BIT 9
882 007202 001405 BEQ  1$
883 007204 012711 004000 MOV  #4000,(R1) ;LOAD TWO'S COMP ORIGIN
884 007210 012712 004000 MOV  #4000,(R2)
885 007214 000402 BR   2$
886 007216 005011 1$: CLR  (R1) ;CLEAR AXIS
887 007220 005012 CLR  (R2)
888 ;DRAW BOTTOM LINE
889 007222 012700 000377 2$: MOV  #377,R0
890 007226 032777 002000 171704 BIT  #SW10,QSWR ;TEST IF SW10 =1
891 007234 001403 BEQ  P3A ;BR IF NOT
892 007236 052777 000020 172200 BIS  #BIT4,QSTAT ;ENABLE EXT. DELAY
893 007244 060411 P3A: ADD  R4,(1)
894 007246 005213 INC  (R3) ;INTENSIFY
895 007250 105713 TSTB (3)
896 007252 100376 BPL  #-2 ;WAIT FOR READY
897 007254 005300 DEC  R0
898 007256 001372 BNE  P3A ;NO
899 ;DRAW RIGHT LINE
900 007260 012700 000377 MOV  #377,R0
901 007264 060412 P3B: ADD  R4,(2)
902 007266 005213 INC  (R3) ;INTENSIFY
903 007270 105713 TSTB (3)
904 007272 100376 BPL  #-2 ;WAIT FOR READY
905 007274 005300 DEC  R0
906 007276 001372 BNE  P3B ;NO
907 ;DRAW TOP LINE
908 007300 012700 000377 MOV  #377,R0
909 007304 160411 P3C: SUB  R4,(1)
910 007306 005213 INC  (R3) ;INTENSIFY
911 007310 105713 TSTB (3)
912 007312 100376 BPL  #-2 ;WAIT FOR READY
913 007314 005300 DEC  R0
914 007316 001372 BNE  P3C ;NO
915 ;DRAW LEFT LINE
916 007320 012700 000377 MOV  #377,R0
917 007324 160412 P3D: SUB  R4,(2)
918 007326 005213 INC  (R3) ;INTENSIFY
919 007330 105713 TSTB (3)
920 007332 100376 BPL  #-2 ;WAIT FOR READY
921 007334 005300 DEC  R0

```

```

922 007336 001372          BNE      P3D          ;NO
923 007340 004737 014124  JSR      PC,TIMER
924 007344 000713          BR       P3
925          .SBTTL     PLOT AN X
926
927 007346 005077 172072  PIC4:   CLR      @STAT
928 007352 012737 000100 020414  MOV      #100,TICKS
929 007360 004737 014240          JSR      PC,CHTIME   ;CHANGE TIMER FOR CP TYPE
930 007364 013701 001446  PIC4B:  MOV      DAC0,R1
931 007370 013702 001450          MOV      DAC1,R2
932 007374 032777 000010 171536  BIT      #SW03,@SWR   ;TEST SWR BIT 3
933 007402 001404          BEQ      1$          ;BR IF DAC #0 AND 1
934 007404 013701 001452          MOV      DAC2,R1     ;USE DAC #2 AND 3
935 007410 013702 001454          MOV      DAC3,R2
936 007414 013703 001444          MOV      STAT,R3
937 007420 012704 000004          MOV      #4,R4
938 007424 032777 001000 171506  P4:     BIT      #SW09,@SWR   ;TEST BIT 9
939 007432 001405          BEQ      1$
940 007434 012712 004000          MOV      #4000,(R2)
941 007440 012711 004000          MOV      #4000,(R1)
942 007444 000402          BR       2$
943 007446 005012          1$:    CLR      (R2)
944 007450 005011          CLR      (R1)
945
946          ;PLOT LINE BEGINNING IN LOWER LEFT CORNER
947 007452 012700 001777          2$:    MOV      #1777,R0
948 007456 032777 002000 171454  BIT      #SW10,@SWR   ;TEST IF SW10 =1
949 007464 001403          BEQ      P4A        ;BR IF NOT
950 007466 052777 000020 171750  P4A:   BIS      #BIT4,@STAT ;ENABLE EXT. DELAY
951 007474 005213          INC      (R3)       ;INTENSIFY
952 007476 105713          TSTB     (3)
953 007500 100376          BPL      -2
954 007502 060412          ADD      R4,(2)     ;+4 TO Y
955 007504 060411          ADD      R4,(1)     ;+4 TO X
956 007506 005300          DEC      R0
957 007510 001371          BNE      P4A        ;NO
958 007512 105713          TSTB     (3)
959 007514 100376          BPL      -2
960          ;PLOT LINE BEGINNING IN UPPER LEFT CORNER
961 007516 012712 007774          MOV      #7774,(2)
962 007522 005011          CLR      (1)
963 007524 012700 001777          P4B:   MOV      #1777,R0
964 007530 005213          INC      (R3)       ;INTENSIFY
965 007532 105713          TSTB     (3)
966 007534 100376          BPL      -2
967 007536 160412          SUB      R4,(2)     ; -4 TO Y
968 007540 060411          ADD      R4,(1)     ; +4 TO X
969 007542 005300          DEC      R0
970 007544 001371          BNE      P4B        ;NO
971 007546 004737 014124  JSR      PC,TIMER
972 007552 000724          BR       P4

```



```

974
975
976
977 007554 012737 000060 020414 PIC5: MOV #60,TICKS ;LOAD TIMER
978 007562 004737 014240 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
979
980 007566 005077 171652 1$: CLR @STAT ;CLEAR STATUS
981 007572 005077 171650 CLR @DAC0 ;CLEAR X AXIS
982 007576 005077 171646 CLR @DAC1 ;CLEAR Y AXIS
983 007602 005077 171644 CLR @DAC2
984 007606 005077 171642 CLR @DAC3
985 007612 032777 002000 171320 BIT #SW10,@SWR ;TEST SW BIT 10
986 007620 001403 BEQ 10$ ;BR IF NOT
987 007622 052777 000020 171614 BIS #BIT4,@STAT ;ENABLE EXT. DELAY
988 007630 032777 000010 171302 10$: BIT #SW03,@SWR ;TEST SR BIT 3
989 007636 001405 BEQ 2$ ;BR IF DAC #0 AND 1
990 007640 013700 001454 MOV DAC3,R0 ;LOAD Y DAC
991 007644 013701 001452 MOV DAC2,R1 ;LOAD X DAC
992 007650 000404 BR 3$
993 007652 013700 001450 2$: MOV DAC1,R0 ;LOAD Y DAC
994 007656 013701 001446 MOV DAC0,R1 ;LOAD X DAC
995 007662 032777 000200 171250 3$: BIT #SW07,@SWR ;TEST X OR Y TEST BIT
996 007670 001403 BEQ 4$ ;NORMAL
997 007672 010003 MOV R0,R3 ;REVERSE THE AXIS
998 007674 010100 MOV R1,R0
999 007676 010301 MOV R3,R1
1000 007700 013703 001444 4$: MOV STAT,R3 ;LOAD STATUS REG
1001 007704 032777 001000 171226 5$: BIT #SW09,@SWR ;TEST TWO'S COMP SWITCH
1002 007712 001007 BNE 6$
1003 007714 004537 007762 JSR R5,LODPNT ;LOAD A LINE
1004 007720 000000 0 JSR R5,LODPNT ;LOAD A LINE
1005 007722 004537 007762 JSR R5,LODPNT ;LOAD A LINE
1006 007726 007777 7777 BR 7$
1007 007730 000406 BR 7$
1008 007732 004537 007762 6$: JSR R5,LODPNT ;LOAD A LINE
1009 007736 004000 4000 JSR R5,LODPNT
1010 007740 004537 007762 JSR R5,LODPNT
1011 007744 003777 3777 TST (R0) ;END
1012 007746 005710 7$: BNE 5$ ;BR IF NOT
1013 007750 001355 JSR PC,TIMER ;TEST TIME
1014 007752 004737 014124 BR 1$
1015 007756 000703 BR 1$
1016 007760 000414 BR PIC6
1017
1018 007762 012702 000200 LODPNT: MOV #200,R2
1019 007766 011511 1$: MOV (R5),(R1) ;LOAD AXIS
1020 007770 005213 INC (R3) ;INTENSIFY
1021 007772 105713 2$: TSTB (R3) ;DONE
1022 007774 100376 BPL 2$ ;WAIT
1023 007776 062710 000002 ADD #2,(R0) ;UPDATE AXIS
1024 010002 005302 DEC R2 ;DONE
1025 010004 001370 BNE 1$ ;BR IF NOT
1026 010006 005725 TST (R5)+ ;UPDATE
1027 010010 000205 RTS R5 ;EXIT

```

```

1040 .SBTTL PLOT CHARACTER SET
1041
1042 010012 012737 000100 020414 PIC6: MOV #100,TICKS
1043 010020 004737 014240 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
1044 010024 013737 001446 010510 MOV DAC0,VCXTMP ;LOAD THE X AXIS
(1) 010032 013737 001450 010512 MOV DAC1,VCYTMP ;LOAD THE Y AXIS
(1) 010040 032777 000010 171072 BIT #SW03,@SWR ;TEST SR BIT 3
(1) 010046 001406 BEQ 1$ ;BR IF DAC #0 AND 1
(1) 010050 013737 001452 010510 MOV DAC2,VCXTMP ;LOAD THE X AXIS
(1) 010056 013737 001454 010512 MOV DAC3,VCYTMP
(1) 010064 000240 1$: NOP
1045 010066 012737 001400 010516 PIC6A: MOV #1400,XPOS
1046 010074 012737 005400 010514 MOV #5400,YPOS
1047 010102 032777 001000 171030 BIT #SW09,@SWR ;TEST SW 9
1048 010110 001406 BEQ 1$
1049 010112 012737 005400 010516 MOV #5400,XPOS ;LOAD 2'S COMP ORIGIN
1050 010120 012737 001400 010514 MOV #1400,YPOS
1051 010126 012702 010524 1$: MOV #A,R2 ;LOAD STARTING CHARACTER
1052 010132 005077 171306 CLR @STAT
1053 010136 004737 010304 JSR PC,PIC6B
1054 010142 012737 001400 010516 MOV #1400,XPOS ;LOAD X POS
1055 010150 012737 004000 010514 MOV #4000,YPOS ;LOAD Y POS
1056 010156 032777 001000 170754 BIT #SW09,@SWR ;TEST SW 9
1057 010164 001406 BEQ 2$ ;BR IF 0
1058 010166 012737 005400 010516 MOV #5400,XPOS ;LOAD 2'S COMP ORIGIN
1059 010174 012737 000000 010514 MOV #0,YPOS
1060 010202 012702 010625 2$: MOV #N,R2 ;LOAD STARTING CHARACTER
1061 010206 005077 171232 CLR @STAT
1062 010212 004737 010304 JSR PC,PIC6B ;DISPLAY CHARACTERS
1063 010216 012737 001400 010516 MOV #1400,XPOS ;LOAD X POS
1064 010224 012737 002400 010514 MOV #2400,YPOS ;LOAD Y POS
1065 010232 032777 001000 170700 BIT #SW09,@SWR ;TEST SW 9
1066 010240 001406 BEQ 3$
1067 010242 012737 005400 010516 MOV #5400,XPOS
1068 010250 012737 006400 010514 MOV #6400,YPOS ;LOAD 2'S COMP ORIGIN
1069 010256 012702 010726 3$: MOV #NO,R2 ;LOAD STARTING CHARACTER
1070 010262 005077 171156 CLR @STAT
1071 010266 004737 010304 JSR PC,PIC6B ;DISPLAY CHARACTERS
1072 010272 004737 014124 JSR PC,TIMER
1073 010276 000673 BR PIC6A
1074 010300 000137 011030 JMP PIC7
1075 010304 012737 177763 010520 PIC6B: MOV #-13,CHRCOL ;CHARACTERS PER ROW
1076 010312 013705 001444 MOV STAT,R5
1077 010316 004737 010332 GENI: JSR PC,CHAR
1078 010322 005237 010520 INC CHRCOL ;
1079 010326 001373 BNE GENI
1080 010330 000207 RTS
1081

```

```

1083
1084
1085 010332 013737 010514 010522 :PLOT CHARACTER
1086 010340 042715 000016 CHAR: MOV YPOS, YPT
1087 010344 013777 010516 000136 BIC #16, (5)
1088 010352 013777 010514 000132 MOV XPOS, @VCXTMP
1089 010360 032777 002000 170552 MOV YPOS, @VCYTMP
1090 010366 001403 BEQ CHAR4 ;TEST SR BIT 10
1091 010370 052777 000020 171046 BIS #BIT4, @STAT ;BR IF NOT
1092 010376 105777 171042 CHAR4: TSTB @STAT ;ENABLE EXT. DELAY
1093 010402 100375 BPL CHAR4
1094 010404 052715 000002 BIS #BIT1, (5) ;ENABLE FAST INTENSIFY ON LOADING Y
1095 010410 012704 000040 MOV #40, R4
1096 010414 012700 177773 MOV #-5, R0 ;INITIALIZE COLUMN COUNT
1097 010420 012701 177771 CHAR1: MOV #-7, R1 ;INITIALIZE ROW COUNT
1098 010424 112203 MOVB (2)+, R3 ;PUT CHARACTER POINTS IN R3
1099 010426 106103 CHAR2: ROLB R3
1100 010430 100007 BPL CHAR3 ;NO
1101 010432 013777 010516 000050 MOV XPOS, @VCXTMP
1102 010440 013777 010514 000044 MOV YPOS, @VCYTMP
1103 010446 005215 INC (R5) ;INTENSIFY
1104 010450 105715 CHAR3: TSTB (R5)
1105 010452 100376 BPL CHAR3
1106 010454 060437 010514 ADD R4, YPOS
1107 010460 005201 INC R1 ;+1 TO ROW
1108 010462 001361 BNE CHAR2 ;FINISH ROW
1109 010464 013737 010522 010514 MOV YPT, YPOS ;REINITIALIZE ROW FOR NEXT COLUMN
1110 010472 060437 010516 ADD R4, XPOS
1111 010476 005200 INC R0 ;+1 TO COLUMN COUNT
1112 010500 001347 BNE CHAR1
1113 010502 060437 010516 ADD R4, XPOS
1114 010506 000207 RTS PC ;EXIT
1115
1116 010510 000000 VCXTMP: 0
1117 010512 000000 VCYTMP: 0
1118 010514 000000 YPOS: 0 ;CONTAINS Y POSITION AT ANY GIVEN TIME
1119 010516 000000 XPOS: 0 ;CONTAINS X POSITION AT ANY GIVEN TIME
1120 010520 000000 CHRCOL: 0
1121 010522 000000 YPT: 0
1122

```

1124						
1125	010524	176	021	021	A:	.BYTE 176,21,21,21,176
	010527	021	176			
1126	010531	177	111	111	B:	.BYTE 177,111,111,111,66
	010534	111	066			
1127	010536	076	101	101	C:	.BYTE 76,101,101,101,42
	010541	101	042			
1128	010543	177	101	101	D:	.BYTE 177,101,101,101,76
	010546	101	076			
1129	010550	177	111	111	E:	.BYTE 177,111,111,111,101
	010553	111	101			
1130	010555	177	011	011	F:	.BYTE 177,11,11,11,1
	010560	011	001			
1131	010562	076	101	121	G:	.BYTE 76,101,121,121,62
	010565	121	062			
1132	010567	177	010	010	H:	.BYTE 177,10,10,10,177
	010572	010	177			
1133	010574	000	101	177	I:	.BYTE 0,101,177,101,0
	010577	101	000			
1134	010601	060	100	100	J:	.BYTE 60,100,100,100,77
	010604	100	077			
1135	010606	177	010	024	K:	.BYTE 177,10,24,42,101
	010611	042	101			
1136	010613	177	100	100	L:	.BYTE 177,100,100,100,100
	010616	100	100			
1137	010620	177	004	010	M:	.BYTE 177,4,10,4,177
	010623	004	177			
1138	010625	177	004	010	N:	.BYTE 177,4,10,20,177
	010630	020	177			
1139	010632	076	101	101	O:	.BYTE 76,101,101,101,76
	010635	101	076			
1140	010637	177	011	011	P:	.BYTE 177,11,11,11,6
	010642	011	006			
1141	010644	076	101	121	Q:	.BYTE 76,101,121,141,176
	010647	141	176			
1142	010651	177	011	031	R:	.BYTE 177,11,31,51,106
	010654	051	106			
1143	010656	046	111	111	S:	.BYTE 46,111,111,111,62
	010661	111	062			
1144	010663	001	001	177	T:	.BYTE 1,1,177,1,1
	010666	001	001			
1145	010670	077	100	100	U:	.BYTE 77,100,100,100,77
	010673	100	077			
1146	010675	037	040	100	V:	.BYTE 37,40,100,40,37
	010700	040	037			
1147	010702	177	020	010	W:	.BYTE 177,20,10,20,177
	010705	020	177			
1148	010707	143	024	010	X:	.BYTE 143,24,10,24,143
	010712	024	143			
1149	010714	003	004	170	Y:	.BYTE 3,4,170,4,3
	010717	004	003			
1150	010721	141	121	111	Z:	.BYTE 141,121,111,105,103
	010724	105	103			
1151	010726	076	121	111	NO:	.BYTE 76,121,111,105,76

1152	010731	105	076
	010733	000	102
	010736	100	000
1153	010740	142	121
	010743	105	102
1154	010745	042	101
	010750	111	066
1155	010752	030	024
	010755	177	020
1156	010757	047	105
	010762	105	071
1157	010764	076	111
	010767	111	062
1158	010771	101	041
	010774	011	007
1159	010776	066	111
	011001	111	066
1160	011003	046	111
	011006	111	076
1161	011010	000	000
	011013	000	000
1162	011015	000	000
	011020	000	000
1163	011022	000	000
	011025	000	000

177	N1:	.BYTE 0,102,177,100,0
111	N2:	.BYTE 142,121,111,105,102
111	N3:	.BYTE 42,101,111,111,66
022	N4:	.BYTE 30,24,22,177,20
105	N5:	.BYTE 47,105,105,105,71
111	N6:	.BYTE 76,111,111,111,62
021	N7:	.BYTE 101,41,21,11,7
111	N8:	.BYTE 66,111,111,111,66
111	N9:	.BYTE 46,111,111,111,76
000	SPACEA:	.BYTE 0,0,0,0,0
000		.BYTE 0,0,0,0,0
000		.BYTE 0,0,0,0,0

1164
1165
1166

011030

.EVEN
.SBTTL CHANNEL 1 AND CHANNEL 2 TEST

1168	011030	012737	000200	020414	PIC7:	MOV	#200,TICKS	;SET UP A TIMER
1169	011036	004737	014240			JSR	PC,CHTIME	;CHANGE TIMER FOR CP TYPE
1170	011042	013737	001446	010510		MOV	DAC0,VCXTMP	;LOAD THE X AXIS
(1)	011050	013737	001450	010512		MOV	DAC1,VCYTMP	;LOAD THE Y AXIS
(1)	011056	032777	000010	170054		BIT	#SW03,ASWR	;TEST SR BIT 3
(1)	011064	001406				BEQ	1\$;BR IF DAC #0 AND 1
(1)	011066	013737	001452	010510		MOV	DAC2,VCXTMP	;LOAD THE X AXIS
(1)	011074	013737	001454	010512		MOV	DAC3,VCYTMP	
(1)	011102	000240			1\$:	NOP		
1171	011104	013705	001444			MOV	STAT,R5	;SET UP R5 FOR STATUS REGISTER POINTER
1172	011110	012777	000000	170326	PIC7AA:	MOV	#0,ASAT	;SET UP SCOPE CONTROL
1173	011116	012737	002000	010516		MOV	#2000,XPOS	;LOAD X POSITION
1174	011124	012737	005000	010514		MOV	#5000,YPOS	;LOAD Y POSITION
1175	011132	032777	001000	170000		BIT	#SW09,ASWR	;TEST BIT 9
1176	011140	001406				BEQ	1\$	
1177	011142	012737	006000	010516		MOV	#6000,XPOS	;LOAD 2'S COMP ORIGIN
1178	011150	012737	001000	010514		MOV	#1000,YPOS	
1179	011156	012737	000011	020434	1\$:	MOV	#9,P7CNT	;SAVE THE NUMBER OF CHARACTERS
1180	011164	012737	020440	020436		MOV	#CH01,P7PNT	;SAVE CHANNEL 1 POINTER
1181	011172	017702	007240		PIC7A:	MOV	AP7PNT,R2	;MOVE MESSAGE POINTER INTO R2 FOR DISPLAY ROUTINE
1182	011176	004737	010332			JSR	PC,CHAR	;DISPLAY A CHARACTER
1183	011202	062737	000002	020436		ADD	#2,P7PNT	;ADD 2 TO THE MESSAGE POINTER
1184	011210	005337	020434			DEC	P7CNT	;DECREMENT CHARACTER COUNT
1185	011214	001366				BNE	PIC7A	;NOT FINISHED WITH ALL CHARACTERS
1186	011216	012777	001000	170220		MOV	#1000,ASAT	
1187	011224	012737	002000	010516		MOV	#2000,XPOS	;SET UP X POS FOR CHANNEL 2
1188	011232	012737	003000	010514		MOV	#3000,YPOS	;SET UP Y
1189	011240	032777	001000	167672		BIT	#SW09,ASWR	;TEST BIT 9
1190	011246	001406				BEQ	1\$	
1191	011250	012737	006000	010516		MOV	#6000,XPOS	;LOAD 2'S COMP ORIGIN
1192	011256	012737	007000	010514		MOV	#7000,YPOS	
1193	011264	012737	000011	020434	1\$:	MOV	#9,P7CNT	;SET UP CHARACTER COUNT
1194	011272	012737	020462	020436		MOV	#CH02,P7PNT	;SET UP CHANNEL 2 POINTER
1195	011300	017702	007132		PIC7B:	MOV	AP7PNT,R2	;SET UP
1196	011304	004737	010332			JSR	PC,CHAR	;DISPLAY A CHARACTER
1197	011310	062737	000002	020436		ADD	#2,P7PNT	;ADD 2 TO THE POINTER
1198	011316	005337	020434			DEC	P7CNT	;DECREMENT COUNT
1199	011322	001366				BNE	PIC7B	;NOT FINISHED
1200	011324	004737	014124			JSR	PC,TIMER	;CHECK THE RUNTIME OF THIS ROUTINE
1201	011330	000667				BR	PIC7AA	;NOT FINISHED
1202	011332	000400				BR	PIC12	

```

1204                                     .SBTTL PHOSPHOR TEST
1205
1206 011334 012737 000002 020414 PIC12: MOV #2,TICKS
1207 011342 004737 013524 1$: JSR PC,CLRVC ;ERASE THE SCREEN
1208 011346 004737 013642 JSR PC,LOADVC ;LOAD THE SCREEN
1209 011352 004737 014124 JSR PC,TIMER ;CHECK THE TIME
1210 011356 000771 BR 1$
1211 011360 004737 013524 JSR PC,CLRVC
1212 011364 000005 RESET
1213 011366 000137 006664 JMP PICO
1214
1221                                     .SBTTL MANUAL DISPLAY ROUTINE
1222
1223 011372 012706 001100 FULRMP: MOV #STACK,SP ;LOAD POINTER
1224 011376 004737 001742 JSR PC,LDTRAP ;LOAD BUS ADDRESS
1225 011402 032777 002000 167530 BIT #SW10,ASWR ;TEST SR 10=1
1226 011410 001403 BEQ 1$ ;BR IF NOT
1227 011412 052777 000020 170024 BIS #BIT4,ASTAT ;ENABLE EXT. DELAY
1228 011420 013700 001446 1$: MOV DAC0,RO ;GET BUS ADDRESS
1229 011424 004737 011462 JSR PC,10$ ;LOAD THE RAMP ON DAC #1
1230 011430 013700 001450 MOV DAC1,RO ;GET BUS ADDRESS
1231 011434 004737 011462 JSR PC,10$ ;LOAD THE RAMP ON DAC #1
1232 011440 013700 001452 MOV DAC2,RO ;GET BUS ADDRESS
1233 011444 004737 011462 JSR PC,10$ ;LOAD THE RAMP ON DAC #2
1234 011450 013700 001454 MOV DAC3,RO ;GET THE BUS ADDRESS
1235 011454 004737 011462 JSR PC,10$ ;LOAD THE RAMP ON DAC #3
1236 011460 000757 BR 1$ ;BR BACK
1237
1238 011462 005010 10$: CLR (RO) ;CLEAR DAC
1239 011464 062710 000010 11$: ADD #10,(RO) ;UPDATE THE DATA
1240 011470 005710 TST (RO) ;TEST IF DONE
1241 011472 001374 BNE 11$ ;BR IF NOT
1242 011474 004737 013246 JSR PC,ACTRLC ;TEST FOR CTRL C
1243 011500 000207 RTS PC ;EXIT
1244 011502 000240 NOP
1245 011504 000240 NOP
1246

```

```

1248 .SBTTL AUTO CALIBRATION
1249 011506 012706 001100 AUTCAL: MOV #STACK, SP ;LOAD STACK POINTER
1250 011512 004737 001742 JSR PC, LDTRAP ;LOAD TRAN AND ADDRESSES
1251 011516 052737 100000 020430 BIS #BIT15, EVER1 ;INDICATE PROG. IS RUNNING ON THE TESTER HARD.
1252 011524 104401 TYPE
1253 011526 017325 AUTOCL ;TELL OPERATOR
1254 011530 104401 TYPE
1255 011532 017033 SELD01
1256
1257 ;LOAD BUS TRAP FOR OPERTOR FALSE SELECTION
1258
1259 011534 012737 013064 000004 MOV #AUTRAP, ERRVEC
1260 011542 012737 000054 001102 MOV #STN-1, $TSTNM ;LOAD TEST NUMBER
1261 011550 012737 011574 001110 MOV #PSUPPLY, $LPERR ;LOAD LOOP RETURN
1262 011556 012737 011574 001106 MOV #PSUPPLY, $LPADR ;LOAD LOOP ADDRESS
1302 ;
1303 011564 004537 014062 JSR R5, CROSS ;CHANGE RELAYS
1304 011570 020000 BIT13 ;USING BIT 13 FOR CRISS-CROSS WRAP MODE
1305 ;
1306 ;*****
(3) ;*TEST 55 TEST THAT CH 3 IS AT +2.5 BIPOLAR
(3) ;*****
(2) †ST55: SCOPE
1307 011572 000004 PSUPPLY: MOV V1754, $GDDAT ;LOAD EXPECTED VALUE
1308 011574 013737 014514 001124 JSR R5, CONVRT ;CONVERT
1309 011602 004537 014370 CH3 ;CH 3
1310 011610 013737 014526 001126 MOV ADEND, $BDDAT ;LOAD VALUE READ
1311 011616 013737 014516 014534 MOV V24, $SPREAD ;LOAD + OR - COUNT SPREAD
1312 011624 004737 014536 JSR PC, COMPAR ;COMPARE $GDDAT AND $BDDAT
1313 011630 000402 BR NSUPPLY ;;BR IF WITHIN TOLERANCE
1314 011632 104012 ERROR 12 ;CH 3 FAILED TO EQUAL EXPECTED VALUE
1315
1316 ;*****
(3) ;*TEST 56 TEST THAT CH 4 IS AT -2.5 BIPOLAR
(3) ;*****
(2) †ST56: SCOPE
1317 011634 000004 NSUPPLY: MOV V24, $GDDAT ;LOAD EXPECTED VALUE
1318 011636 013737 014516 001124 JSR R5, CONVRT ;CONVERT
1319 011644 004537 014370 CH4 ;CH 4
1320 011652 013737 014526 001126 MOV ADEND, $BDDAT ;LOAD VALUE READ
1321 011660 013737 014516 014534 MOV V24, $SPREAD ;LOAD + OR - COUNT SPREAD
1322 011666 004737 014536 JSR PC, COMPAR ;COMPARE $GDDAT AND $BDDAT
1323 011672 000401 BR TST57 ;;BR IF WITHIN TOLERANCE
1324 011674 104012 ERROR 12 ;CH 4 FAILED TO EQUAL EXPECTED VALUE

```


E05

MAINDEC-11-DZAAC-B
DZAACB.P11 T57

AA11-K DIAGNOSTIC MACY11 27(663) 19-DEC-76 08:39 PAGE 33
TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH

SEQ 0056

```

1326
(3)
(3)
(2) 011676 000004
1327 011700 012777 000000 167536
1328 011706 012737 001146 001124
1329 011714 004537 014370
1330 011720 000053
1331 011722 013737 014526 001126
1332 011730 023737 001124 001126
1333 011736 003401
1334 011740 104014
1335
1336
(3)
(3)
(2) 011742 000004
1337 011744 005077 167474
1338 011750 012777 012000 167466
1339 011756 012737 000120 001124
1340 011764 004537 014370
1341 011770 000053
1342 011772 013737 014526 001126
1343 012000 023737 001124 001126
1344 012006 002001
1345 012010 104015
1346

```

```

*****
;*TEST 57 TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
*****
†T57: SCOPE
MOV #0, @STAT ;LOAD STATUS
MOV #1146, $GDDAT ;LOAD EXPECTED VALUE
JSR R5, CONVRT ;CONVERT
CH53 ;CH 53
MOV ADEND, $BDDAT ;LOAD VALUE READ
CMP $GDDAT, $BDDAT ;COMPARE
BLE TST60 ;;BR IF LESS THAN OR EQUAL
ERROR 14 ;CH 53 FAILED TO EQUAL EXPECTED VALUE

```

```

*****
;*TEST 60 TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
*****
†T60: SCOPE
CLR @STAT ;CLEAR STATUS
MOV #BIT12!BIT10, @STAT ;SET ERASE
MOV #120, $GDDAT ;LOAD EXPECTED VALUE
JSR R5, CONVRT ;CONVERT
CH53 ;CH 53
MOV ADEND, $BDDAT ;LOAD VALUE READ
CMP $GDDAT, $BDDAT ;COMPARE
BGE TST61 ;;BR IF GREATER THAN OR EQUAL
ERROR 15 ;CH 53 FAILED TO EQUAL EXPECTED VALUE

```

```

1348
(3)
(3)
(2) 012012 000004
1349 012014 012777 006000 167422
1350 012022 012737 000120 001124
1351 012030 004537 014370
1352 012034 000053
1353 012036 013737 014526 001126
1354 012044 023737 001124 001126
1355 012052 002001
1356 012054 104015
1357
(3)
(3)
(2) 012056 000004
1358 012060 005077 167360
1359 012064 012737 000120 001124
1360 012072 004537 014370
1361 012076 000055
1362 012100 013737 014526 001126
1363 012106 023737 001124 001126
1364 012114 002001
1365 012116 104015
1366
(3)
(3)
(2) 012120 000004
1367 012122 012777 017000 167314
1368 012130 012737 001146 001124
1369 012136 004537 014370
1370 012142 000055
1371 012144 013737 014526 001126
1372 012152 023737 001124 001126
1373 012160 003401
1374 012162 104014
1375
(3)
(3)
(2) 012164 000004
1376 012166 012777 000020 167250
1377 012174 012737 000020 001124
1378 012202 005277 167236
1379 012206 017737 167232 001126
1380 012214 023737 001124 001126
1381 012222 001401
1382 012224 104001
1383 012226 052777 002000 167210
1384 012234 012737 002220 001124
1385 012242 017737 167176 001126
1386 012250 023737 001124 001126
1387 012256 001401
1388 012260 104001
1389 012262 005077 167156

```

```

*****
*TEST 61 TEST THAT WRITE-THRU (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
*****
TST61: SCOPE
MOV #BIT11!BIT10,@STAT ;SET WRITE-THRU
MOV #120,$GDDAT ;LOAD EXPECTED VALUE
JSR R5,CONVRT ;CONVERT
CH53 ;CH 54
MOV ADEND,$BDDAT ;LOAD VALUE READ
CMP $GDDAT,$BDDAT ;COMPARE
BGE TST62 ;;BR IF GREATER THAN OR EQUAL
ERROR 15 ;CH 54 FAILED TO EQUAL EXPECTED VALUE
*****
*TEST 62 TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
*****
TST62: SCOPE
CLR @STAT ;CLEAR STORE MODE
MOV #120,$GDDAT ;LOAD EXPECTED VALUE
JSR R5,CONVRT ;CONVERT
CH55 ;CH 55
MOV ADEND,$BDDAT ;LOAD VALUE READ
CMP $GDDAT,$BDDAT ;COMPARE
BGE TST63 ;;BR IF GREATER THAN OR EQUAL
ERROR 15 ;CH 53 FAILED TO EQUAL EXPECTED VALUE
*****
*TEST 63 TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
*****
TST63: SCOPE
MOV #BIT12!BIT11!BIT10!BIT9,@STAT ;SET STORE MODE
MOV #1146,$GDDAT ;LOAD EXPECTED VALUE
JSR R5,CONVRT ;CONVERT
CH55 ;CH 55
MOV ADEND,$BDDAT ;LOAD VALUE READ
CMP $GDDAT,$BDDAT ;COMPARE
BLE TST64 ;;BR IF LESS THAN OR EQUAL
ERROR 14 ;CH 55 FAILED TO EQUAL EXPECTED VALUE
*****
*TEST 64 TEST THAT DELAY RETURN SETS READY
*****
TST64: SCOPE
MOV #BIT4,@STAT ;LOAD EXT DELAY BIT
MOV #BIT4,$GDDAT ;LOAD EXPECTED
INC @STAT ;MAKE READY GO AWAY
MOV @STAT,$BDDAT ;READ STAT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF EXPECTED
ERROR 1 ;READY FAILED TO CLEAR
1$: BIS #BIT10,@STAT ;LOAD NON-STORE L
MOV #BIT10!BIT7!BIT4,$GDDAT ;LOAD EXPECTED
MOV @STAT,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;;BR IF SAME
ERROR 1 ;EXT. DELAY RETURN FAILED TO SET READY
2$: CLR @STAT

```

```

1391 (3)
1392 (3)
1393 (2) 012266 000004
1394 012270 012777 014000 167146
1395 012276 012737 001100 001124
1396 012304 004537 014370
1397 012310 000056
1398 012312 013737 014526 001126
1399 012320 023737 001124 001126
1400 012326 003401
1401 012330 104014
1402 (3)
1403 (3)
1404 (2) 012332 000004
1405 012334 012777 003000 167102
1406 012342 012737 000120 001124
1407 012350 004537 014370
1408 012354 000056
1409 012356 013737 014526 001126
1410 012364 023737 001124 001126
1411 012372 002001
1412 012374 104015
1413 012376 005077 167042

;*****
;*TEST 65 TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
;*****
TST65: SCOPE
MOV #BIT12!BIT11,@STAT ;LOAD STATUS
MOV #1100,$GDDAT ;LOAD EXPECTED VALUE
JSR RS,CONVRT ;CONVERT
CH56 ;CH 56
MOV ADEND,$BDDAT ;LOAD VALUE READ
CMP $GDDAT,$BDDAT ;COMPARE
BLE TST66 ;;BR IF LESS THAN OR EQUAL
ERROR 14 ;CH 56 FAILED TO EQUAL EXPECTED
; VALUE

;*****
;*TEST 66 TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
;*****
TST66: SCOPE
MOV #BIT9!BIT10,@STAT ;SET CHANNEL 2
MOV #120,$GDDAT ;LOAD EXPECTED VALUE
JSR RS,CONVRT ;CONVERT
CH56 ;CH 56
MOV ADEND,$BDDAT ;LOAD VALUE READ
CMP $GDDAT,$BDDAT ;COMPARE
BGE 1$ ;;BR IF GREATER THAN
ERROR 15 ;CH 56 FAILED TO EQUAL EXPECTED
; VALUE
1$: CLR @STAT ;CLEAR BIT 9

```

```

1415
(3)
(3)
(2) 012402 000004
1416 012404 005077 167034
1417 012410 005037 001124
1418 012414 012777 010000 167022
1419 012422 017737 167016 001126
1420 012430 023737 001124 001126
1421 012436 001401
1422 012440 104001
1423
1424 012442 052777 001000 002054 1$:
1425 012450 042777 001000 002046
1426 012456 012737 000200 001124
1427 012464 017737 166754 001126
1428 012472 023737 001124 001126
1429 012500 001401
1430 012502 104001
1431

```

```

:*****
:*TEST 67 TEST THAT "ERASE" AND DONE CAN BE CLEARED AND SET
:*****
†T67:

```

```

SCOPE
CLR @STAT ;CLEAR
CLR $GDDAT ;CLEAR EXPECTED
MOV #BIT12,@STAT ;LOAD EXPECTED
MOV @STAT,$BDDAT ;READ REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF SET
ERROR 1 ;ERASE FAILED TO CLEAR READY, CHECK G5036-
;BCOBR CONNECTION: A-VV, VV-A
;SET CH 2 <GENERATE ERASE RETURN OF GOOD AR11>
BIS #BIT9,@GSTAT ;CLEAR CH 2
BIC #BIT9,@GSTAT ;LOAD EXPECTED
MOV #BIT7,$GDDAT ;READ REG
MOV @STAT,$BDDAT ;COMPARE
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST70 ;;BR IF EQUAL
ERROR 1 ;READY FAILED TO SET ON THE END
; OF ERASE RETURN (CLEARING CH 2)

```

```

1433      ;*****
1434      ;*TEST 70      ADJUSTMENT ROUTINES FOR DAC 0 - 1
1435      ;*****
1436      TST70:  SCOPE
1437      (3)      012504 000004
1438      (3)      012506 012737 000001 001166
1439      (2)      012514 004537 013770      ;DO 1 ITERATION
1440      (1)      012506 012737 000001 001166 ;OFFSET ADJUST FOR DAC 0
1441      1434      012514 004537 013770      JSR      R5,SNDVLT      ;LOAD THE VOLTAGE
1442      1435      012520 020207      NS0000
1443      1436      012522 005077 166720      CLR      @DAC0      ;LOAD DAC REGISTER
1444      1437      012526 004537 014062      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
1445      (1)      012532 060000      BIT14!BIT13
1446      1438      012534 104401      TYPE
1447      1439      012536 017537      R38      ;TELL OPR. TO ADJUST R38
1448      1440      012540 004737 013310      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1449      1441      ;OFFSE T ADJUSTMENT FOR DAC 1
1450      1442      012544 004537 014062      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
1451      (1)      012550 020000      BIT13
1452      1443      012552 005077 166672      CLR      @DAC1      ;LOAD THE DAC REGISTER
1453      1444      012556 104401      TYPE
1454      1445      012560 017604      R40      ;TELL OPR. TO ADJUST R40
1455      1446      012562 004737 013310      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1456      1447      ;GAIN ADJUST FOR DAC 0
1457      1448      012566 004537 013770      JSR      R5,SNDVLT
1458      1449      012572 020222      P49976
1459      1450      012574 012777 007777 166644      MOV      #7777,@DAC0      ;LOAD DAC 0
1460      (1)      012602 004537 014062      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
1461      1451      012606 060000      BIT14!BIT13
1462      1452      012610 104401      TYPE
1463      1453      012612 017763      R37      ;TELL OPR. TO ADJUST R37
1464      1454      012614 004737 013310      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1465      1455      ;GAIN ADJUST FOR DAC 1
1466      1456      012620 004537 014062      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
1467      (1)      012624 020000      BIT13
1468      1457      012626 012777 007777 166614      MOV      #7777,@DAC1      ;LOAD DAC 1
1469      1458      012634 104401      TYPE
1470      1459      012636 020030      R39      ;TELL OPR. TO ADJUST R39
1471      1460      012640 004737 013310      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1472      1461      ;DAC 0 D/A FUNCTION CHECK
1473      1462      012644 004537 014266      JSR      R5,FUNDAC      ;GO TO DAC FUNCTION SUB.
1474      1463      012650 001446      DAC0
1475      1464      012652 000051      CHS1
1476      1465      ;DAC 1 D/A FUNCTION CHECK
1477      1466      012654 004537 014266      JSR      R5,FUNDAC
1478      1467      012660 001450      DAC1
1479      1468      012662 000052      CHS2
1480      1469

```

```

1477      ;*****
(3)      ;*TEST 71      ADJUSTMENT ROUTINES FOR DAC 2 - 3
(3)      ;*****
(2) 012664 000004
(1) 012666 012737 000001 001166
1478      †ST71: SCOPE
1479      MOV      #1,$TIMES      ;;DO 1 ITERATION
1480      ;NOW DO DAC 2 AND 3
1481      TYPE
1482      SELD23
1483      ;OFFSET ADJUST TOR DAC 2
1484      JSR      R5,$NDVLT      ;LOAD E.D.C
1485      N50000
1486      CLR      @DAC2      ;LOAD DAC VALUE
1487      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
(1) 012716 060000      BIT14!BIT13
1488      TYPE
1489      R42      ;TELL OPR. TO ADJUST R42
1490      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1491      ;OFFSET ADJUST FOR DAC 3
1492      CLR      @DAC3
1493      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 012740 020000      BIT13
1494      CLR      @DAC3      ;LOAD THE DAC REGISTER
1495      TYPE
1496      R44      ;TELL OPR. TO ADJUST R44
1497      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1498
1499      ;GAIN ADJUST FOR DAC 2
1500      JSR      R5,$NDVLT
1501      P49976
1502      MOV      #7777,@DAC2      ;LOAD DAC 2
1503      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
(1) 012776 060000      BIT14!BIT13
1504
1505      TYPE
1506      R41      ;TELL OPR. TO ADJUST R41
1507      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1508
1509      ;GAIN ADJUST FOR DAC 3
1510      MOV      #7777,@DAC3      ;LOAD DAC 3
1511      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 013022 020000      BIT13
1512
1513      TYPE
1514      R43      ;TELL OPR. TO ADJUST R43
1515      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1516

```

1518 ;DAC 2 D/A FUNCTION CHECK

1519
1520 013034 004537 014266 JSR R5,FUNDAC
1521 013040 001452 DAC2
1522 013042 000051 CH51
1523

1524 ;DAC 3 D/A FUNCTION CHECK

1525
1526 013044 004537 014266 JSR R5,FUNDAC
1527 013050 001454 DAC3
1528 013052 000052 CH52
1529

1530 013054 104401 TYPE
1531 013056 017145 CALDON
1532 013060 000137 011506 JMP AUTCAL
1533

1534 013064 022626
1535 013066 012737 000006 000004 AUTRAP: CMP (SP)+,(SP)+
1536 013074 104401 MOV #6,ERRVEC
1537 013076 017411 TYPE
1538 013100 000137 002312 JMP CTRLC

				.SBTTL MANUAL LOGIC TEST	
1540					
1541					
1542	013104	012706	001100	MANUL:	MOV #STACK,SP ;LOAD STACK
1543	013110	004737	001742		JSR PC,LDTRAP ;LOAD BUS ADDRESS
1544	013114	104401			TYPE
1545	013116	017204			MANHED
1546	013120	004737	013246	1\$:	JSR PC,ACTRLC ;TEST FOR CTRL C
1547	013124	104407			CKSWR
1548	013126	017700	166006		MOV @SWR,R0 ;READ THE SWITCHES
1549	013132	010001			MOV R0,R1 ;COPY
1550	013134	042700	017777		BIC #17777,R0 ;MASK OFF BITS
1551	013140	001003			BNE 2\$;BR IF NOT DAC 0
1552	013142	013702	001446		MOV DAC0,R2 ;LOAD DAC 0 BUS ADDRESS
1553	013146	000424			BR 10\$
1554	013150	022700	020000	2\$:	CMP #BIT13,R0 ;TEST FOR DAC 1
1555	013154	001003			BNE 3\$;BR IF NOT
1556	013156	013702	001450		MOV DAC1,R2 ;LOAD DAC 1 BUS ADDRESS
1557	013162	000416			BR 10\$
1558	013164	022700	040000	3\$:	CMP #BIT14,R0 ;TEST FOR DAC 2
1559	013170	001003			BNE 4\$;BR IF NOT
1560	013172	013702	001452		MOV DAC2,R2 ;LOAD DAC 2 BUS ADDRESS
1561	013176	000410			BR 10\$
1562	013200	022700	060000	4\$:	CMP #BIT14!BIT13,R0 ;TEST FOR DAC 3
1563	013204	001003			BNE 5\$;BR IF NOT
1564	013206	013702	001454		MOV DAC3,R2 ;LOAD DAC 3 BUS ADDRESS
1565	013212	000402			BR 10\$
1566	013214	013702	001444	5\$:	MOV STAT,R2 ;LOAD STATUS REG. BUS ADDRESS
1567					
1568	013220	010112		10\$:	MOV R1,(R2) ;LOAD THE SWITCHES INTO THE SELECTED ADDRESS
1569	013222	013703	001442		MOV DELAY,R3 ;PRESET THE DELAY
1570	013226	005303		11\$:	DEC R3 ;DELAY
1571	013230	001376			BNE 11\$
1572	013232	005012			CLR (R2) ;CLEAR THE REGISTER
1573	013234	013703	001442		MOV DELAY,R3 ;LOAD THE PRESET
1574	013240	005303		12\$:	DEC R3 ;DELAY
1575	013242	001376			BNE 12\$
1576	013244	000725			BR 1\$


```

1578
1579
1580
1581 013246 105777 165672 ACTRLC: TSTB @STKS ;INPUT FLAG ?
1582 013252 100014 BPL 1$ ;NO
1583 013254 017737 165666 013306 MOV @STKB,10$ ;READ THE CHARACTER
1584 013262 042737 177600 013306 BIC #177600,10$
1585 013270 022737 000003 013306 CMP #3,10$ ;TEST FOR CTRL C
1586 013276 001002 BNE 1$ ;BR IF NOT
1587 013300 000137 002312 JMP CTRLC
1588 013304 000207 1$: RTS PC ;EXIT
1589 013306 000000 10$: 0
1590
1591 ;WAIT FOR A "SPACE" CHARACTER
1592
1593 013310 105777 165630 CSPACE: TSTB @STKS ;WAIT FOR CHAR
1594 013314 100375 BPL CSPACE
1595 013316 017737 165624 013366 MOV @STKB,10$ ;READ CHAR
1596 013324 042737 177600 013366 BIC #177600,10$ ;MASK
1597 013332 022737 000003 013366 CMP #3,10$ ;TEST FOR CTRL C
1598 013340 001002 BNE 1$
1599 013342 000137 002312 JMP CTRLC
1600 013346 022737 000040 013366 1$: CMP #40,10$ ;TEST FOR SPACE
1601 013354 001001 BNE 2$
1602 013356 000207 RTS PC ;EXIT
1603 013360 104401 2$: TYPE
1604 013362 016746 QMARK
1605 013364 000751 BR CSPACE
1606
1607 013366 000000 10$: 0

```

```

1609
1610
1611
1612 013370 012706 001100
1613 013374 004737 001742
1614 013400 104407
1615 013402 004737 013246
1616 013406 017700 165526
1617 013412 010077 166030
1618 013416 010077 166026
1619 013422 010077 166024
1620 013426 010077 166022
1621 013432 000762
1622
1623
1624
1625 013434 012706 001100
1626 013440 004737 001742
1627 013444 104407
1628 013446 004737 013246
1629 013452 017700 165462
1630 013456 004737 013472
1631 013462 005000
1632 013464 004737 013472
1633 013470 000765
1634
1635 013472 010077 165750
1636 013476 010077 165746
1637 013502 010077 165744
1638 013506 010077 165742
1639 013512 012700 000020
1640 013516 005300
1641 013520 100376
1642 013522 000207
1643
1644

      .SBTTL  MANUAL DAC CALIBRATION
CALDAC: MOV  #STACK, SP      ;LOAD STACK POINTER
        JSR  PC, LDTRAP      ;LOAD BUS ADDRESSES
1$:     CKSWR                 ;TEST FOR CTRL G
        JSR  PC, ACTRLC      ;TEST FOR CTRL C
        MOV  @SWR, RO        ;READ SWITCHES
        MOV  RO, @DAC0       ;LOAD DAC #0
        MOV  RO, @DAC1       ;LOAD DAC #1
        MOV  RO, @DAC2       ;LOAD DAC #2
        MOV  RO, @DAC3       ;LOAD DAC #3
        BR   1$

      .SBTTL  DYNAMIC DAC CALIBRATION
DYNCAL: MOV  #STACK, SP      ;LOAD STACK POINTER
        JSR  PC, LDTRAP      ;LOAD BUS ADDRESSES
1$:     CKSWR                 ;TEST FOR CTRL G
        JSR  PC, ACTRLC      ;TEST FOR CTRL C
        MOV  @SWR, RO        ;READ SWR
        JSR  PC, 10$         ;LOAD THE SWR VALUE TO ALL DACS
        CLR  RO              ;CLEAR RO
        JSR  PC, 10$         ;LOAD ALL DAC'S WITH 0
        BR   1$

10$:    MOV  RO, @DAC0       ;LOAD DAC #0
        MOV  RO, @DAC1       ;LOAD DAC #1
        MOV  RO, @DAC2       ;LOAD DAC #2
        MOV  RO, @DAC3       ;LOAD DAC #3
        MOV  #20, RO        ;LOAD DELAY COUNTER
11$:    DEC  RO              ;DELAY
        BPL  11$            ;WAIT
        RTS  PC              ;EXIT
    
```

```

1651          .SBTTL SUBROUTINE TO ERASE STORAGE SCOPE SCREEN
1652 013524 104407 CLRVC: CKSWR
1653 013526 032777 000040 165404 BIT #BITS,@SWR ;TEST SR BIT 5
1654 013534 001441 BEQ 3$ ;BR IF NOT A STORAGE SCOPE
1655 013536 012777 002000 165700 MOV #BIT10,@STAT ;ERASE THE SCREEN
1656 013544 052777 010000 165672 BIS #BIT12,@STAT
1657 013552 012700 000020 MOV #20,R0 ;SET UP DELAY
1658 013556 005001 CLR R1
1659 013560 032777 000040 165352 1$: BIT #BITS,@SWR ;TEST IF NOT STORAGE SCOPE
1660 013566 001424 BEQ 3$
1661 013570 105777 165650 TSTB @STAT ;TEST FOR READY
1662 013574 100421 BMI 3$ ;BRANCH IF SET
1663 013576 005301 DEC R1 ;DELAY
1664 013600 001367 BNE 1$
1665 013602 004737 013246 JSR PC,ACTRLC ;TEST FOR CTRL C
1666 013606 005300 DEC R0 ;DELAY
1667 013610 001363 BNE 1$
1668 013612 037727 165322 010000 BIT @SWR,#SW12 ;TEST INHIBIT PRINTOUT
1669 013620 001002 BNE 2$
1670 013622 104401 TYPE
1671 013624 015336 MES3
1672 013626 104407 165304 2$: CKSWR ;TEST FOR CTRL C
1673 013630 005777 TST @SWR ;TEST @SWR
1674 013634 100001 BPL 3$
1675 013636 000000 HALT
1676 013640 000207 3$: RTS PC ;ERASE RETURN FAILED TO SET READY
1677
1678          .SBTTL SUBROUTINE TO DRAW A HORIZONTAL LINE
1679 013642 005077 165576 LOADVC: CLR @STAT ;CLEAR STATUS
1680 013646 012737 007777 020420 MOV #7777,TEMP1
1681 013654 013700 001444 MOV STAT,R0
1682 013660 032777 000010 165252 BIT #SW03,@SWR ;TEST SR BIT 3
1683 013666 001405 BEQ 4$ ;BR IF DAC #0 AND 1
1684 013670 013701 001452 MOV DAC2,R1 ;LOAD X AXIS
1685 013674 013702 001454 MOV DAC3,R2 ;LOAD Y AXIS
1686 013700 000404 BR 5$
1687 013702 013701 001446 4$: MOV DAC0,R1
1688 013706 013702 001450 MOV DAC1,R2
1689 013712 012710 002000 5$: MOV #BIT10,(0) ;SET STORE MODE
1690 013716 013712 020420 MOV TEMP1,(2)
1691 013722 012711 007777 1$: MOV #7777,(1)
1692 013726 000402 BR 3$
1693 013730 162711 000010 2$: SUB #10,(1)
1694 013734 005210 3$: INC (0)
1695 013736 105710 TSTB (0)
1696 013740 100376 BPL -2
1697 013742 022711 000007 CMP #7,(1)
1698 013746 001370 BNE 2$
1699 013750 104407 CKSWR
1700 013752 004737 013246 JSR PC,ACTRLC ;TEST FOR CTRL G
1701 013756 162712 000010 SUB #10,(2) ;TEST FOR CTRL C
1702 013762 100357 BPL 1$
1703 013764 000207 RTS PC
1704

```

```

1706 013766 167772          FILZ: 167772          ;DR11-C OUTPUT CONTROL REGISTER
1707
1708          ;SUBROUTINE TO LOAD THE EDC VOLTAGE
1709          ;DATA FORMAT IS:          STX
1710          ;                          P OR N
1711          ;                          N VOLTS
1712          ;                          O TENTHS VOLTS
1713          ;                          O
1714          ;                          O
1715          ;                          O
1716          ;                          O
1717          ;                          V          VOLTS
1718          ;                          ETX
1719
1720 013770 012500          SNDVLT: MOV      (R5)+,R0          ;LOAD POINTER
1721 013772 112001          2$:  MOVB   (R0)+,R1          ;GET A BYTE
1722 013774 001422          BEQ     3$          ;BR IF TERM.
1723 013776 005101          COM     R1          ;CONVERT IT
1724 014000 110177 177762  MOVB   R1,DFILZ          ;LOAD FUNNY DATA BYTE
1725 014004 012701 001000  MOV     #1000,R1          ;LOAD DELAY
1726 014010 005301          5$:  DEC     R1          ;DELAY
1727 014012 001376          BNE     5$
1728 014014 042777 000200 177744  BIC     #BIT7,DFILZ          ;CLEAR BIT 7
1729 014022 012701 001000  MOV     #1000,R1          ;LOAD DELAY COUNT
1730 014026 005301          1$:  DEC     R1          ;DELAY
1731 014030 001376          BNE     1$
1732 014032 052777 000200 177726  BIS     #BIT7,DFILZ          ;SET BIT 7
1733 014040 000754          BR      2$          ;DO MORE DATA
1734
1735 014042 012701 000000          3$:  MOV     #0,R1          ;LOAD DELAY
1736 014046 152777 000177 177712  BISB   #177,DFILZ          ;SET BITS 0-6
1737 014054 005301          4$:  DEC     R1          ;DELAY
1738 014056 001376          BNE     4$
1739 014060 000205          RTS     R5          ;EXIT
1740
1741 014062 012537 014122          CROSS: MOV     (R5)+,10$          ;READ REG.
1742 014066 052737 000177 014122  BIS     #177,10$          ;LOAD LOW 7 BITS
1743 014074 013777 014122 177664  MOV     10$,DFILZ          ;LOAD RELAYS
1744 014102 012700 000001          MOV     #1,R0          ;LOAD DELAY
1745 014106 005001          CLR     R1
1746 014110 005301          1$:  DEC     R1          ;DELAY
1747 014112 001376          BNE     1$
1748 014114 005300          DEC     R0          ;DELAY
1749 014116 001374          BNE     1$          ;DONE ?
1750 014120 000205          RTS     R5          ;EXIT
1751 014122 000000          10$: 0

```

```

1753 .SBTTL TIMER ROUTINE FOR VISUAL TEST PATTERNS
1754 ; ENTER VIA JSR PC,TIMER
1755
1756 014124 104407 TIMER: CKSWR
1757 014126 004737 013246 JSR PC,ACTRLC ;TEST FOR CTRL C
1758 014132 017737 165002 020412 MOV @SWR,TIMSV
1759
1760 014140 032737 000400 020412 TIMERA: BIT #BIT8,TIMSV
1761 014146 001006 BNE TIMER2 ;BIT 8 SET ?
1762 014150 005337 020414 DEC TICKS ;NO, DECREMENT TICKS
1763 014154 001002 BNE TIMER1
1764 014156 062716 000002 ADD #2,(6) ;ADD 2 TO STACK POINTER
1765 014162 000207 TIMER1: RTS PC ;RETURN
1766
1767 ; SWR 8=1 SELECT TEST TO LOCK ON
1768 ; SWR 2-0= TEST NUMBER
1769
1770 014164 042737 177770 020412 TIMER2: BIC #177770,TIMSV
1771 014172 006337 020412 ASL TIMSV
1772 014176 062737 014220 020412 ADD #ROUTPT,TIMSV
1773 014204 017737 004202 020412 MOV @TIMSV,TIMSV
1774 014212 022600 CMP (SP)+,R0
1775 014214 000177 004172 TIMER4: JMP @TIMSV
1776
1777 014220 006664 ROUTPT: PICO ;DISPLAY A HORIZONTAL LINE
1778 014222 006740 PIC1 ;DISPLAY A VERTICAL LINE
1779 014224 007116 PIC3 ;DISPLAY A SQUARE
1780 014226 007346 PIC4 ;DISPALY A "X"
1781 014230 007554 PIC5 ;DISPLAY SETTLING TIME
1782 014232 010012 PIC6 ;DISPLAY CHARACTER SET
1783 014234 011030 PIC7 ;DISPLAY CHANNEL TEST
1784 014236 011334 PIC12 ;DISPLAY ERASE AND PHOSPHOR TEST
1785
1786 014240 013737 014264 020422 CHTIME: MOV CPTYPE,BRLEV1
1787 014246 005337 020422 1$: DEC BRLEV1
1788 014252 001403 BEQ 2$
1789 014254 006337 020414 ASL TICKS
1790 014260 000772 BR 1$
1791 014262 000207 2$: RTS PC
1792
1793 014264 000001 CPTYPE: 1

```

```

1795
1796
1797
1798
1799 014266 013537 014366 FUNDAC: MOV @ (R5)+, XDACUT ; GET BUS ADDRESS OF ARUT DAC
1800 014272 012537 014316 MOV (R5)+, 60$ ; GET CHANNEL INPUT FOR ARKG
1801 014276 012777 007600 000062 MOV #7600, @XDACUT ; PRESET THE DAC
1802 014304 012737 001700 001124 MOV #1700, $GDDAT ; LOAD EXPECTED
1803
1804 014312 004537 014370 1$: JSR R5, CONVRT ; CONVERT USING ARKG
1805 014316 000051 60$: CHS1 ;
1806
1807 014320 012737 000010 014534 MOV #10, SPREAD ; LOAD LIMIT
1808 014326 004737 014536 JSR PC, COMPAR ; TEST IF WITHIN LIMIT
1809 014332 000401 BR 2$ ;; BR IF WITHIN
1810 014334 104016 ERROR 16 ; SELECTED DAC HAS A LINEARITY ERROR
1811
1812 014336 162777 000400 000022 2$: SUB #400, @XDACUT ; UPDATE DAC
1813 014344 162737 000200 001124 SUB #200, $GDDAT ; UPDATE GOOD VALUE
1814 014352 100357 BPL 1$ ;; BR IF NOT FINISHED
1815 014354 000205 RTS R5 ; EXIT
1816 014356 000240 NOP
1817 014360 000240 NOP
1818 014362 000240 NOP
1819
1820 014364 000000 10$: 0
1821 014366 000000 XDACUT: 0

```

```

1823                                     ;SUBROUTINE TO GET AVERAGE OF 32 CONVERSIONS FROM THE KNOWN GOOD AR 11
1824
1825 014370 012537 014510 CONVRT: MOV (R5)+,10$
1826 014374 013737 014510 014532 MOV 10$,CHANL
1827 014402 000337 014510 SWAB 10$
1828 014406 012737 000020 014512 MOV #16.,11$
1829 014414 005037 014526 CLR ADEND
1830 014420 013777 014510 000072 MOV 10$,GADCS
1831 014426 005277 000066 2$: INC GADCS
1832 014432 105777 000062 1$: TSTB GADCS
1833 014436 100375 BPL 1$
1834 014440 067737 000056 014526 ADD GADDBR,ADEND
1835 014446 005337 014512 DEC 11$
1836 014452 001365 BNE 2$
1837 014454 006237 014526 ASR ADEND
1838 014460 006237 014526 ASR ADEND
1839 014464 006237 014526 ASR ADEND
1840 014470 006237 014526 ASR ADEND
1841 014474 005537 014526 ADC ADEND ;ROUND UP
1842 014500 013737 014526 001126 MOV ADEND,$BDDAT
1843 014506 000205 RTS RS
1844
1845 014510 000000 10$: C
1846 014512 000000 11$: 0
1847 014514 001754 V1754: 1754
1848 014516 000024 V24: 24
1849 014520 170440 GADCS: 170440
1850 014522 170442 GADDBR: 170442
1851 014524 170450 GSTAT: 170450
1852 014526 000000 ADEND: 0
1853 014530 000000 FCHANL: 0
1854 014532 000000 CHANL: 0
1855 014534 000000 SPREAD: 0
1856
1857 014536 010046 COMPAR: MOV R0,-(SP)
1858 014540 010146 MOV R1,-(SP)
1859 014542 013700 001124 MOV $GDDAT,R0 ;LOAD R0
1860 014546 013701 001126 MOV $BDDAT,R1 ;LOAD R1
1861 014552 160100 SUB R1,R0 ;SUBTRACT
1862 014554 100001 BPL B$ ;
1863 014556 005400 NEG R0
1864 014560 020037 014534 B$: CMP R0,SPREAD ;MAGNITUDE OF DIFF. IN R0
1865 014564 003405 BLE 10$
1866 014566 012601 9$: MOV (SP)+,R1
1867 014570 012600 MOV (SP)+,R0
1868 014572 062716 000002 ADD #2,(SP)
1869 014576 000207 RTS PC
1870
1871 014600 012601 10$: MOV (SP)+,R1
1872 014602 012600 MOV (SP)+,R0
1873 014604 000207 RTS PC
1874

```

1876
1877
1878

014606	005015	040412	030501
014614	026461	020113	044504
014622	043501	047516	052123
014630	041511	052040	051505
014636	026124	024040	040515
014644	047111	042504	026503
014652	030461	042055	040532
014660	041501	041055	024460
014666	005015		

.SBTTL ASCII MESSAGES

TITLE: .ASCII <15><12><12>'AA11-K DIAGNOSTIC TEST, (MAINDEC-11-DZAAC-B0)'<15><12>

1879

014670	042523	042514	052103
014676	052040	051505	051524
014704	041040	020131	054524
014712	044520	043516	040440
014720	052040	047527	046040
014726	052105	042524	020122
014734	027111	027104	005015

.ASCII /SELECT TESTS BY TYPING A TWO LETTER I.D./<15><12>

1880

014742	046101	035011	052501
014750	047524	046040	043517
014756	041511	052040	051505
014764	006524	012	

.ASCII /AL :AUTO LOGIC TEST/<15><12>

1881

014767	101	004504	040472
014774	052125	020117	044504
015002	050123	040514	020131
015010	042524	052123	024040
015016	044504	050123	040514
015024	020131	041523	050117
015032	020105	047503	047116
015040	041505	042524	024504
015046	005015		

.ASCII /AD :AUTO DISPLAY TEST (DISPLAY SCOPE CONNECTED)/<15><12>

1882

015050	041501	035011	052501
015056	047524	041440	046101
015064	041111	040522	044524
015072	047117	024040	040501
015100	030461	045455	047440
015106	052120	047511	020116
015114	042524	052123	040440
015122	042522	020101	047117
015130	054514	006451	012

.ASCII /AC :AUTO CALIBRATION (AA11-K OPTION TEST AREA ONLY)/<15><12>

1883

015135	115	004514	046472
015142	047101	040525	020114
015150	047514	044507	020103
015156	042524	052123	024040
015164	053523	020122	032461
015172	030455	020063	020075
015200	042522	044507	052123
015206	051105	020040	031061
015214	030055	036440	042040
015222	052101	024501	005015

.ASCII /ML :MANUAL LOGIC TEST (SWR 15-13 = REGISTER 12-0 = DATA)/<15><12>

1884

015230	042115	035011	040515
015236	052516	046101	042040
015244	051511	046120	054501
015252	024040	047516	042040

.ASCII /MD :MANUAL DISPLAY (NO DISPLAY SCOPE)/<15><12>

	015260	051511	046120	054501	
	015266	051440	047503	042520	
	015274	006451	012		
1885	015277	115	004503	046472	.ASCII /MC :MANUAL CALIBRATION LOOP/<15><12>
	015304	047101	040525	020114	
	015312	040503	044514	051102	
	015320	052101	047511	020116	
	015326	047514	050117	005015	
1886	015334	000056			.ASCIZ /./
1887	015336	005015	051105	051501	MES3: .ASCIZ <15><12>"ERASE RETURN FAILED TO SET READY"<15><12>
	015344	020105	042522	052524	
	015352	047122	043040	044501	
	015360	042514	020104	047524	
	015366	051440	052105	051040	
	015374	040505	054504	005015	
	015402	000			
1888	015403	015	051412	047503	MES6: .ASCIZ <15><12>'SCOPE ADJUSTMENT TEST'
	015410	042520	040440	045104	
	015416	051525	046524	047105	
	015424	020124	042524	052123	
	015432	000			
1889	015433	015	051412	051127	MES15: .ASCII <15><12>/SWR8 AND 0 THRU 2 CONTROL PATTERN/<15><12>
	015440	020070	047101	020104	
	015446	020060	044124	052522	
	015454	031040	041440	047117	
	015462	051124	046117	050040	
	015470	052101	042524	047122	
	015476	005015			
1890	015500	053523	020122	044502	.ASCII 'SWR BIT 3 SELECTS DAC 0+1 OR DAC 2+3'<15><12>
	015506	020124	020063	042523	
	015514	042514	052103	020123	
	015522	040504	020103	025460	
	015530	020061	051117	042040	
	015536	041501	031040	031453	
	015544	005015			
1891	015546	053523	020122	044502	.ASCII /SWR BIT 5 SELECTS STORAGE SCOPE MODE/<15><12>
	015554	020124	020065	042523	
	015562	042514	052103	020123	
	015570	052123	051117	043501	
	015576	020105	041523	050117	
	015604	020105	047515	042504	
	015612	005015			
1892	015614	053523	020122	044502	.ASCII /SWR BIT 7 ENABLES VERTICAL SETTling TEST/<15><12>
	015622	020124	020067	047105	
	015630	041101	042514	020123	
	015636	042526	052122	041511	
	015644	046101	051440	052105	
	015652	046124	047111	020107	
	015660	042524	052123	005015	
1893	015666	053523	020122	044502	.ASCIZ /SWR BIT 9 SELECTS TWO'S COMPLEMENT DISPLAY MODE/<15><12>
	015674	020124	020071	042523	
	015702	042514	052103	020123	
	015710	053524	023517	020123	
	015716	047503	050115	042514	

	015724	042515	052116	042040		
	015732	051511	046120	054501		
	015740	046440	042117	006505		
	015746	000012				
1894	015750	052123	052101	051525	EM1:	.ASCIZ /STATUS REGISTER IN ERROR/
	015756	051040	043505	051511		
	015764	042524	020122	047111		
	015772	042440	051122	051117		
	016000	000				
1895	016001	104	041501	020060	EM2:	.ASCIZ /DAC0 REGISTER IN ERROR/
	016006	042522	044507	052123		
	016014	051105	044440	020116		
	016022	051105	047522	000122		
1896	016030	040504	030503	051040	EM3:	.ASCIZ /DAC1 REGISTER IN ERROR/
	016036	043505	051511	042524		
	016044	020122	047111	042440		
	016052	051122	051117	000		
1897	016057	104	041501	020062	EM4:	.ASCIZ /DAC2 REGISTER IN ERROR/
	016064	042522	044507	052123		
	016072	051105	044440	020116		
	016100	051105	047522	000122		
1898	016106	040504	031503	051040	EM5:	.ASCIZ /DAC3 REGISTER IN ERROR/
	016114	043505	051511	042524		
	016122	020122	047111	042440		
	016130	051122	051117	000		
1899	016135	101	030501	026461	EM6:	.ASCIZ /AA11-K FAILED TO INTERRUPT/
	016142	020113	040506	046111		
	016150	042105	052040	020117		
	016156	047111	042524	051122		
	016164	050125	000124			
1900	016170	040501	030461	045455	EM7:	.ASCIZ /AA11-K INTERRUPTED IN ERROR/
	016176	044440	052116	051105		
	016204	052522	052120	042105		
	016212	044440	020116	051105		
	016220	047522	000122			
1901	016224	052502	020123	044524	EM10:	.ASCIZ /BUS TIME-CUT WHEN REFERENCING THE AA11-K/
	016232	042515	047455	052125		
	016240	053440	042510	020116		
	016246	042522	042506	042522		
	016254	041516	047111	020107		
	016262	044124	020105	040501		
	016270	030461	045455	000		
1902	016275	101	030501	026461	EM11:	.ASCIZ /AA11-K READY BIT ERROR/
	016302	020113	042522	042101		
	016310	020131	044502	020124		
	016316	051105	047522	000122		
1903	016324	047520	042527	020122	EM12:	.ASCIZ /POWER SUPPLY VOLTAGE WAS INCORRECT/
	016332	052523	050120	054514		
	016340	053040	046117	040524		
	016346	042507	053440	051501		
	016354	044440	041516	051117		
	016362	042522	052103	000		
1904	016367	117	042516	041040	EM13:	.ASCIZ /ONE BIT THE DUST/
	016374	052111	052040	042510		

1905	016402	042040	051525	000124				
	016410	040501	030461	045455	EM14:	.ASCIZ	/AA11-K LOGIC SIGNAL HIGH OUTPUT TO LOW/	
	016416	046040	043517	041511				
	016424	051440	043511	040516				
	016432	020114	044510	044107				
	016440	047440	052125	052520				
	016446	020124	047524	046040				
	016454	053517	000					
1906	016457	101	030501	026461	EM15:	.ASCIZ	/AA11-K LOGIC SIGNAL LOW OUTPUT TO HIGH/	
	016464	020113	047514	044507				
	016472	020103	044523	047107				
	016500	046101	046040	053517				
	016506	047440	052125	052520				
	016514	020124	047524	044040				
	016522	043511	000110					
1907	016526	042523	042514	052103	EM16:	.ASCIZ	/SELECTED DAC HAS A LINEARITY ERROR/	
	016534	042105	042040	041501				
	016542	044040	051501	040440				
	016550	046040	047111	040505				
	016556	044522	054524	042440				
	016564	051122	051117	000				
1908	016571	105	051122	041520	DH1:	.ASCIZ	/ERRPC IBASE EXPECT BAD/	
	016576	044411	040502	042523				
	016604	020040	042440	050130				
	016612	041505	020124	020040				
	016620	041040	042101	000				
1909	016625	105	051122	041520	DH6:	.ASCIZ	/ERRPC IBASE/	
	016632	044411	040502	042523				
	016640	000						
1910	016641	105	051122	041520	DH13:	.ASCIZ	/ERRPC IBASE # FIRST # NOW/	
	016646	044411	040502	042523				
	016654	021411	043040	051111				
	016662	052123	021411	047040				
	016670	053517	000					
1911	016673	105	051122	041520	DH14:	.ASCIZ	/ERRPC IBASE GOOD BAD/	
	016700	044411	040502	042523				
	016706	043411	047517	004504				
	016714	040502	000104					
1912	016720	051105	050122	004503	DH16:	.ASCIZ	/ERRPC DACADR GOOD BAD/	
	016726	040504	040503	051104				
	016734	043411	047517	004504				
	016742	040502	000104					
1913								
1914	016746	015	012	077	QMARK:	.BYTE	15,12,77,15,12,0	
	016751	015	012	000				
1915	016754	136	103	015	CONTC:	.BYTE	136,103,15,12,56,0	
	016757	012	056	000				
1916	016762	015	012		FOUND1:	.BYTE	15,12	
1917	016764	051120	043517	040522		.ASCIZ	/PROGRAM DETECTED /	
	016772	020115	042504	042524				
	017000	052103	042105	000040				
1918	017006	034050	020051	020040	FOUND2:	.ASCIZ	/(8) AA11-K(S) /	
	017014	040501	030461	045455				
	017022	051450	020051	000040				

```

1919 017030 015 012 000
1920 017033 015 012
1921 017035 123 052105 051440
      017042 044527 041524 020110
      017050 047524 051440 046105
      017056 041505 020124 040504
      017064 020103 020060 047101
      017072 020104 006461 000012
1922 017100 015 012
1923 017102 042523 020124 053523
      017110 052111 044103 052040
      017116 020117 042523 042514
      017124 052103 042040 041501
      017132 031040 040440 042116
      017140 031440 005015 000
1924 017145 015 012
1925 017147 101 052125 020117
      017154 040503 044514 051102
      017162 052101 047511 020116
      017170 047503 050115 042514
      017176 042524 006504 000012
1926 017204 015 012
1927 017206 053523 030440 026465
      017214 031461 051440 046105
      017222 041505 020124 044124
      017230 020105 042522 044507
      017236 052123 051105 005015
1928 017244 053523 030440 026462
      017252 030060 040440 042522
      017260 046040 040517 042504
      017266 020104 047111 047524
      017274 052040 042510 051440
      017302 046105 041505 042524
      017310 020104 042522 044507
      017316 052123 051105 005015
1929 017324 000
1930 017325 015 012
1931 017327 101 052125 020117
      017334 040503 044514 051102
      017342 052101 047511 020116
      017350 040450 030501 026461
      017356 020113 050117 044524
      017364 047117 052040 051505
      017372 020124 051101 040505
      017400 047440 046116 024531
1932 017406 015 012 000
1933 017411 015 041012 051525
      017416 052040 040522 020120
      017424 044127 047105 046040
      017432 040517 044504 043516
      017440 053040 046117 040524
      017446 042507 047440 020122
      017454 044103 047101 044507
      017462 043516 040440 051040

```

```

SELDO1: .BYTE 15,12,0
        .BYTE 15,12
        .ASCIZ /SET SWITCH TO SELECT DAC 0 AND 1/<15><12>

```

```

SELDO3: .BYTE 15,12
        .ASCIZ /SET SWITCH TO SELECT DAC 2 AND 3/<15><12>

```

```

CALDON: .BYTE 15,12
        .ASCIZ /AUTO CALIBRATION COMPLETED/<15><12>

```

```

MANHED: .BYTE 15,12
        .ASCII /SW 15-13 SELECT THE REGISTER/<15><12>

```

```

        .ASCII /SW 12-00 ARE LOADED INTO THE SELECTED REGISTER/<15><12>

```

```

AUTOCL: .BYTE 0
        .BYTE 15,12
        .ASCII /AUTO CALIBRATION (AA11-K OPTION TEST AREA ONLY)/

```

```

AUTOER: .BYTE 15,12,0
        .ASCII <15><12>/BUS TRAP WHEN LOADING VOLTAGE OR CHANGING A RELAY/<15><12>

```

1934	017470	046105	054501	005015		
	017476	037477	040411	030501	.ASCIZ	/? ? RA11-K OPTION TEST AREA ? ? /<15><12>
	017504	026461	020113	050117		
	017512	044524	047117	052040		
	017520	051505	020124	051101		
	017526	040505	037440	004477		
	017534	005015	000			
1935						
1939	017537	015	040412	045104	R38:	.ASCIZ <15><12>/ADJUST R38 FOR A NULL ON THE METER/
(1)	017544	051525	020124	031522		
(1)	017552	020070	047506	020122		
(1)	017560	020101	052516	046114		
(1)	017566	047440	020116	044124		
(1)	017574	020105	042515	042524		
(1)	017602	000122				
1940	017604	005015	042101	052512	R40:	.ASCIZ <15><12>/ADJUST R40 FOR A NULL ON THE METER/
(1)	017612	052123	051040	030064		
(1)	017620	043040	051117	040440		
(1)	017626	047040	046125	020114		
(1)	017634	047117	052040	042510		
(1)	017642	046440	052105	051105		
(1)	017650	000				
1941	017651	015	040412	045104	R42:	.ASCIZ <15><12>/ADJUST R42 FOR A NULL ON THE METER/
(1)	017656	051525	020124	032122		
(1)	017664	020062	047506	020122		
(1)	017672	020101	052516	046114		
(1)	017700	047440	020116	044124		
(1)	017706	020105	042515	042524		
(1)	017714	000122				
1942	017716	005015	042101	052512	R44:	.ASCIZ <15><12>/ADJUST R44 FOR A NULL ON THE METER/
(1)	017724	052123	051040	032064		
(1)	017732	043040	051117	040440		
(1)	017740	047040	046125	020114		
(1)	017746	047117	052040	042510		
(1)	017754	046440	052105	051105		
(1)	017762	000				
1943	017763	015	040412	045104	R37:	.ASCIZ <15><12>/ADJUST R37 FOR A NULL ON THE METER/
(1)	017770	051525	020124	031522		
(1)	017776	020067	047506	020122		
(1)	020004	020101	052516	046114		
(1)	020012	047440	020116	044124		
(1)	020020	020105	042515	042524		
(1)	020026	000122				
1944	020030	005015	042101	052512	R39:	.ASCIZ <15><12>/ADJUST R39 FOR A NULL ON THE METER/
(1)	020036	052123	051040	034463		
(1)	020044	043040	051117	040440		
(1)	020052	047040	046125	020114		
(1)	020060	047117	052040	042510		
(1)	020066	046440	052105	051105		
(1)	020074	000				
1945	020075	015	040412	045104	R41:	.ASCIZ <15><12>/ADJUST R41 FOR A NULL ON THE METER/
(1)	020102	051525	020124	032122		
(1)	020110	020061	047506	020122		
(1)	020116	020101	052516	046114		

(1)	020124	047440	020116	044124
(1)	020132	020105	042515	042524
(1)	020140	000122		
1946	020142	005015	042101	052512
(1)	020150	052123	051040	031464
(1)	020156	043040	051117	040440
(1)	020164	047040	046125	020114
(1)	020172	047117	052040	042510
(1)	020200	046440	052105	051105
(1)	020206	000		
1947		000001		
1948		000003		
1949	020207	001		
1950	020210	032516	030060	030060
	020216	053060		
1951	020220	003	000	
1952	020222	001		
1953	020223	120	034464	033471
	020230	030066	126	
1954	020233	003	000	
1955		020236		
1956	020236	001116	001444	001124
	020244	001126	000000	
1957	020250	001116	001446	001124
	020256	001126	000000	
1958	020262	001116	001450	001124
	020270	001126	000000	
1959	020274	001116	001452	001124
	020302	001126	000000	
1960	020306	001116	001454	001124
	020314	001126	000000	
1961	020320	001116	001444	000000
1962	020326	001116	001444	020426
	020334	001206	000000	
1963	020340	001116	001444	001124
	020346	001126	000000	
1964	020352	001116	014366	001124
	020360	001126	000000	
1965	020364	000000	000000	000000
	020372	000000	000000	000000
	020400	000000	000000	
1966	020404	000000		
1967	020406	000000		
1968	020410	000010		
1969	020412	000000		
1970	020414	000000		
1971	020416	000000		
1972	020420	000000		
1973	020422	000000		
1974	020424	000000		
1975	020426	000000		
1976	020430	000000		
1977	020432	000000		
1978	020434	000000		

R43: .ASCIZ <15><12>/ADJUST R43 FOR A NULL ON THE METER/

STX=1
ETX=3
N50000: .BYTE STX
.ASCII /N500000V/

P49976: .BYTE ETX,0
.BYTE STX
.ASCII /P499760V/

.BYTE ETX,0
.EVEN

DT1: SERRPC, STAT, SGDDAT, SBDDAT, 0

DT2: SERRPC, DAC0, SGDDAT, SBDDAT, 0

DT3: SERRPC, DAC1, SGDDAT, SBDDAT, 0

DT4: SERRPC, DAC2, SGDDAT, SBDDAT, 0

DT5: SERRPC, DAC3, SGDDAT, SBDDAT, 0

DT6: SERRPC, STAT, 0

DT13: SERRPC, STAT, EVER, SUNIT, 0

DT14: SERRPC, STAT, SGDDAT, SBDDAT, 0

DT16: SERRPC, XDACUT, SGDDAT, SBDDAT, 0

DF0: 0,0,0,0,0,0,0,0

LOW: 0
HIGH: 0
INCR: 10
TIMSV: 0
TICKS: 0
TEMP: 0
TEMP1: 0
BALEV1: 0
BALEV2: 0
EVER: 0
EVER1: 0
OPRIN: 0
P7CNT: 0

; TEMPORARY STORAGE

```

1979 020436 000000
1980 020440 010536
1981 020442 010567
1982 020444 010524
1983 020446 010625
1984 020450 010625
1985 020452 010550
1986 020454 010613
1987 020456 011010
1988 020460 010733
1989 020462 010536
1990 020464 010567
1991 020466 010524
1992 020470 010625
1993 020472 010625
1994 020474 010550
1995 020476 010613
1996 020500 011010
1997 020502 010740
1998
1999

```

```

P7PNT: 0
CH01:  CHANNEL
        SPACEA
        N1
CH02:  CHANNEL
        SPACEA
        N2

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2) 020504
(3) 020504 010046
(3) 020506 010146
(3) 020510 010246
(3) 020512 010346
(3) 020514 010546
(1) 020516 012746 020200
(1) 020522 016605 000020
(1) 020526 100004
(1) 020530 005405
(1) 020532 112766 000055 000001
(1) 020540 005000
(1) 020542 012703 020720
(1) 020546 112723 000040
(1) 020552 005002
(1) 020554 016001 020710
(1) 020560 160105
(1) 020562 002402
(1) 020564 005202
(1) 020566 000774
(1) 020570 060105
(1) 020572 005702

```

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:
*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*      TYPDS      ;;GO TO THE ROUTINE

STYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      R5             ;;BR IF INPUT IS POS.
NEG      R5             ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
MOV      SDBLK(R0),R1    ;;GET THE CONSTANT
3$:      SUB      R1,R5    ;;FORM THIS BCD DIGIT
BLT      R5             ;;BR IF DONE
INC      R2             ;;INCREASE THE BCD DIGIT BY 1
4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT
TST      R2             ;;CHECK IF BCD DIGIT=0

```



```

(1) 020750 013746 000004          MOV    Q#ERRVEC, -(SP)    ;; SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 020754 012737 020774 000004  MOV    #5$, Q#ERRVEC    ;; SET FOR TIMEOUT
(1) 020762 005737 177060          TST    Q#177060         ;; TIME OUT ON XOR?
(1) 020766 012637 000004          MOV    (SP)+, Q#ERRVEC  ;; RESTORE THE ERROR VECTOR
(1) 020772 000446                BR     $SVLAD           ;; GO TO THE NEXT TEST
(1) 020774 022626                5$:   CMP    (SP)+, (SP)+  ;; CLEAR THE STACK AFTER A TIME OUT
(1) 020776 012637 000004          MOV    (SP)+, Q#ERRVEC  ;; RESTORE THE ERROR VECTOR
(1) 021002 000451                BR     $OVER           ;; LOOP ON THE PRESENT TEST
(1) 021004                6$:; #####END OF CODE FOR THE XOR TESTER#####
(1) 021004 032777 000400 160126  BIT    #BIT08, QSWR     ;; LOOP ON SPEC. TEST?
(1) 021012 001404                BEQ    2$              ;; BR IF NO
(1) 021014 127737 160120 001102  CMPB   QSWR, $STNM     ;; ON THE RIGHT TEST?   SWR<7:0>
(1) 021022 001441                BEQ    $OVER          ;; BR IF YES
(1) 021024 105737 001103                2$:   TSTB   $ERFLG     ;; HAS AN ERROR OCCURRED?
(1) 021030 001404                BEQ    3$              ;; BR IF NO
(1) 021032 105037 001103                4$:   CLRB   $ERFLG     ;; ZERO THE ERROR FLAG
(1) 021036 005037 001166                CLR    $TIMES         ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 021042 032777 004000 160070  3$:   BIT    #BIT11, QSWR  ;; INHIBIT ITERATIONS?
(1) 021050 001011                BNE    1$              ;; BR IF YES
(1) 021052 005737 001202                TST    $PASS         ;; IF FIRST PASS OF PROGRAM
(1) 021056 001406                BEQ    1$              ;; INHIBIT ITERATIONS
(1) 021060 005237 001104                INC    $ICNT         ;; INCREMENT ITERATION COUNT
(1) 021064 023737 001166 001104  CMP    $TIMES, $ICNT   ;; CHECK THE NUMBER OF ITERATIONS MADE
(1) 021072 002015                BGE    $OVER          ;; BR IF MORE ITERATION REQUIRED
(1) 021074 012737 000001 001104  1$:   MOV    #1, $ICNT     ;; REINITIALIZE THE ITERATION COUNTER
(1) 021102 013737 021142 001166  MOV    $MXCNT, $TIMES  ;; SET NUMBER OF ITERATIONS TO DO
(1) 021110 105237 001102                $SVLAD: INCB   $STNM     ;; COUNT TEST NUMBERS
(1) 021114 113737 001102 001200  MOVB   $STNM, $TESTN   ;; SET TEST NUMBER IN APT MAILBOX
(1) 021122 011637 001106                MOV    (SP), $LPADR    ;; SAVE SCOPE LOOP ADDRESS
(1) 021126 013777 001102 160006  $OVER: MOV    $STNM, QDISPLAY ;; DISPLAY TEST NUMBER
(1) 021134 013716 001106                MOV    $LPADR, (SP)   ;; FUDGE RETURN ADDRESS
(1) 021140 000002                RTI                    ;; FIXES PS
(1) 021142 003720                $MXCNT: 2000          ;; MAX. NUMBER OF ITERATIONS
2001 .SBTTL ERROR HANDLER ROUTINE
(1)
(2)
(1) ;; *****
(1) ;; *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
(1) ;; *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) ;; *AND GO TO $ERRTYP ON ERROR
(1) ;; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;; *SW15=1          HALT ON ERROR
(1) ;; *SW13=1          INHIBIT ERROR TYPEOUTS
(1) ;; *CALL
(1) ;; *   ERROR   N           ;; ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1) $ERROR:
(1) 021144 104407                CKSWR                ;; TEST FOR CHANGE IN SOFT-SWR
(2) 021146 004737 013246  JSR    PC, ACTRLC
(1) 021152 105237 001103                7$:   INCB   $ERFLG     ;; SET THE ERROR FLAG
(1) 021156 001775                BEQ    7$             ;; DON'T LET THE FLAG GO TO ZERO
(1) 021160 013777 001102 157754  MOV    $STNM, QDISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
(1) 021166 005237 001112                INC    $ERTTL        ;; INC THE ERROR COUNT
(1) 021172 011637 001116                MOV    (SP), $ERRPC  ;; GET ADDRESS OF ERROR INSTRUCTION
(1) 021176 162737 000002 001116  SUB    #2, $ERRPC

```

```

(1) 021204 117737 157706 001114      MOVB    Q$ERRPC,$ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 021212 032777 020000 157720      BIT     #BIT13,Q$SWR     ;;SKIP TYPEOUT IF SET
(1) 021220 001004                      BNE     20$              ;;SKIP TYPEOUTS
(1) 021222 004737 021274              JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1) 021226 104401 001171              TYPE   ,$CRLF
(1) 021232                                20$:
(1) 021232 122737 000001 001214      CMPB   #APTENV,$ENV     ;;RUNNING IN APT MODE
(1) 021240 001007                      BNE     2$              ;;NO SKIP APT ERROR REPORT
(1) 021242 113737 001114 021254      MOVB   $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 021250 004737 023254              JSR     PC,$ATY4       ;;REPORT FATAL ERROR TO APT
(1) 021254 000                      21$:  .BYTE 0
(1) 021255 000                      .BYTE 0
(1) 021256 000777                      22$:  BR    *22$          ;;APT ERROR LOOP
(1) 021260 005777 157654              2$:   TST   Q$SWR      ;;HALT ON ERROR
(1) 021264 100002                      BPL    3$              ;;SKIP IF CONTINUE
(1) 021266 000000                      HALT
(1) 021270 104407                      CKSWR  ;;HALT ON ERROR!
(1) 021272                                3$:   ;;TEST FOR CHANGE IN SOFT-SWR
(1) 021272 000002                      RTI     ;;RETURN
2002 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 021274                                ;;*****
(1) 021274 104401 001171              ;;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1) 021300 010046                      ;;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1) 021302 005000                      ;;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1) 021304 153700 001114              $ERRTYP:
(1) 021310 001004                      TYPE   ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 021312 013746 001116              MOV    RO,-(SP)        ;;SAVE RO
(2) 021316 104402                      CLR    RO              ;;PICKUP THE ITEM INDEX
(1) 021320 000426                      BISB   Q#$ITEMB,RO    ;; IF ITEM NUMBER IS ZERO, JUST
(1) 021322 005300                      BNE    1$              ;;TYPE THE PC OF THE ERROR
(1) 021324 006300                      MOV    $ERRPC,-(SP)   ;;SAVE $ERRPC FOR TYPEOUT
(1) 021326 006300                      TYP0C  ;;ERROR ADDRESS
(1) 021330 006300                      BR     6$              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 021332 062700 001256              1$:   DEC    RO          ;;GET OUT
(1) 021336 012037 021346              ASL   RO              ;;ADJUST THE INDEX SO THAT IT WILL
(1) 021342 001404                      ASL   RO              ;;WORK FOR THE ERROR TABLE
(1) 021344 104401                      ASL   RO
(1) 021346 000000                      ADD    # $ERRTB,RO    ;;FORM TABLE POINTER
(1) 021350 104401 001171              MOV    (RO)+,2$       ;;PICKUP "ERROR MESSAGE" POINTER
(1) 021354 012037 021364              BEQ    3$              ;;SKIP TYPEOUT IF NO POINTER
(1) 021360 001404                      TYPE   "ERROR MESSAGE"
(1) 021362 104401                      2$:   .WORD 0          ;;"ERROR MESSAGE" POINTER GOES HERE
(1) 021364 000000                      TYPE   ,$CRLF         ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 021372 011000                      3$:   MOV    (RO)+,4$    ;;PICKUP "DATA HEADER" POINTER
(1) 021374 001004                      BEQ    5$              ;;SKIP TYPEOUT IF 0
(1)                                4$:   .WORD 0          ;;TYPE THE "DATA HEADER"
(1)                                5$:   .WORD 0          ;;"DATA HEADER" POINTER GOES HERE
(1)                                TYPE   "DATA HEADER"
(1)                                TYPE   ,$CRLF         ;;"CARRIAGE RETURN" & "LINE FEED"
(1)                                MOV    (RO),RO        ;;PICKUP "DATA TABLE" POINTER
(1)                                BNE   7$              ;;GO TYPE THE DATA

```

```

(1) 021376 012600          6$:  MOV    (SP)+,RO      ;;RESTORE RO
(1) 021400 104401 001171  TYPE    ,SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 021404 000207          RTS     PC          ;;RETURN
(2) 021406 013046          7$:  MOV    @ (RO)+, -(SP)  ;;SAVE @ (RO)+ FOR TYPEOUT
(2) 021410 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 021412 005710          TST    (RO)          ;;IS THERE ANOTHER NUMBER?
(1) 021414 001770          BEQ    6$           ;;BR IF NO
(1) 021416 104401 021424  TYPE    ,8$           ;;TYPE TWO(2) SPACES
(1) 021422 000771          BR     7$           ;;LOOP
(1) 021424 020040 000          8$:  .ASCIZ  / /          ;;TWO(2) SPACES
(1) 021430 021430          .SBTTL POWER DOWN AND UP ROUTINES

(1)
(2)
(1)
(1) 021430 012737 021574 000024 $PWRDN: MOV    #SILLUP, @#PWRVEC ;;SET FOR FAST UP
(1) 021436 012737 000340 000026  MOV    #340, @#PWRVEC+2 ;;PRIO:7
(3) 021444 010046          MOV    RO, -(SP)      ;;PUSH RO ON STACK
(3) 021446 010146          MOV    R1, -(SP)     ;;PUSH R1 ON STACK
(3) 021450 010246          MOV    R2, -(SP)     ;;PUSH R2 ON STACK
(3) 021452 010346          MOV    R3, -(SP)     ;;PUSH R3 ON STACK
(3) 021454 010446          MOV    R4, -(SP)     ;;PUSH R4 ON STACK
(3) 021456 010546          MOV    R5, -(SP)     ;;PUSH R5 ON STACK
(3) 021460 017746 157454  MOV    @SWR, -(SP)    ;;PUSH @SWR ON STACK
(1) 021464 010637 021600  MOV    SP, $SAVR6    ;;SAVE SP
(1) 021470 012737 021502 000024  MOV    #PWRUP, @#PWRVEC ;;SET UP VECTOR
(1) 021476 000000          HALT
(1) 021500 000776          BR     .-2          ;;HANG UP

(1)
(2)
(1)
(1) 021502 012737 021574 000024 $PWRUP: MOV    #SILLUP, @#PWRVEC ;;SET FOR FAST DOWN
(1) 021510 013706 021600  MOV    $SAVR6, SP    ;;GET SP
(1) 021514 005037 021600  CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
(1) 021520 005237 021600  1$:  INC    $SAVR6        ;;WAIT FOR THE INC
(1) 021524 001375          BNE    1$           ;;OF WORD
(3) 021526 012677 157406  MOV    (SP)+, @SWR   ;;POP STACK INTO @SWR
(3) 021532 012605          MOV    (SP)+, R5    ;;POP STACK INTO R5
(3) 021534 012604          MOV    (SP)+, R4    ;;POP STACK INTO R4
(3) 021536 012603          MOV    (SP)+, R3    ;;POP STACK INTO R3
(3) 021540 012602          MOV    (SP)+, R2    ;;POP STACK INTO R2
(3) 021542 012601          MOV    (SP)+, R1    ;;POP STACK INTO R1
(3) 021544 012600          MOV    (SP)+, RO    ;;POP STACK INTO RO
(1) 021546 012737 021430 000024  MOV    #PWRDN, @#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 021554 012737 000340 000026  MOV    #340, @#PWRVEC+2 ;;PRIO:7
(1) 021562 104401          TYPE    PWRMSG       ;;REPORT THE POWER FAILURE
(1) 021564 021602          $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
(1) 021566 012716          $PWRAD: MOV    (PC)+, (SP) ;;RESTART AT BEGIN
(1) 021570 001462          $PWRAD: .WORD BEGIN  ;;RESTART ADDRESS
(1) 021572 000002          RTI
(1) 021574 000000          $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
(1) 021576 000776          BR     .-2          ;;BEFORE THE POWER DOWN WAS COMPLETE
(1) 021600 000000          $SAVR6: 0           ;;PUT THE SP HERE

```



```

(1) 022112 017600 000002      MOV      02(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
(1) 022116 122737 000001 001214  CMPB    #APTENV,$ENV    ;; RUNNING IN APT MODE
(1) 022124 001011          BNE     62$           ;; NO, GO CHECK FOR APT CONSOLE
(1) 022126 132737 000100 001215  BITB    #APTSPool,$ENVM ;; SPOOL MESSAGE TO APT
(1) 022134 001405          BEQ     62$           ;; NO, GO CHECK FOR CONSOLE
(1) 022136 010037 022146      MOV     RO,61$        ;; SETUP MESSAGE ADDRESS FOR APT
(1) 022142 004737 023244      JSR    PC,$ATY3      ;; SPOOL MESSAGE TO APT
(1) 022146 000000          .WORD   0            ;; MESSAGE ADDRESS
(1) 022150 132737 000040 001215 61$:   BITB    #APTC SUP,$ENVM ;; APT CONSOLE SUPPRESSED
(1) 022156 001003          BNE     60$           ;; YES, SKIP TYPE OUT
(1) 022160 112046          2$:     MOVB    (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 022162 001005          BNE     4$            ;; BR IF IT ISN'T THE TERMINATOR
(1) 022164 005726          TST    (SP)+         ;; IF TERMINATOR POP IT OFF THE STACK
(1) 022166 012600          60$:   MOV     (SP)+,RO     ;; RESTORE RO
(1) 022170 062716 000002          3$:   ADD     #2,(SP)     ;; ADJUST RETURN PC
(1) 022174 000002          RTI                    ;; RETURN
(1) 022176 122716 000011          4$:   CMPB    #HT,(SP)    ;; BRANCH IF <HT>
(1) 022202 001430          BEQ     8$            ;;
(1) 022204 122716 000200          CMPB    #CRLF,(SP)   ;; BRANCH IF NOT <CRLF>
(1) 022210 001006          BNE     5$            ;;
(1) 022212 005726          TST    (SP)+         ;; POP <CR><LF> EQUIV
(1) 022214 104401          TYPE   $CRLF        ;; TYPE A CR AND LF
(1) 022216 001171          $CRLF
(1) 022220 105037 022354          CLRB   $CHARCNT     ;; CLEAR CHARACTER COUNT
(1) 022224 000755          BR     2$            ;; GET NEXT CHARACTER
(1) 022226 004737 022310          5$:   JSR    PC,$TYPEC    ;; GO TYPE THIS CHARACTER
(1) 022232 123726 001156          6$:   CMPB    $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
(1) 022236 001350          BNE     2$            ;; IF NO GO GET NEXT CHAR.
(1) 022240 013746 001154          MOV     $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
(1)                                ;; AND THE NULL CHAR.
(1) 022244 105366 000001          7$:   DECB   1(SP)       ;; DOES A NULL NEED TO BE TYPED?
(1) 022250 002770          BLT    6$            ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 022252 004737 022310          JSR    PC,$TYPEC    ;; GO TYPE A NULL
(1) 022256 105337 022354          DECB   $CHARCNT     ;; DO NOT COUNT AS A COUNT
(1) 022262 000770          BR     7$           ;; LOOP
(1)
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1) 022264 112716 000040          8$:   MOVB    #' ,(SP)    ;; REPLACE TAB WITH SPACE
(1) 022270 004737 022310          9$:   JSR    PC,$TYPEC    ;; TYPE A SPACE
(1) 022274 132737 000007 022354  BITB    #7,$CHARCNT  ;; BRANCH IF NOT AT
(1) 022302 001372          BNE     9$           ;; TAB STOP
(1) 022304 005726          TST    (SP)+         ;; POP SPACE OFF STACK
(1) 022306 000724          BR     2$            ;; GET NEXT CHARACTER
(1) 022310 105777 156634          $TYPEC: TSTB   $STPB       ;; WAIT UNTIL PRINTER IS READY
(1) 022314 100375          BPL    $TYPEC
(1) 022316 116677 000002 156626  MOVB    2(SP), $STPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 022324 122766 000015 000002  CMPB    #CR,2(SP)    ;; IS CHARACTER A CARRIAGE RETURN?
(1) 022332 001003          BNE     1$           ;; BRANCH IF NO
(1) 022334 105037 022354          CLRB   $CHARCNT     ;; YES--CLEAR CHARACTER COUNT
(1) 022340 000406          BR     $TYPEX
(1) 022342 122766 000012 000002 1$:   CMPB    #LF,2(SP)   ;; IS CHARACTER A LINE FEED?
(1) 022350 001402          BEQ    $TYPEX       ;; BRANCH IF YES
(1) 022352 105227          INCB   (PC)+        ;; COUNT THE CHARACTER

```

```

(1) 022354 000000 $CHARCNT: WORD 0 ;; CHARACTER COUNT STORAGE
(1) 022356 000207 $TYPEX: RTS PC
2009 .SBTTL TTY INPUT ROUTINE
(1) ;;*****
(2) .ENABL LSB
(1) ;;*****
(1) *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) *WHEN OPERATING IN TTY FLAG MODE.
(1) 022360 022737 000176 001140 $CKSWR: CMP #SWREG, SWR ;; IS THE SOFT-SWR SELECTED?
(1) 022366 001074 BNE 15$ ;; BRANCH IF NO
(1) 022370 105777 156550 TSTB @STKS ;; CHAR THERE?
(1) 022374 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
(1) 022376 117746 156544 MOVB @STKB, -(SP) ;; SAVE THE CHAR
(1) 022402 042716 177600 BIC #177, (SP) ;; STRIP-OFF THE ASCII
(1) 022406 022726 000007 CMP #7, (SP)+ ;; IS IT A CONTROL G?
(1) 022412 001062 BNE 15$ ;; NO RETURN TO USER
(1) 022414 123727 001134 000001 CMPB $AUTOB, #1 ;; ARE WE RUNNING IN AUTO-MODE?
(1) 022422 001456 BEQ 15$ ;; BRANCH IF YES
(1) 022424 104401 023105 $GTSWR: TYPE , $CNTLG ;; ECHO THE CONTROL-G (↑G)
(1) 022430 104401 023112 TYPE , $MSWR ;; TYPE CURRENT CONTENTS
(2) 022434 013746 000176 MOV SWREG, -(SP) ;; SAVE SWREG FOR TYPEOUT
(2) 022440 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 022442 104401 023123 TYPE , $MNEW ;; PROMPT FOR NEW SWR
(1) 022446 005046 19$: CLR -(SP) ;; CLEAR COUNTER
(1) 022450 005046 CLR -(SP) ;; THE NEW SWR
(1) 022452 105777 156466 7$: TSTB @STKS ;; CHAR THERE?
(1) 022456 100375 BPL 7$ ;; IF NOT TRY AGAIN
(1) 022460 117746 156462 MOVB @STKB, -(SP) ;; PICK UP CHAR
(1) 022464 042716 177600 BIC #177, (SP) ;; MAKE IT 7-BIT ASCII
(1)
(1)
(1) 022470 021627 000025 9$: CMP (SP), #25 ;; IS IT A CONTROL-U?
(1) 022474 001005 BNE 10$ ;; BRANCH IF NOT
(1) 022476 104401 023100 TYPE , $CNTLU ;; YES, ECHO CONTROL-U (↑U)
(1) 022502 062706 000006 20$: ADD #6, SP ;; IGNORE PREVIOUS INPUT
(1) 022506 000757 BR 19$ ;; LET'S TRY IT AGAIN
(1)
(1)
(1) 022510 021627 000015 10$: CMP (SP), #15 ;; IS IT A <CR>?
(1) 022514 001022 BNE 16$ ;; BRANCH IF NO
(1) 022516 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
(1) 022522 001403 BEQ 11$ ;; BRANCH IF YES
(1) 022524 016677 000002 156406 MOV 2(SP), @SWR ;; SAVE NEW SWR
(1) 022532 062706 000006 11$: ADD #6, SP ;; CLEAR UP STACK
(1) 022536 104401 001171 14$: TYPE , $CALF ;; ECHO <CR> AND <LF>
(1) 022542 123727 001135 000001 CMPB $INTAG, #1 ;; RE-ENABLE TTY KBD INTERRUPTS?

```



```

(1)          ;*      RDLIN          ;: INPUT A STRING FROM THE TTY
(1)          ;*      RETURN HERE      ;: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)          ;*                               ;: TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) 022762 010346          SRDLIN: MOV      R3, -(SP)      ;: SAVE R3
(1) 022764 012703 023070 1$:      MOV      #STTYIN, R3      ;: GET ADDRESS
(1) 022770 022703 023100 2$:      CMP      #STTYIN+8., R3      ;: BUFFER FULL?
(1) 022774 101405          BLOS      4$              ;: BR IF YES
(1) 022776 104410          RDCHR          ;: GO READ ONE CHARACTER FROM THE TTY
(1) 023000 112613          MOVB     (SP)+, (R3)      ;: GET CHARACTER
(1) 023002 122713 000177 10$:     CMPB     #177, (R3)      ;: IS IT A RUBOUT
(1) 023006 001003          BNE      3$              ;: SKIP IF NOT
(1) 023010 104401 001170 4$:      TYPE     $QUES      ;: TYPE A '?'
(1) 023014 000763          BR       1$              ;: CLEAR THE BUFFER AND LOOP
(1) 023016 111337 023066 3$:      MOVB     (R3), 9$      ;: ECHO THE CHARACTER
(1) 023022 104401 023066          TYPE     9$              ;:
(1) 023026 122723 000015          CMPB     #15, (R3)+      ;: CHECK FOR RETURN
(1) 023032 001356          BNE      2$              ;: LOOP IF NOT RETURN
(1) 023034 105063 177777          CLRB     -1(R3)         ;: CLEAR RETURN (THE 15)
(1) 023040 104401 001172          TYPE     $LF           ;: TYPE A LINE FEED
(1) 023044 012603          MOV      (SP)+, R3      ;: RESTORE R3
(1) 023046 011646          MOV      (SP), -(SP)     ;: ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 023050 016666 000004 000002 MOV      4(SP), 2(SP)    ;: FIRST ASCII CHARACTER ON IT
(1) 023056 012766 023070 000004 MOV      #STTYIN, 4(SP)
(1) 023064 000002          RTI                    ;: RETURN
(1) 023066 000          9$:      .BYTE     0              ;: STORAGE FOR ASCII CHAR. TO TYPE
(1) 023067 000          .BYTE     0              ;: TERMINATOR
(1) 023070 000010          .BLKB     8.           ;: RESERVE 8 BYTES FOR TTY INPUT
(1) 023100 052536 005015 000 $CNTLU: .ASCIZ  /?U/<15><12> ;: CONTROL "U"
(1) 023105 136 006507 000012 $CNTLG: .ASCIZ  /?G/<15><12> ;: CONTROL "G"
(1) 023112 005015 053523 020122 $MSWR: .ASCIZ  <15><12>/SWR = /
(1) 023120 020075 000
(1) 023123 040 047040 053505 $MNEW: .ASCIZ  / NEW = /
(1) 023130 036440 000040
2010          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1)          ;: *****
(2)          ;: THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)          ;: CHANGE IT TO BINARY.
(1)          ;: CALL:
(1)          ;*      RDOCT          ;: READ AN OCTAL NUMBER
(1)          ;*      RETURN HERE      ;: LOW ORDER BITS ARE ON TOP OF THE STACK
(1)          ;*                               ;: HIGH ORDER BITS ARE IN $HIOCT
(1) 023134 011646          SRDOCT: MOV      (SP), -(SP)      ;: PROVIDE SPACE FOR THE
(1) 023136 016666 000004 000002 MOV      4(SP), 2(SP)      ;: INPUT NUMBER
(3) 023144 010046          MOV      R0, -(SP)       ;: PUSH R0 ON STACK
(3) 023146 010146          MOV      R1, -(SP)       ;: PUSH R1 ON STACK
(3) 023150 010246          MOV      R2, -(SP)       ;: PUSH R2 ON STACK
(1) 023152 104411 1$:      RDLIN          ;: READ AN ASCII LINE
(1) 023154 012600          MOV      (SP)+, R0      ;: GET ADDRESS OF 1ST CHARACTER
(1) 023156 005001          CLR      R1              ;: CLEAR DATA WORD
(1) 023160 005002          CLR      R2              ;:
(1) 023162 112046          2$:      MOVB     (R0)+, -(SP)      ;: PICKUP THIS CHARACTER

```

```

(1) 023164 001412 BEQ 3$ ;; IF ZERO GET OUT
(1) 023166 006301 ASL R1 ;; *2
(1) 023170 006102 ROL R2
(1) 023172 006301 ASL R1 ;; *4
(1) 023174 006102 ROL R2
(1) 023176 006301 ASL R1 ;; *8
(1) 023200 006102 ROL R2
(1) 023202 042716 177770 BIC #C7,(SP) ;; STRIP THE ASCII JUNK
(1) 023206 062601 ADD (SP)+,R1 ;; ADD IN THIS DIGIT
(1) 023210 000764 BR 2$ ;; LOOP
(1) 023212 005726 3$: TST (SP)+ ;; CLEAN TERMINATOR FROM STACK
(1) 023214 010166 000012 MOV R1,12(SP) ;; SAVE THE RESULT
(1) 023220 010237 023234 MOV R2,$HIOCT
(3) 023224 012602 MOV (SP)+,R2 ;; POP STACK INTO R2
(3) 023226 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
(3) 023230 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
(1) 023232 000002 RTI ;; RETURN
(1) 023234 000000 SHIOCT: .WORD 0 ;; HIGH ORDER BITS GO HERE
2011 .SBTTL APT COMMUNICATIONS ROUTINE
(1)
(2)
(1) 023236 112737 000001 023502 $ATY1: MOVB #1,$FFLG ;; TO REPORT FATAL ERROR
(1) 023244 112737 000001 023500 $ATY3: MOVB #1,$MFLG ;; TO TYPE A MESSAGE
(1) 023252 000403 BR $ATYC
(1) 023254 112737 000001 023502 $ATY4: MOVB #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
(2) 023262 $ATYC:
(3) 023262 010046 MOV R0,-(SP) ;; PUSH R0 ON STACK
(3) 023264 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
(1) 023266 105737 023500 TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
(1) 023272 001450 BEQ 5$ ;; IF NOT: BR
(1) 023274 122737 000001 001214 CMPB #APTENV,$ENV ;; OPERATING UNDER APT?
(1) 023302 001031 BNE 3$ ;; IF NOT: BR
(1) 023304 132737 000100 001215 BITB #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
(1) 023312 001425 BEQ 3$ ;; IF NOT: BR
(1) 023314 017600 000004 MOV #4(SP),R0 ;; GET MESSAGE ADDR.
(1) 023320 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
(1) 023326 005737 001174 1$: TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
(1) 023332 001375 BNE 1$ ;; IF NOT: WAIT
(1) 023334 010037 001210 MOV R0,$MSGAD ;; PUT ADDR IN MAILBOX
(1) 023340 105720 2$: TSTB (R0)+ ;; FIND END OF MESSAGE
(1) 023342 001376 BNE 2$
(1) 023344 163700 001210 SUB $MSGAD,R0 ;; SUB START OF MESSAGE
(1) 023350 006200 ASR R0 ;; GET MESSAGE LNTH IN WORDS
(1) 023352 010037 001212 MOV R0,$MSGGLT ;; PUT LENGTH IN MAILBOX
(1) 023356 012737 000004 001174 MOV #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
(1) 023364 000413 BR 5$
(1) 023366 017637 000004 023412 3$: MOV #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
(1) 023374 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDRESS
(3) 023402 013746 177776 MOV 177776,-(SP) ;; PUSH 177776 ON STACK
(1) 023406 004737 022076 JSR PC,$TYPE ;; CALL TYPE MACRO
(1) 023412 000000 4$: .WORD 0
(1) 023414 5$:
(1) 023414 105737 023502 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
(1) 023420 001416 BEQ 12$ ;; IF NOT: BR

```

```

(1) 023422 005737 001214          TST      $ENV          ;;RUNNING UNDER APT?
(1) 023426 001413                    BEQ      12$          ;;IF NOT: BR
(1) 023430 005737 001174          11$:    TST      $MSGTYPE  ;;FINISHED LAST MESSAGE?
(1) 023434 001375                    BNE      11$          ;;IF NOT: WAIT
(1) 023436 017637 000004 001176  MOV      24(SP), $FATAL ;;GET ERROR #
(1) 023444 062766 000002 000004  ADD      #2, 4(SP)      ;;BUMP RETURN ADDR.
(1) 023452 005237 001174          INC      $MSGTYPE      ;;TELL APT TO TAKE ERROR
(1) 023456 105037 023502          12$:    CLRB     $FFLG      ;;CLEAR FATAL FLAG
(1) 023462 105037 023501          CLRB     $LFLG         ;;CLEAR LOG FLAG
(1) 023466 105037 023500          CLRB     $MFLG         ;;CLEAR MESSAGE FLAG
(3) 023472 012601                    MOV      (SP)+, R1     ;;POP STACK INTO R1
(3) 023474 012600                    MOV      (SP)+, R0     ;;POP STACK INTO R0
(1) 023476 000207                    RTS      PC            ;;RETURN
(1) 023500          000          $MFLG: .BYTE 0        ;;MESSG. FLAG
(1) 023501          000          $LFLG: .BYTE 0        ;;LOG FLAG
(1) 023502          000          $FFLG: .BYTE 0        ;;FATAL FLAG
(1)          023504          .EVEN
(1)          000200          APTSIZE=200
(1)          000001          APTENV=001
(1)          000100          APTSPOOL=100
(1)          000040          APTCSUP=040

2012
2013
(1)          .SBTTL TRAP DECODER
(2)          ;;*****
(1)          ;;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)          ;;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)          ;;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)          ;;*GO TO THAT ROUTINE.
(1) 023504 010046                    STRAP:  MOV      R0, -(SP) ;;SAVE R0
(1) 023506 016600 000002          MOV      2(SP), R0     ;;GET TRAP ADDRESS
(1) 023512 005740                    TST      -(R0)         ;;BACKUP BY 2
(1) 023514 111000                    MOVB     (R0), R0      ;;GET RIGHT BYTE OF TRAP
(1) 023516 006300                    ASL      R0            ;;POSITION FOR INDEXING
(1) 023520 016000 023540          MOV      $TRAPAD(R0), R0 ;;INDEX TO TABLE
(1) 023524 000200                    RTS      R0            ;;GO TO ROUTINE

(1)          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1) 023526 011646                    STRAP2: MOV      (SP), -(SP) ;;MOVE THE PC DOWN
(1) 023530 016666 000004 000002  MOV      4(SP), 2(SP)  ;;MOVE THE PSW DOWN
(1) 023536 000002                    RTI                    ;;RESTORE THE PSW

(3)          .SBTTL TRAP TABLE
(3)          ;;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;;*BY THE "TRAP" INSTRUCTION.
(3)          :
(3)          ROUTINE
(3)          -----
(3) 023540 023526                    $TRPAD: .WORD  $STRAP2
(3) 023542 022076                    $TYPE  :;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE

```

(3)	023544	021674	\$TYPOC	::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)	023546	021650	\$TYPOS	::CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)	023550	021710	\$TYPON	::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)	023552	020504	\$TYPDS	::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
(1)						
(3)	023554	022430	\$GTSWR	::CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
(1)						
(3)	023556	022360	\$CKSWR	::CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
(3)	023560	022642	\$RDCHR	::CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
(3)	023562	022762	\$RDLIN	::CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
(3)	023564	023134	\$RDOCT	::CALL=RDOCT	TRAP+12(104412)	READ AN OCTAL NUMBER FROM TTY
2014						
2015	023566	000074	BUF2:	.BLKW	60.	
2016	023756	000074	BUF3:	.BLKW	60.	
2017	024146	000074	BUFFER:	60.		
2018						
2019	024150	000240		NOP		
2020		000001		.END		

TRTVEC=	000014	14#			
TST1	002470	240	272#		
TST10	003056	338#			
TST11	003112	343	346#		
TST12	003150	351	354#		
TST13	003222	366#			
TST14	003256	371	374#		
TST15	003314	379	382#		
TST16	003366	392#			
TST17	003554	419	422#		
TST2	002546	285#			
TST20	003614	427	430#		
TST21	003704	441	444#		
TST22	003774	455	460#		
TST23	004064	472	475#		
TST24	004154	486	489#		
TST25	004212	494	497#		
TST26	004250	502	505#		
TST27	004306	510	513#		
TST3	002602	290	293#		
TST30	004344	518	522#		
TST31	004406	528	532#		
TST32	004462	540	543#		
TST33	004552	552	554	561#	
TST34	004700	571	580	583	592#
TST35	005024	602	609	611	618#
TST36	005124	620	628	630	639#
TST37	005220	654#			
TST4	002640	298	301#		
TST40	005340	665	675#		
TST41	005406	681	683#		
TST42	005454	689	691#		
TST43	005530	692	698	700#	
TST44	005604	701	707	710#	
TST45	005652	711	716	718#	
TST46	005720	719	724	738#	
TST47	005776	738	739#		
TST5	002712	311#			
TST50	006054	739	740#		
TST51	006132	740	741#		
TST52	006210	741	742#		
TST53	006300	744	752	759#	
TST54	006370	761	775#		
TST55	011572	1306#			
TST56	011634	1316#			
TST57	011676	1323	1326#		
TST6	002746	316	319#		
TST60	011742	1333	1336#		
TST61	012012	1344	1348#		
TST62	012056	1355	1357#		
TST63	012120	1364	1366#		
TST64	012164	1373	1375#		
TST65	012266	1391#			
TST66	012332	1398	1402#		

ADC	1841														
ADD	156	163	233	780	781	782	783	784	785	786	857	893	901	954	955
	968	1023	1106	1110	1113	1183	1197	1239	1764	1772	1834	1868	1999	2002	2007
	2008	2009	2010	2011											
ASL	1771	1789	2002	2009	2010	2013									
ASLB	1999														
ASR	308	334	361	389	1837	1838	1839	1840	2011						
BCC	1999														
BEQ	136	199	201	203	205	207	209	255	290	298	306	316	324	332	343
	351	359	371	379	387	401	407	413	419	427	435	441	449	455	465
	472	480	486	494	502	510	518	528	540	620	681	689	698	707	716
	724	738	739	740	741	761	779	791	823	837	848	876	882	891	933
	939	949	986	989	996	1044	1048	1057	1066	1090	1170	1176	1190	1226	1381
	1387	1421	1429	1654	1660	1683	1722	1788	2000	2001	2002	2007	2008	2009	2010
	2011														
BGE	1344	1355	1364	1409	2000										
BGT	791	1999	2007	2009											
BIC	161	165	196	791	1086	1425	1550	1584	1596	1728	1770	2007	2009	2010	
BIS	252	624	849	992	950	987	1091	1094	1227	1251	1383	1424	1656	1732	1742
	1999	2007	2009												
BISB	1736	2002													
BIT	619	692	701	711	719	738	739	740	741	743	760	822	836	847	853
	875	881	890	932	938	948	985	988	995	1001	1044	1047	1056	1065	1089
	1170	1175	1189	1225	1653	1659	1668	1682	1760	2000	2001				
BITB	136	2008	2011												
BLE	1333	1373	1398	1865											
BLOS	2009														
BLT	1999	2007	2008	2009											
BMI	242	554	568	583	599	611	630	749	766	1662	1999				
BNE	136	148	153	159	187	189	238	309	335	362	390	556	585	613	632
	634	646	661	692	701	711	719	744	751	768	811	854	862	899	906
	914	922	957	970	1002	1013	1025	1079	1108	1112	1185	1199	1241	1551	1555
	1559	1563	1571	1575	1586	1598	1601	1664	1667	1669	1698	1727	1731	1738	1747
	1749	1761	1763	1836	1999	2000	2001	2002	2003	2007	2008	2009	2011		
BPL	550	578	607	626	807	860	896	904	912	920	953	959	966	1022	1093
	1100	1105	1582	1594	1641	1674	1696	1702	1814	1833	1862	1999	2001	2007	2008
	2009														
BR	132	136	212	224	235	240	253	280	552	571	580	602	609	629	649
	662	665	670	752	630	844	845	866	885	924	942	972	992	1007	1015
	1016	1073	1201	1202	1210	1236	1313	1323	1553	1557	1561	1565	1576	1605	1621
	1633	1686	1692	1733	1790	1809	1999	2000	2001	2002	2003	2007	2008	2009	2010
	2011														
CLR	129	130	131	136	137	144	223	230	257	286	312	339	367	397	437
	451	468	482	546	574	604	622	647	652	663	667	668	673	694	703
	713	721	787	791	846	856	864	865	870	886	887	927	943	944	962
	980	981	982	983	984	1052	1061	1070	1238	1337	1358	1389	1412	1416	1417
	1437	1445	1484	1492	1494	1572	1631	1658	1679	1745	1829	1999	2000	2002	2003
	2007	2009	2010												
CLRB	525	537	1999	2000	2008	2009	2011								
CMP	136	147	152	158	198	200	202	204	206	208	236	281	289	297	305
	315	323	331	342	350	358	370	378	386	400	406	412	419	426	434
	440	448	454	464	471	479	485	493	501	509	517	527	539	650	671
	680	688	697	706	738	739	740	741	861	1332	1343	1354	1363	1372	1380
	1386	1397	1408	1420	1428	1534	1554	1558	1562	1585	1597	1600	1637	1774	1864

	1999	2000	2009	2001	2008	2009	2011								
CMPB	254	778	2000												
COM	1723														
DEC	555	584	612	631	633	645	660	750	767	791	897	905	913	921	956
	969	1024	1184	1198	1570	1574	1640	1663	1666	1726	1730	1737	1745	1748	1782
	1787	1835	2002												
DECB	2007	2008													
EMT	14														
HALT	23	1675	2001	2003	2008										
INC	234	536	777	791	858	894	902	910	918	951	964	1020	1078	1103	1107
	1111	1378	1694	1831	1999	2000	2001	2003	2007	2009	2011				
INCB	746	764	816	2000	2001	2008									
IOY	14														
JMP	23	24	25	138	214	215	216	217	218	219	788	791	1074	1213	1532
	1538	1587	1599	1775											
JSR	185	258	791	808	827	828	829	841	842	843	872	923	929	971	978
	1003	1005	1008	1010	1014	1043	1053	1062	1071	1072	1077	1169	1182	1196	1200
	1207	1208	1209	1211	1224	1229	1231	1233	1235	1242	1250	1303	1308	1312	1318
	1322	1329	1340	1351	1360	1369	1394	1405	1435	1438	1441	1444	1448	1451	1454
	1457	1460	1464	1467	1472	1482	1485	1489	1493	1497	1500	1503	1507	1511	1515
	1520	1526	1543	1546	1613	1615	1626	1628	1630	1632	1665	1700	1757	1804	1808
	2000	2001	2008	2009	2011										
MOV	133	136	141	142	143	145	150	151	154	155	160	162	166	192	195
	229	231	245	251	273	274	283	287	288	294	295	296	301	302	303
	304	313	314	320	321	322	327	328	329	330	340	341	347	348	349
	354	355	356	357	368	369	375	376	377	382	383	384	385	393	394
	395	396	398	399	404	405	410	411	416	417	422	423	425	431	432
	433	438	439	445	446	447	452	453	461	462	463	469	470	476	477
	478	483	484	490	491	492	498	499	500	506	507	508	514	515	516
	523	524	526	533	534	535	538	545	547	548	557	558	563	564	565
	566	573	575	586	587	594	595	596	597	603	605	614	615	619	621
	623	639	640	641	642	643	644	651	654	655	656	657	658	659	666
	672	675	676	677	679	683	684	685	687	691	693	696	700	702	705
	710	712	715	718	720	723	738	739	740	741	742	745	747	753	754
	759	762	763	769	770	775	791	805	820	821	824	825	826	834	835
	838	839	840	850	851	852	855	871	873	874	877	878	879	880	893
	884	889	900	908	916	928	930	931	934	935	936	937	940	941	947
	961	963	977	990	991	993	994	997	998	999	1000	1018	1019	1042	1044
	1045	1046	1049	1050	1051	1054	1055	1058	1059	1060	1063	1064	1067	1068	1069
	1075	1076	1085	1087	1088	1095	1096	1097	1101	1102	1109	1168	1170	1171	1172
	1173	1174	1177	1178	1179	1180	1181	1186	1187	1188	1191	1192	1193	1194	1195
	1206	1223	1228	1230	1232	1234	1249	1259	1260	1261	1262	1307	1310	1311	1317
	1320	1321	1327	1328	1331	1338	1339	1342	1349	1350	1353	1359	1362	1367	1368
	1371	1376	1377	1379	1394	1385	1392	1393	1396	1403	1404	1407	1418	1419	1426
	1427	1433	1453	1461	1477	1502	1510	1535	1542	1548	1549	1552	1556	1560	1564
	1566	1568	1569	1573	1583	1595	1612	1616	1617	1618	1619	1620	1625	1629	1635
	1636	1637	1638	1639	1655	1657	1680	1681	1684	1685	1687	1688	1689	1690	1691
	1720	1725	1729	1735	1741	1743	1744	1758	1773	1786	1799	1800	1801	1802	1807
	1825	1826	1828	1830	1842	1857	1858	1859	1860	1866	1867	1871	1872	1899	2000
	2001	2002	2003	2007	2008	2009	2010	2011	2013						
MOV8	164	1098	1721	1724	1999	2000	2001	2007	2008	2009	2010	2011	2013		
NEG	1863	1999	2007												
NOP	260	577	791	1044	1170	1244	1245	1816	1817	1818	2019				
RESET	134	424	678	686	695	704	714	722	738	739	740	741	773	776	791

ROL	1212														
ROLB	2007	2010													
RTI	1099														
RTS	136	1999	2000	2001	2003	2007	2008	2009	2010	2013					
	168	863	1027	1080	1114	1243	1588	1602	1642	1676	1703	1739	1750	1765	1791
	1815	1843	1869	1873	2002	2008	2011	2013							
SUB	167	909	917	967	1693	1701	1812	1813	1861	1999	2001	2011			
SWAB	1827														
TRAP	2013														
TST	146	157	186	188	232	237	241	275	276	277	278	279	806	1012	1026
	1240	1673	1999	2000	2001	2002	2007	2008	2009	2010	2011	2013			
TSTB	549	553	567	576	582	598	606	610	625	629	748	765	810	859	895
	903	911	919	952	958	965	1021	1092	1104	1581	1593	1661	1695	1832	1999
.ASCII	2000	2008	2009	2011											
	30	178	179	180	181	182	183	1878	1879	1880	1881	1882	1883	1884	1885
.ASCIZ	1889	1890	1891	1892	1927	1928	1931	1933	1950	1953	1898	1899	1900	1901	1902
	30	791	1886	1887	1888	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902
	1903	1904	1905	1906	1907	1908	1909	1910	1911	1912	1917	1918	1921	1923	1925
	1934	1939	1940	1941	1942	1943	1944	1945	1946	2002	2004	2009			
.BLKB	2009														
.BLKW	1999	2015	2016												
.BYTE	30	247	248	791	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150
	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165
	1916	1919	1920	1922	1924	1926	1929	1930	1932	1949	1951	1952	1954	2001	2007
.DSABL	2009	2011													
.ENABL	2009														
.END	4	2009													
.ENDC	2020														
	13	14	21	23	27	29	30	32	135	136	235	240	253	255	272
	280	285	290	293	298	301	306	311	316	319	324	327	332	338	343
	346	351	354	359	366	371	374	379	382	387	392	401	407	413	419
	422	427	430	435	441	444	449	455	460	465	472	475	480	486	489
	494	497	502	505	510	513	518	522	528	532	540	543	550	552	554
	561	568	571	578	580	583	585	592	599	602	607	609	611	613	618
	620	626	628	630	632	634	639	649	654	662	665	670	675	681	683
	689	691	692	698	700	701	707	710	711	716	718	719	724	738	739
	740	741	742	744	752	759	761	775	779	791	1306	1313	1316	1323	1326
	1333	1336	1344	1348	1355	1357	1364	1366	1373	1375	1381	1387	1391	1398	1402
	1409	1415	1421	1429	1433	1477	1809	1814	1999	2000	2001	2002	2003	2007	2008
.EQUIV	2009	2010	2011	2013											
.EVEN	14														
.IF	30	1165	1955	2002	2005	2011									
	13	14	21	23	27	29	30	32	135	136	235	240	253	255	272
	280	285	290	293	298	301	306	311	316	319	324	327	332	338	343
	346	351	354	359	366	371	374	379	382	387	392	401	407	413	419
	422	427	430	435	441	444	449	455	460	465	472	475	480	486	489
	494	497	502	505	510	513	518	522	528	532	540	543	550	552	554
	561	568	571	578	580	583	585	592	599	602	607	609	611	613	618
	620	626	628	630	632	634	639	649	654	662	665	670	675	681	683
	689	691	692	698	700	701	707	710	711	716	718	719	724	738	739
	740	741	742	744	752	759	761	775	779	791	1306	1313	1316	1323	1326
	1333	1336	1344	1348	1355	1357	1364	1366	1373	1375	1381	1387	1391	1398	1402
	1409	1415	1421	1429	1433	1477	1809	1814	1999	2000	2001	2002	2003	2007	2008

	2009	2010	2011	2013	30	136	235	240	253	255	272	280	285	290	293
.IFF	14	21	27	29	316	319	324	327	332	338	343	346	351	354	359
	298	301	306	311	382	387	392	401	407	413	419	422	427	430	435
	366	371	374	379	460	465	472	475	480	486	489	494	497	502	505
	441	444	449	455	528	532	540	543	550	552	554	561	568	571	578
	510	513	518	522	599	602	607	609	611	613	618	620	626	628	630
	580	583	585	592	654	662	665	670	675	681	683	689	691	692	698
	632	634	639	649	711	716	718	719	724	738	739	740	741	742	744
	700	701	707	710	779	791	1306	1313	1316	1323	1326	1333	1336	1344	1348
	752	759	761	775	1373	1375	1381	1387	1391	1398	1402	1409	1415	1421	1429
	1355	1357	1364	1366	1999	2000	2001	2002	2003	2007	2008	2009	2010	2011	2013
.IFT	2000	2001	2009	2010											
.IFTF	2000	2001	2009	2010											
.IIF	13	21	23	30	136	791	2000	2001	2002	2008	2009	2013			
.IRP	32	135	272	285	293	301	311	319	327	338	346	354	366	374	382
	392	422	430	444	460	475	489	497	505	513	522	532	543	561	592
	618	639	654	675	683	691	700	710	718	738	739	740	741	742	759
	775	1306	1316	1326	1336	1348	1357	1366	1375	1391	1402	1415	1433	1477	1999
.LIST	2000	2001	2003	2010	2011										
	2	12	14	21	23	30	32	135	136	174	272	285	293	301	311
	319	327	338	346	354	366	374	382	392	422	430	444	460	475	489
	497	505	513	522	532	543	561	592	618	639	654	675	683	691	700
	710	718	738	739	740	741	742	759	775	791	804	819	933	1039	1220
	1301	1306	1316	1326	1336	1348	1357	1366	1375	1391	1402	1415	1433	1477	1648
.MACRO	2000	2001	2009	2013											
.MCALL	21	30	136	261	264	268	726	1030	1936	2013					
.MEXIT	7	8	9	10	11	14	30	136							
.NLIST	30														
	1	3	14	21	23	30	32	135	136	169	272	285	293	301	311
	319	327	338	346	354	366	374	382	392	422	430	444	460	475	489
	497	505	513	522	532	543	561	592	618	639	654	675	683	691	700
	710	718	738	739	740	741	742	759	775	791	793	817	831	1029	1215
	1263	1306	1316	1326	1336	1348	1357	1366	1375	1391	1402	1415	1433	1477	1644
.PAGE	2000	2001	2009	2013											
.REPT	30														
.SBTTL	23	30													
	14	21	23	27	29	30	136	140	149	170	171	172	173	177	227
	272	285	293	301	311	319	327	338	346	354	366	374	382	392	422
	430	444	460	475	489	497	505	513	522	532	543	561	592	618	639
	654	675	683	691	700	710	718	738	739	740	741	742	759	775	791
	794	795	796	797	798	799	800	801	802	803	818	832	868	925	975
	1040	1166	1204	1216	1217	1218	1219	1221	1248	1306	1316	1326	1336	1348	1357
	1366	1375	1391	1402	1415	1433	1477	1540	1610	1623	1645	1646	1647	1651	1678
	1753	1876	1999	2000	2001	2002	2003	2007	2008	2009	2010	2011	2013		
.TITLE	13														
.WORD	23	27	29	30	791	2002	2003	2007	2008	2010	2011	2013			

ERRORS DETECTED: 0

*DSKZ:DZARCB.DSKZ:DZARCB/CRF=DSKZ:DZARCB

109

NAINDEC-11-DZACB-B
DZACB.P11

AA11-K DIAGNOSTIC

MACY11 27(663) 19-DEC-76 08:39 PAGE 48-41

SEQ 0112

RUN-TIME: 88 45 7 SECONDS
CORE USED: 25K

	FOR ANY	...B1	1171	011104	013705	...B5
	OVER THE PROGRAM	...C1	1213	011366	000137	...C5
	2. THE	...D1	1257			...D5
	ENAB	...E1	1332	011730	023737	...E5
	SWITCH	...F1	1354	012044	023737	...F5
	-----	...G1	1397	012320	023737	...G5
	2. SW10	...H1	1421	012436	001401	...H5
	DISPLAY SQUARE W	...I1	1438	012526	004537	...I5
	THIS TEST PROVID	...J1	1482	012700	004537	...J5
	WHICH TIME THE P	...K1	1527	013050	001454	...K5
	INITIALI	...L1	1549	013132	010001	...L5
	END OF P	...M1	1587	013300	000137	...M5
	TTY INPU	...N1	1618	013416	010077	...N5
136						
794						
2009						
(1)		...B2	1660	013566	001424	...B6
(1)	000020	...C2	1715			...C6
(1)	000060	...D2	1762	014150	005337	...D6
(1)		...E2	1804	014312	004537	...E6
(1)	000214	...F2	1832	014432	105777	...F6
(1)	001102 000	...G2		014660	041501	...G6
(2)	001220 000000	...H2	1887	015336	005015	...H6
(1)		...I2	1895	016001	104	...I6
80	001352 020320	...J2		016464	020113	...J6
134	001506 000005	...K2	1923	017102	042523	...K6
149		...L2	(1)	017544	051525	...L6
186	002132 005737	...M2	(1)	020206	000	...M6
236	002364 022626	...N2	1988	020460	010733	...N6
291	002600 104002	...B3	(1)	020626	005720	...B7
317	002744 104003	...C3	(1)	021004	032777	...C7
343	003106 001401	...D3	(1)	021250	004737	...D7
370	003244 023737	...E3	(1)	021422	000771	...E7
398	003424 012737	...F3	2007			...F7
427	003610 001401	...G3	(1)	022006	005704	...G7
465	004026 001401	...H3	(1)	022156	001003	...H7
495	004210 104001	...I3	(1)			...I7
528	004402 001401	...J3	(1)	022606	005766	...J7
549	004510 105737	...K3	(1)	023000	112613	...K7
598	004730 105737	...L3	(1)	023210	000764	...L7
644	005162 012777	...M3	(1)	023466	105037	...M7
680	005374 023737	...N3	(3)	023562	022762	...N7
714	005636 000005	...B4	ADDW1 =	000000		...B8
(1)	006110 000005	...C4	B	010531		...C8
764	006332 105277	...D4	CHO2	020462		...D8
(1)	006500	...E4				...E8
814	006654 104401	...F4	GEN1	010316		...F8
		...G4				...G8
877	007154 013701	...H4				...H8
931	007370 013702	...I4				...I8
983	007602 005077	...J4	TST17	003554		...J8
(1)	010056 013737	...K4	TYPON =	104404		...K8
1092	010376 105777	...L4	\$DEVN	001252		...L8
1129	010550 177	...M4	\$MAMS1	001224		...M8
1156	010757 047	...N4				...N8

ESCAPE	14#	1324	...B9
\$\$\$SKIP	14#	240	...C9
	351	359	...D9
INC	234	536	...E9
TST	146	157	...F9
	752	759	...G9
			...H9
			...I9
			...J9