

# AA11-K

DAIGNOSTIC  
MD-11-DZAAC-A

EP-DZAAC-A-DL-A

OCT 1976

COPYRIGHT ©1976

digital

FICHE 1 OF 1

Made In U.S.A.



IDENTIFICATION

PRODUCT CODE:           MAINDEC-11-DZAAC-A  
PRODUCT NAME:           AA11-K DIAGNOSTIC TEST  
DATE:                    OCTOBER 1976  
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 BY DIGITAL EQUIPMENT CORPORATION



1. ABSTRACT

THIS DIAGNOSTIC EXERCISES THE "A11-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 A11-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. A11-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 A11-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



A. THIS TEST IS DESIGNED TO AID IN THE ADJUSTING AND ALIGNMENT OF THE VR14 OR STORAGE SCOPE SCOPE ON THE RA11-K DISPLAY CONTROL.

B. STARTING SEQUENCE

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

C. CONTROL SWITCHES

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

<u>CONSOLE SWITCHES</u>	<u>FUNCTION</u>
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.

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



7.4 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 &lt;VR14&gt;

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 &amp; 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	INITIAL HEADER TYPEOUT AND WAIT FOR OPERATOR
172	
173	
177	INITIAL HEADER TYPEOUT AND WAIT FOR OPERATOR
227	DETERMINE THE NUMBER OF A11-K ON THIS SYSTEM
272	T1 TEST THAT THE A11-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)
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  
 813 DISPLAY HORIZONTAL LINE  
 825 DISPLAY A VERTICAL LINE  
 860 PINCUSHION TEST (DISPLAY SQUARE)  
 917 PLOT AN X  
 967 SCOPE SETTling TIME TEST  
 1032 PLOT CHARACTER SET  
 1158 CHANNEL 1 AND CHANNEL 2 TEST  
 1196 PHOSPHOR TEST  
 1208

### DAC ADJUSTMENT ROUTINES

-----

1210  
 1211  
 1213 MANUAL DISPLAY ROUTINE  
 1240 AUTO CALIBRATION  
 1297 T55 TEST THAT CH 3 IS AT +2.5 BIPOLAR  
 1307 T56 TEST THAT CH 4 IS AT -2.5 BIPOLAR  
 1320 T57 TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH  
 1330 T60 TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW  
 1342 T61 TEST THAT WRITE-THRU (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW  
 1351 T62 TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW  
 1360 T63 TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH  
 1369 T64 TEST THAT DELAY RETURN SETS READY  
 1385 T65 TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH  
 1396 T66 TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW  
 1409 T67 TEST THAT "ERASE" AND DONE CAN BE CLEARED AND SET  
 1436 T70 ADJUSTMENT ROUTINES FOR DAC 0 - 1  
 1480 T71 ADJUSTMENT ROUTINES FOR DAC 2 - 3

### MANUAL LOGIC TEST

MANUAL DAC CALIBRATION  
 DYNAMIC DAC CALIBRATION

### MISC. SUB-ROUTINES, ASCII MESSAGES AND SOFTWARE HANDLERS

1543  
 1613 SUBROUTINE TO ERASE STORAGE SCOPE SCREEN  
 1626 SUBROUTINE TO DRAW A HORIZONTAL LINE  
 1648 TIMER ROUTINE FOR VISUAL TEST PATTERNS  
 1649 ASCII MESSAGES  
 1650 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE  
 1654 SCOPE HANDLER ROUTINE  
 1681  
 1756  
 1875  
 1996  
 1997



1998	ERROR HANDLER ROUTINE
1999	ERROR MESSAGE TYPEOUT ROUTINE
2000	POWER DOWN AND UP ROUTINES
2004	BINARY TO OCTAL (ASCII) AND TYPE
2005	TYPE ROUTINE
2006	TTY INPUT ROUTINE
2007	READ AN OCTAL NUMBER FROM THE TTY
2008	APT COMMUNICATIONS ROUTINE
2010	TRAP DECODER
(3)	TRAP TABLE





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

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

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

(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

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) 000004  
 (1) 000010  
 (1) 000014  
 (1) 000014

ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS  
 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS  
 TBITVEC= 14 ;: "T" BIT  
 TRTVEC= 14 ;: TRACE TRAP

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

BPTVEC= 14  
 IOTVEC= 20  
 PWRVEC= 24  
 EMTVEC= 30  
 TRAPVEC=34  
 TKVEC= 60  
 TPVEC= 64  
 PIRQVEC=240

ABASE=170416  
 AVECT1=100360  
 APRIOR=200

:::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  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
22  
23  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
24  
25

```

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 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

```

000000

000174

000000

000176

000000

000200

00137

001462

001500

013520

27  
 (1)  
 (2)  
 (1)  
 (1) 000214  
 (1) 000046  
 (1) 000046 006566  
 (1) 000052  
 (1) 000052 000000  
 (1) 000214  
 28 001000  
 29  
 (1)  
 (2)  
 (1)  
 (2)  
 (1) 001000  
 (1) 000024  
 (1) 000024 000200  
 (1) 000044  
 (1) 000044 001000  
 (1) 001000  
 (1) 001000 000000  
 (1) 001002 001174  
 (1) 001004 000030  
 (1) 001006 000060  
 (1) 001010 000120  
 (1) 001012 000031

```

.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
$SVPC=      ;SAVE PC
.=46
SENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
.=52
.WORD 0     ;;2)SET LOC.52 TO ZERO
.= $SVPC    ;; RESTORE PC
.=1000

.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.SX=      ;;SAVE CURRENT LOCATION
.=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200      ;;FOR APT START UP
.=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR  ;;POINT TO APT HEADER BLOCK
.=.SX    ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MADR:  .WORD $MAIL  ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 30     ;; RUN TIM OF LONGEST TEST
$PASTM: .WORD 60     ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 120    ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

```



30  
 (1)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1) 001100  
 (1) 001100 000000  
 (1) 001102 000  
 (1) 001103 000  
 (1) 001104 000000  
 (1) 001106 000000  
 (1) 001110 000000  
 (1) 001112 000000  
 (1) 001114 000  
 (1) 001115 001  
 (1) 001116 000000  
 (1) 001120 000000  
 (1) 001122 000000  
 (1) 001124 000000  
 (1) 001126 000000  
 (1) 001130 000000  
 (1) 001132 000000  
 (1) 001134 000  
 (1) 001135 000  
 (1) 001136 000000  
 (1) 001140 177570  
 (1) 001142 177570  
 (1) 001144 177560  
 (1) 001146 177562  
 (1) 001150 177564  
 (1) 001152 177566  
 (1) 001154 000  
 (1) 001155 002  
 (1) 001156 012  
 (1) 001157 000  
 (1) 001160 000000  
 (1)  
 (3) 001162 000000  
 (3) 001164 000000  
 (1) 001166 000000  
 (1) 001170 077  
 (1) 001171 015  
 (1) 001172 000012  
 (2)  
 (2)  
 (2)  
 (3)  
 (2)  
 (2) 001174  
 (2) 001174 000000  
 (2) 001176 000000  
 (2) 001200 000000  
 (2) 001202 000000  
 (2) 001204 000000

.SBTTL COMMON TAGS

```

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

```

SCMTAG: . =1100

```

; START OF COMMON TAGS
STSTNM: .WORD 0
SERFLG: .BYTE 00
SICNT: .WORD 00
SLPADP: .WORD 00
SLPERR: .WORD 00
SERTTL: .WORD 00
SITEMB: .BYTE 0
SERMAX: .BYTE 1
SERRPC: .WORD 0
SGDADR: .WORD 00
SBDADR: .WORD 00
SGDDAT: .WORD 00
SBDAT: .WORD 00
SAUTOB: .BYTE 0
SINTAG: .BYTE 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
STKS: 177560
STKB: 177562
STPS: 177564
STPB: 177566
SNUL: .BYTE 0
SFILLS: .BYTE 2
SFILLC: .BYTE 12
STPFLG: .BYTE 0
SREGAD: .WORD 0

```

```

; CONTAINS THE TEST NUMBER
; CONTAINS ERROR FLAG
; CONTAINS SUBTEST ITERATION COUNT
; CONTAINS SCOPE LOOP ADDRESS
; CONTAINS SCOPE RETURN FOR ERRORS
; CONTAINS TOTAL ERRORS DETECTED
; CONTAINS ITEM CONTROL BYTE
; CONTAINS MAX. ERRORS PER TEST
; CONTAINS PC OF LAST ERROR INSTRUCTION
; CONTAINS ADDRESS OF 'GOOD' DATA
; CONTAINS ADDRESS OF 'BAD' DATA
; CONTAINS 'GOOD' DATA
; CONTAINS 'BAD' DATA
; RESERVED--NOT TO BE USED
; AUTOMATIC MODE INDICATOR
; INTERRUPT MODE INDICATOR
; ADDRESS OF SWITCH REGISTER
; ADDRESS OF DISPLAY REGISTER
; TTY KBD STATUS
; TTY KBD BUFFER
; TTY PRINTER STATUS REG. ADDRESS
; TTY PRINTER BUFFER REG. ADDRESS
; CONTAINS NULL CHARACTER FOR FILLS
; CONTAINS # OF FILLER CHARACTERS REQUIRED
; INSERT FILL CHARS. AFTER A "LINE FEED"
; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
; CONTAINS THE ADDRESS FROM WHICH ($REG0) WAS OBTAINED
; CONTAINS (($REGAD)+0)
; CONTAINS (($REGAD)+2)
; MAX. NUMBER OF ITERATIONS
; QUESTION MARK
; CARRIAGE RETURN
; LINE FEED

```

```

*****
.SBTTL APT MAILBOX-ETABLE

```

```

*****
.EVEN
SMAIL:
MSGTY: .WORD AMSGTY ; APT MAILBOX
SFATAL: .WORD AFATAL ; MESSAGE TYPE CODE
STESTN: .WORD ATESTN ; FATAL ERROR NUMBER
SPASS: .WORD APASS ; TEST NUMBER
SDEVCT: .WORD ADEVCT ; PASS COUNT
; DEVICE COUNT

```

# H02

MAINDEC-11-DZAAC-A  
DZAACA.P11

AA11-K DIAGNOSTIC  
APT MAILBOX-ETABLE

MACY11 27(732)

20-AUG-76 09:27 PAGE 3-2

SEQ 0020

(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			



```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
(1)
(1) 001256 $ERRTB:
(1)
(1) 31 ;ITEM 1
(1) 33 ;EM1 ;AA11-K STATUS REGISTER IN ERROR
(1) 34 ;DH1 ;ERRPC IBASE GOOD BAD
(1) 35 001256 016024 ;DT1 ;$ERRPC STAT $GDDAT $BDDAT
(1) 36 001260 016645 ;DF0
(1) 37 001262 020264
(1) 38 001264 020400
(1) 39
(1) 40 ;ITEM 2
(1) 41 001266 016055 ;EM2 ;DAC0 REGISTER IN ERROR
(1) 42 001270 016645 ;DH1 ;ERRPC IBASE GOOD BAD
(1) 43 001272 020276 ;DT2 ;$ERRPC DAC0 $GDDAT $BDDAT
(1) 44 001274 020400 ;DF0
(1) 45
(1) 46 ;ITEM 3
(1) 47 001276 016104 ;EM3 ;DAC1 REGISTER IN ERROR
(1) 48 001300 016645 ;DH1 ;ERRPC IBASE GOOD BAD
(1) 49 001302 020310 ;DT3 ;$ERRPC DAC1 $GDDAT $BDDAT
(1) 50 001304 020400 ;DF0
(1) 51
(1) 52 ;ITEM 4
(1) 53 001306 016133 ;EM4 ;DAC #2 REGISTER IN ERROR
(1) 54 001310 016645 ;DH1 ;ERRPC IBASE GOOD BAD
(1) 55 001312 020322 ;DT4 ;$ERRPC DAC2 $GDDAT $BDDAT
(1) 56 001314 020400 ;DF0
(1) 57
(1) 58 ;ITEM 5
(1) 59 001316 016162 ;EM5 ;DAC #3 REGISTER IN ERROR
(1) 60 001320 016645 ;DH1 ;ERRPC IBASE GOOD BAD
(1) 61 001322 020334 ;DT5 ;$ERRPC DAC3 $GDDAT $BDDAT
(1) 62 001324 020400 ;DF0
(1) 63
(1) 64 ;ITEM 6
(1) 65 001326 016211 ;EM6 ;AA11-K FAILED TO INTERRUPT
(1) 66 001330 016701 ;DH6 ;ERRPC IBASE
(1) 67 001332 020346 ;DT6 ;$ERRPC STAT
(1) 68 001334 020400 ;DF0
(1) 69

```

71			; ITEM	7					
72	001336	016244		EM7					;AA11-K INTERRUPTED IN ERROR
73	001340	016701		DH6					;ERRPC IBASE
74	001342	020346		DT6					;SERRPC STAT
75	001344	020400		DFO					
76									
77			; ITEM	10					
78	001346	016300		EM10					;BUS TIME-OUT ON AA11-K ADDRESSES
79	001350	016701		DH6					;ERRPC IBASE
80	001352	020346		DT6					;SERRPC STAT
81	001354	020400		DFO					
82									
83			; ITEM	11					
84	001356	016351		EM11					;AA11-K READY BIT ERROR
85	001360	016645		DH1					;ERRPC IBASE GOOD BAD
86	001362	020264		DT1					;SERRPC STAT SGDDAT SBDDAT
87	001364	020400		DFO					
88									
89			; ITEM	12					
90	001366	016400		EM12					;POWER SUPPLY VOLTAGE WAS INCORRECT
91	001370	016747		DH14					;ERRPC BASE GOOD BAD
92	001372	020366		DT14					;SERRPC SBASE SGDDAT SBDDAT
93	001374	020400		DFO					
94									
95			; ITEM	13					
96	001376	016443		EM13					;A PREVIOUSLY EXISTING AA11-K DOES NOT RESPOND NOW
97	001400	016715		DH13					;ERRPC BASE FIRST# #NOW
98	001402	020354		DT13					;SERRPC SBASE EVER SUNIT
99	001404	020400		DFO					
100									
101			; ITEM	14					
102	001406	016464		EM14					;AA11-K LOGIC SIGNAL HIGH OUTPUT TO LOW
103	001410	016747		DH14					;ERRPC BASE GOOD BAD
104	001412	020366		DT14					;SERRPC SBASE SGDDAT SBDDAT SPREAD
105	001414	020400		DFO					
106									
107			; ITEM	15					
108	001416	016533		EM15					;AA11-K LOGIC LOW OUTPUT TO HIGH
109	001420	016747		DH14					;ERRPC BASE GOOD BAD
110	001422	020366		DT14					;SERRPC SBASE SGDDAT SBDDAT SPREAD
111	001424	020400		DFO					
112									
113			; ITEM	16					
114	001426	016602		EM16					;SELECTED DAC HAS A LINEARITY ERROR
115	001430	016747		DH14					;ERRPC BASE GOOD BAD
116	001432	020366		DT14					;SERRPC SBASE SGDDAT SBDDAT
117	001434	020400		DFO					
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 020432  BEGIN: CLR    TEMP
130 001466 005037 020442          CLR    EVER          ;RESET POINTER
131 001472 005037 020444          CLR    EVER1
132 001476 000403          BR     BEG
133 001500 012737 000001 020432 BEGIN1: MOV   #1,TEMP
134 001506 000005          BEG:  RESET
136          .SBTTL INITIALIZE THE COMMON TAGS
(1)          ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
(1) 001510 012706 001100          MOV   #SCMTAG,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 020744 000020  MOV   #SCOPE,#IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
(1) 001536 012737 000340 000022  MOV   #340,#IOTVEC+2 ;; LEVEL 7
(1) 001544 012737 021160 000030  MOV   #ERROR,#EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
(1) 001552 012737 000340 000032  MOV   #340,#EMTVEC+2 ;; LEVEL 7
(1) 001560 012737 023520 000034  MOV   #TRAP,#TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
(1) 001566 012737 000340 000036  MOV   #340,#TRAPVEC+2; LEVEL 7
(1) 001574 012737 021444 000024  MOV   #SPWRDH,#PWRVEC ;; POWER FAILURE VECTOR
(1) 001602 012737 000340 000026  MOV   #340,#PWRVEC+2 ;; LEVEL 7
(1) 001610 005037 001166          CLR   $TIMES        ;; INITIALIZE NUMBER OF ITERATIONS
(1) 001614 012737 001614 0C1106  MOV   #.SLPADR      ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
(2)          ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)          ;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001622 013746 000004          MOV   #ERRVEC,-(SP) ;; SAVE ERROR VECTOR
(2) 001626 012737 001662 000004  MOV   #64$,#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  55$: MOV   #SWREG,SWR  ;; POINT TO SOFTWARE SWR
(2) 001676 012737 000174 001142  MOV   #DISPREG,DISPLAY
(2) 001704 012637 000004          66$: 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          67$:
137 001732 005037 177776          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,R0	;LOAD POINTER
151	001774	013720	001250	10\$:	MOV \$BASE,(R0)+	;LOAD BASE ADDRESS
152	002000	022700	001456		CMP #IV,R0	;TEST FOR DONE
153	002004	001373			BNE 10\$	;BR
154	002006	012700	001446		MOV #DAC0,R0	;LOAD 2ND ADDRESS
155	002012	012701	000002		MOV #2,R1	;LOAD R1
156	002016	060120		12\$:	ADD R1,(R0)+	;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	020436	MOV \$VECT1+1,BRLEV1	;LOAD BR LEVEL
165	002066	042737	177437	020436	BIC #177437,BRLEV1	
166	002074	013737	020436	020440	MOV BRLEV1,BRLEV2	
167	002102	162737	000040	020440	SUB #40,BRLEV2	
168	002110	000207			RTS PC	;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 020432 TST TEMP ;TEST IF START OR RESTART
187 002136 001065 BNE CTRLC ;RESTART
188 002140 005737 000042 TST @#42 ;TEST IF MONITOR
189 002144 001067 BNE MTEST ;BR IF NOT
190 002146 104400 TYPE ;CALL MESSAGE PRINTER VIA 'EMT'
191 002150 014662 TITLE ;TYPE PROGRAM HEADER.
192 002152 012706 001100 WAITIN: MOV #STACK,SP ;RESET STACK
193 002156 104406 CKSWR ;TEST FOR CTRL G
194 002160 104410 RDLIN ;READ TWO CHARACTERS
195 002162 013637 020446 MOV @ (SP)+,OPRIN ;READ THE CHARACTER
196 002166 042737 000000 020446 BIC #0,OPRIN ;MASK OTHER CHARACTERS
197
198 002174 023737 002112 020446 CMP AL,OPRIN ;TEST FOR "AL"
199 002202 001427 BEQ 10$ ;BR IF YES
200 002204 023737 002114 020446 CMP AD,OPRIN ;TEST FOR "AD"
201 002212 001425 BEQ 11$ ;BR IF YES
202 002214 023737 002116 020446 CMP AC,OPRIN ;TEST FOR "AC"
203 002222 001423 BEQ 12$ ;BR IF YES
204 002224 023737 002120 020446 CMP ML,OPRIN ;TEST FOR "ML"
205 002232 001421 BEQ 13$ ;BR FI YES
206 002234 023737 002122 020446 CMP MD,OPRIN ;TEST FOR "MD"
207 002242 001417 BEQ 14$ ;BR IF YES
208 002244 023737 002124 020446 CMP MC,OPRIN ;TEST FOR "MC"
209 002252 001415 BEQ 15$ ;BR IF YES
210 002254 104400 TYPE
211 002256 016774 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 011500 12$: JMP AUTCAL ;RUN THE AUTO CALIBRATION (FACTORY ONLY)
217 002276 000137 013170 13$: JMP MANUL ;RUN THE MANUAL LOGIC TEST
218 002302 000137 011364 14$: JMP FULRMP ;RUN THE RAMP PATTERN ON FOUR DAC'S
219 002306 000137 013454 15$: JMP CALDAC ;RUN THE MANUAL CAL OF THE DAC
220
221 002312 104400 CTRLC: TYPE
222 002314 017002 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 1$: TST @BDDAT ;TEST IF ADDR EXISTS
233 002350 063737 001436 001126 ADD VADDR,$BDDAT ;UPDATE THE BUS ADDRESS
234 002356 005237 001206 INC $UNIT ;UPDATE UNIT COUNT
235 002362 000406 BR 3$ ;
236 002364 022626 2$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
237 002366 005737 001206 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 020442 3$: TST EVER ;TEST IF # HAS BEEN REPORTED
242 002404 100417 BMI 4$ ;BR IF IT HAS
243 002406 104400 TYPE
244 002410 017010 FOUND1 ;TELL OPERATOR THE # OF AA11-K'S
245 002412 013746 001206 MOV $UNIT,-(SP)
246 002416 104402 TYPOS
247 002420 002 .BYTE 2
248 002421 000 .BYTE 0
249 002422 104400 TYPE
250 002424 017034 FOUND2
251 002426 013737 001206 020442 MOV $UNIT,EVER ;SAVE THE # OF AA11-K'S FOR LATER
252 002434 052737 100000 020442 BIS #BIT1$,EVER ;SET "REPORTED # FLAG"
253 002442 000405 BR 5$ ;
254 002444 123737 020442 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 $UNIT ;RESET UNIT POINTER
258 002462 004737 001742 JSR PC,LDTRAP ;LOAD TRAP CATCHER AND BUS ADDRESSES
259
260 002466 000240 LTEST: NOP
272 ;*****
(3) ;*TEST 1 TEST THAT THE AA11-K RESPONDS TO THE CPU
(3) ;*****
(2) †ST1: SCOPE
273 002472 012737 002506 001106 MOV #3$,SLPADR
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 1$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
282 002536 104010 ERROR 10 ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE AA11-K
283 002540 012737 000006 000004 2$: MOV #6$,ERRVEC ;LOAD RETURN

```



```

285      ;:*****
(3)      ;:TEST 2      TEST THAT THE DACO REGISTER CAN BE CLEARED
(3)      ;:*****
(2)      002546 000004
286      002550 005037 001124
287      002554 013777 001124 176664
288      002562 017737 176660 001126
289      002570 023737 001124 001126
290      002576 001401
291      002600 104002
          SCOPE
          CLR      $GDDAT      ;LOAD EXPECTED
          MOV      $GDDAT,2DACO ;LOAD REG
          MOV      2DACO,$BDDAT ;READ REG
          CMP      $GDDAT,$BDDAT ;COMPARE
          BEQ      TST3      ;;BR IF EQUAL
          ERROR   2          ;ERROR, DACO REGISTER NOT = 0

(3)      ;:*****
(3)      ;:TEST 3      TEST THAT THE DACO REGISTER CAN BE LOADED WITH #7777
(3)      ;:*****
(2)      002602 000004
294      002604 012737 007777 001124
295      002612 013777 001124 176626
296      002620 017737 176622 001126
297      002626 023737 001124 001126
298      002634 001401
299      002636 104002
          SCOPE
          MOV      #7777,$GDDAT ;LOAD EXPECTED
          MOV      $GDDAT,2DACO ;LOAD REG
          MOV      2DACO,$BDDAT ;READ REG
          CMP      $GDDAT,$BDDAT ;COMPARE
          BEQ      TST4      ;;BR IF EQUAL
          ERROR   2          ;ERROR, DACO REGISTER NOT = 7777

(3)      ;:*****
(3)      ;:TEST 4      TEST THAT THE DACO REGISTER CAN HOLD A FLOATING 1 PATTERN
(3)      ;:*****
(2)      002640 000004
(1)      002642 012737 700100 001166
302      002650 012737 004000 001124
303      002656 013777 001124 176562
304      002664 017737 176556 001126
305      002672 023737 001124 001126
306      002700 001401
307      002702 104002
308      002704 006237 001124
309      002710 001362
          SCOPE
          MOV      #100,$TIMES ;;DO 100 ITERATIONS
          MOV      #BIT11,$GDDAT ;LOAD EXPECTED
          MOV      $GDDAT,2DACO ;LOAD DACO REGISTER
          MOV      2DACO,$BDDAT ;READ THE REGISTER
          CMP      $GDDAT,$BDDAT ;COMPARE THE DATA
          BEQ      2$          ;;BR IF SAME
          ERROR   2          ;ERROR, DACO REGISTER FAILED TO HOLD A FLOATING
          ASR      $GDDAT      ;CHANGE THE DATA
          BNE     1$          ;BR AND TEST MORE DATA
          1$:
          MOV      $GDDAT,2DACO
          MOV      2DACO,$BDDAT
          CMP      $GDDAT,$BDDAT
          BEQ      2$
          ERROR   2
          2$:
          ASR      $GDDAT
          BNE     1$

```



```

311 (3) *****
(3) *TEST 5 TEST THAT THE DAC #1 REGISTER CAN BE CLEARED
(2) 002712 000004 *****
312 002714 005037 001124
313 002720 013777 001124 176522
314 002726 017737 176516 001126
315 002734 023737 001124 001126
316 002742 001401
317 002744 104003
318
319 *****
(3) *TEST 6 TEST THAT THE Y REGISTER CAN BE LOADED WITH #7777
(3) *****
(2) 002746 000004
320 002750 012737 007777 001124
321 002756 013777 001124 176464
322 002764 017737 176460 001126
323 002772 023737 001124 001126
324 003000 001401
325 003002 104003
326
327 *****
(3) *TEST 7 TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN
(3) *****
(2) 003004 000004
(1) 003006 012737 000100 001166
328 003014 012737 004000 001124
329 003022 013777 001124 176420
330 003030 017737 176414 001126
331 003036 023737 001124 001126
332 003044 001401
333 003046 104003
334 003050 006237 001124
335 003054 001362

```

```

*****
*TEST 5 TEST THAT THE DAC #1 REGISTER CAN BE CLEARED
*****
TST5: SCOPE
CLR $GDDAT ;LOAD EXPECTED
MOV $GDDAT, @DAC1 ;LOAD Y REG
MOV @DAC1, $BDDAT ;READ REG
CMP $GDDAT, $BDDAT ;COMPARE
BEQ TST6 ;;BR IF EQUAL
ERROR 3 ;ERROR, DAC1 REGISTER NOT = 0

*****
*TEST 6 TEST THAT THE Y REGISTER CAN BE LOADED WITH #7777
*****
TST6: SCOPE
MOV #7777, $GDDAT ;LOAD EXPECTED
MOV $GDDAT, @DAC1 ;LOAD Y REG
MOV @DAC1, $BDDAT ;READ REG
CMP $GDDAT, $BDDAT ;COMPARE
BEQ TST7 ;;BR IF EQUAL
ERROR 3 ;ERROR, DAC1 REGISTER NOT = 7777

*****
*TEST 7 TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN
*****
TST7: SCOPE
MOV #100, $TIMES ;;DO 100 ITERATIONS
MOV #BIT11, $GDDAT ;LOAD EXPECTED
1S: MOV $GDDAT, @DAC1 ;LOAD THE REGISTER
MOV @DAC1, $BDDAT ;READ THE REGISTER
CMP $GDDAT, $BDDAT ;COMPARE THE DATA
BEQ 2S ;;BR IF DATA IS SAME
ERROR 3 ;ERROR, DAC1 REGISTER FAILED TO HOLD A FLOATING
2S: ASR $GDDAT ;CHANGE THE DATA
BNE 1S ;BR AND TEST MORE DATA

```

D03

MAINDEC-11-DZAAC-A  
DZARCA.P11 T7

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 10  
TEST THAT THE Y REGISTER CAN HOLD A FLOATING 1 PATTERN

SEQ 0029

```

337
338
(3)
(3)
(2) 003056 000004
339 003060 005037 001124
340 003064 013777 001124 176360
341 003072 017737 176354 001126
342 003100 023737 001124 001126
343 003106 001401
344 003110 104004
345
346
(3)
(3)
(2) 003112 000004
347 003114 012737 007777 001124
348 003122 013777 001124 176322
349 003130 017737 176316 001126
350 003136 023737 001124 001126
351 003144 001401
352 003146 104004
353
354
(3)
(3)
(2) 003150 000004
(1) 003152 012737 000100 001166
355 003160 012737 004000 001124
356 003166 013777 001124 176256
357 003174 017737 176252 001126
358 003202 023737 001124 001126
359 003210 001401
360 003212 104004
361 003214 006237 001124
362 003220 001362

*****
*TEST 10 TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
*****
TST10: 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
*****
TST11: 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
*****
TST12: 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

```

# E03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T12

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 11  
TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN

SEQ 0030

```
364
365
366      ;*****
(3)      ;*TEST 13      TEST THAT THE DAC #3 REGISTER CAN BE CLEARED
(3)      ;*****
(2) 003222 000004
367 003224 005037 001124      †TST13: SCOPE
368 003230 013777 001124 176216      CLR      $GDDAT      ;LOAD EXPECTED
369 003236 017737 176212 001126      MOV      $GDDAT,2DAC3 ;LOAD REG
370 003244 023737 001124 001126      MOV      2DAC3,$BDDAT ;READ REG
371 003252 001401      CMP      $GDDAT,$BDDAT ;COMPARE
372 003254 104005      BEQ      TST14      ;;BR IF EQUAL
373      ERROR      5      ;ERROR, DAC #3 REGISTER NOT = 0
374
(3)      ;*****
(3)      ;*TEST 14      TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777
(3)      ;*****
(2) 003256 000004
375 003260 012737 007777 001124      †TST14: SCOPE
376 003266 013777 001124 176160      MOV      #7777,$GDDAT ;LOAD EXPECTED
377 003274 017737 176154 001126      MOV      $GDDAT,2DAC3 ;LOAD REG
378 003302 023737 001124 001126      MOV      2DAC3,$BDDAT ;READ REG
379 003310 001401      CMP      $GDDAT,$BDDAT ;COMPARE
380 003312 104005      BEQ      TST15      ;;BR IF EQUAL
381      ERROR      5      ;ERROR, DAC #3 REGISTER NOT = 7777
382
(3)      ;*****
(3)      ;*TEST 15      TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN
(3)      ;*****
(2) 003314 000004
(1) 003316 012737 000100 001166      †TST15: SCOPE
383 003324 012737 004000 001124      MOV      #100,$TIMES ;;DO 100 ITERATIONS
384 003332 013777 001124 176114      1$: MOV      #BIT11,$GDDAT ;LOAD EXPECTED
385 003340 017737 176110 001126      MOV      $GDDAT,2DAC3 ;LOAD DAC #3 REGISTER
386 003346 023737 001124 001126      MOV      2DAC3,$BDDAT ;READ THE REGISTER
387 003354 001401      CMP      $GDDAT,$BDDAT ;COMPARE THE DATA
388 003356 104005      BEQ      2$      ;;BR IF SAME
389 003360 006237 001124      2$: ERROR      5      ;ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN
390 003364 001362      ASR      $GDDAT ;CHANGE THE DATA
      BNE      1$ ;BR AND TEST MORE DATA
```



# F03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T16

RA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 12  
TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA

SEQ 0031

```
392 (3) ;*****  
(3) ;*TEST 16 TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA  
(2) ;*****  
393 003366 000004 TST16: SCOPE  
394 003370 012777 001111 176050 MOV #1111, @DAC0 ;LOAD DAC #0  
395 003376 012777 002222 176044 MOV #2222, @DAC1 ;LOAD DAC #1  
396 003404 012777 004444 176040 MOV #4444, @DAC2 ;LOAD DAC #2  
397 003412 012777 007777 176034 MOV #7777, @DAC3 ;LOAD DAC #3  
398 003420 005077 176020 CLR @STAT ;CLEAR STATUS  
399 003424 012737 001111 001124 MOV #1111, $GDDAT ;LOAD EXPECTED  
400 003432 017737 176010 001126 MOV @DAC0, $BDDAT ;READ REG  
401 003440 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE  
402 003446 001401 BEQ 1$ ;;BR IF EQUAL  
403 003450 104002 ERROR 2 ;ERROR, SELECTED DAC0 IN ERROR  
404 003452 012737 002222 001124 1$: MOV #2222, $GDDAT ;LOAD EXPECTED  
405 003460 017737 175764 001126 MOV @DAC1, $BDDAT ;READ REG  
406 003466 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE  
407 003474 001401 BEQ 2$ ;;BR IF EQUAL  
408 003476 104003 ERROR 3 ;ERROR, SELECTED DAC1 IN ERROR  
409 003500 012737 004444 001124 2$: MOV #4444, $GDDAT ;LOAD EXPECTED  
410 003506 017737 175740 001126 MOV @DAC2, $BDDAT ;READ REG  
411 003514 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE  
412 003522 001401 BEQ 3$ ;;BR IF SAME  
413 003524 104004 ERROR 4 ;ERROR, SELECTED DAC #2 IN ERROR  
414 003526 012737 007777 001124 3$: MOV #7777, $GDDAT ;LOAD EXPECTED  
415 003534 017737 175714 001126 MOV @DAC3, $BDDAT ;READ REG  
416 003542 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE  
417 003550 001401 BEQ TST17 ;;BR IF SAME  
418 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      ;COMPARE
435 003646 001401      BEQ      IS      ;;BR IF SAME
436 003650 104001      ERROR      1      ;ERROR, STATUS NOT = 202
437 003652 005077 175566      IS:      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      IS      ;;BR IF EQUAL
450 003740 104001      ERROR      1      ;ERROR, STATUS NOT = 204
451 003742 005077 175476      IS:      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) 003774 000004
461 003776 012777 000010 175440
462 004004 012737 000210 001124
463 004012 017737 175426 001126
464 004020 023737 001124 001126
465 004026 001401
466 004030 104001
467
468 004032 005077 175406
469 004036 012737 000200 001124
470 004044 017737 175374 001126
471 004052 023737 001124 001126
472 004060 001401
473 004062 104001
474
475
(3)
(3)
(2) 004064 000004
476 004066 012777 000020 175350
477 004074 012737 000220 001124
478 004102 017737 175336 001126
479 004110 023737 001124 001126
480 004116 001401
481 004120 104001
482 004122 005077 175316
483 004126 012737 000200 001124
484 004134 017737 175304 001126
485 004142 023737 001124 001126
486 004150 001401
487 004152 104001

```

```

*****
*TEST 22 TEST THAT MODE BIT 3 CAN BE SET
*****

```

```

†ST22: SCOPE
MOV #BIT3,STAT ;LOAD
MOV #BIT7,STAT ;LOAD EXPECTED
MOV STAT,SDDAT ;READ REG.
CMP SGDDAT,SDDAT ;COMPARE
BEQ IS ;;BR IF EQUAL
ERROR 1 ;ERROR, STATUS NOT = 210

IS: CLR STAT ;CLEAR REG.
MOV #BIT7,SGDDAT ;LOAD EXPECTED
MOV STAT,SDDAT ;READ REG.
CMP SGDDAT,SDDAT ;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
*****

```

```

†ST23: SCOPE
MOV #BIT4,STAT ;LOAD STATUS
MOV #BIT7,STAT ;LOAD EXPECTED
MOV STAT,SDDAT ;READ REGISTER
CMP SGDDAT,SDDAT ;COMPARE
BEQ IS ;;BR IF SAME
ERROR 1 ;ERROR, EXT. DELAY BIT FAILED TO SET

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

```



```

489 (3) *****
(3) *TEST 24 TEST THAT INTERRUPT ENABLE (BIT 6) CAN BE SET
(2) *****
(2) 004154 000004 TST24: SCOPE
490 004156 012777 000100 175260 MOV #BIT6,STAT ;LOAD DISPLAY STATUS
491 004164 012737 000300 001124 MOV #BIT7!BIT6,$GDDAT ;LOAD EXPECTED
492 004172 017737 175246 001126 MOV STAT,$BDDAT ;READ REG
493 004200 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
494 004206 001401 BEQ TST25 ;;BR IF EQUAL
495 004210 104001 ERROR 1 ;ERROR, STATUS NOT = 300

```

```

496 (3) *****
497 (3) *TEST 25 TEST THAT CHANNEL (BIT 9) CAN BE SET
(2) *****
(2) 004212 000004 TST25: SCOPE
498 004214 012777 001000 175222 MOV #BIT9,STAT ;LOAD DISPLAY STATUS
499 004222 012737 001200 001124 MOV #BIT9!BIT7,$GDDAT ;LOAD EXPECTED
500 004230 017737 175210 001126 MOV STAT,$BDDAT ;READ REG
501 004236 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
502 004244 001401 BEQ TST26 ;;BR IF EQUAL
503 004246 104001 ERROR 1 ;ERROR, STATUS NOT = 1200

```

```

504 (3) *****
505 (3) *TEST 26 TEST THAT STORE (BIT 10) CAN BE SET
(2) *****
(2) 004250 000004 TST26: SCOPE
506 004252 012777 002000 175164 MOV #BIT10,STAT ;LOAD DISPLAY STATUS
507 004260 012737 002200 001124 MOV #BIT10!BIT7,$GDDAT ;LOAD EXPECTED
508 004266 017737 175152 001126 MOV STAT,$BDDAT ;READ REG
509 004274 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
510 004302 001401 BEQ TST27 ;;BR IF EQUAL
511 004304 104001 ERROR 1 ;ERROR, STATUS NOT = 2200

```

```

512 (3) *****
513 (3) *TEST 27 TEST THAT WRITE THRU (BIT 11) CAN BE SET
(2) *****
(2) 004306 000004 TST27: SCOPE
514 004310 012777 004000 175126 MOV #BIT11,STAT ;LOAD DISPLAY STATUS
515 004316 012737 004200 001124 MOV #BIT11!BIT7,$GDDAT ;LOAD EXPECTED
516 004324 017737 175114 001126 MOV STAT,$BDDAT ;READ REG
517 004332 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
518 004340 001401 BEQ TST30 ;;BR IF EQUAL
519 004342 104001 ERROR 1 ;ERROR, STATUS NOT = 4200
520

```

# J03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T30

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 16  
TEST THAT THE LOW BYTE OF THE STATUS REGISTER CAN BE CLEARED

SEQ 0035

```
522      ;:*****  
(3)      ;*TEST 30      TEST THAT THE LOW BYTE OF THE STATUS REGISTER CAN BE CLEARED  
(3)      ;:*****  
(2) 004344 000004  
523 004346 012777 005014 175070      †TST30: SCOPE  
524 004354 012737 005200 001124      MOV      #BIT11!BIT9!BIT3!BIT2,@STAT      ;LOAD THE STATUS REGISTER  
525 004362 105077 175056      MOV      #BIT11!BIT9!BIT7,$GDDAT      ;LOAD THE EXPECTED RESULT  
526 004366 017737 175052 001126      CLRB     @STAT      ;CLEAR LOW STATUS BYTE  
527 004374 023737 001124 001126      MOV      @STAT,$BDDAT      ;READ THE REGISTER  
528 004402 001401      CMP      $GDDAT,$BDDAT      ;COMPARE  
529 004404 104001      BEQ     TST31      ;;BR IF EXPECTED  
530      ERROR      1      ;FAILED TO ADDRESS ONLY THE LOW BYTE OF STATUS R  
531  
532      ;:*****  
(3)      ;*TEST 31      TEST THAT THE HIGH BYTE OF THE STATUS REGISTER CAN BE CLEARED  
(3)      ;:*****  
(2) 004406 000004  
533 004410 012777 005014 175026      †TST31: SCOPE  
534 004416 012737 000214 001124      MOV      #BIT11!BIT9!BIT3!BIT2,@STAT      ;LOAD THE STATUS REGISTER  
535 004424 013737 001444 010512      MOV      #BIT7!BIT3!BIT2,$GDDAT      ;LOAD EXPECTED RESULTS  
536 004432 005237 010512      MOV      STAT,CHRCOL      ;MAKE THE HIGH BYTE ADDRESS  
537 004436 105077 004050      INC     CHRCOL      ;MAKE IT ODD  
538 004442 017737 174776 001126      CLRB     @CHRCOL      ;CLEAR THE HIGH BYTE  
539 004450 023737 001124 001126      MOV      @STAT,$BDDAT      ;RREAD THE REGISTER  
540 004456 001401      CMP      $GDDAT,$BDDAT      ;COMPARE  
541 004460 104001      BEQ     TST32      ;;BR IF EXPECTED  
      ERROR      1      ;FAILED TO CLEAR ONLY THE HIGH BYTE OF STATUS RE
```

# K03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T32

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 17  
TEST THAT WHEN INTENSIFY BIT IS SET THAT THE READY BIT CLEARS

SEQ 0036

```
543 ;*****  
(3) ;*TEST 32 TEST THAT WHEN INTENSIFY BIT IS SET THAT THE READY BIT CLEARS  
(3) ;*****  
(2) 004462 000004 †TST32: SCOPE  
544 ; 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 $BDDAT ;BR IF NOT SET  
550 004514 100002 BPL 1$ ;READY FAILED TO CLEAR  
551 004516 104011 ERROR 11 ;BR TO SCOPE  
552 004520 000414 BR TST33 ;TEST READY  
553 004522 105777 174716 1$: TSTB ‡STAT ;NEXT TEST  
554 004526 100411 BMI TST33 ;DELAY  
555 004530 005300 DEC RO ;LOAD EXPECTED  
556 004532 001373 BNE 1$ ;READ REG  
557 004534 012737 000200 001124 MOV #BIT7,$GDDAT ;READY FAILED TO SET AFTER A DELAY  
558 004542 017737 174676 001126 MOV ‡STAT,$BDDAT ;IS STORAGE SCOPE CONNECTED BUT NOT TURNED ON ??  
559 004550 104011 ERROR 11 ;*****  
560 ;*****  
561 ;*TEST 33 TEST THAT MODE 1 (INTENSIFY ON DAC0) CLEARS THE READY FLAG  
(3) ;*****  
(3) ;*****  
(2) 004552 000004 †TST33: SCOPE  
562 ; 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 $BDDAT ;TEST READY  
577 004640 000240 NOP ;BR IF CLEAR  
578 004642 100002 BPL 1$ ;ERROR, MODE 1 LOAD DAC0 FAILED TO CLEAR READY F  
579 004644 104011 ERROR 11 ;BR TO SCOPE  
580 004646 000414 BR †TST34 ;TEST READY  
581 582 004650 105777 174570 1$: TSTB ‡STAT ;NEXT TEST  
583 004654 100411 BMI TST34 ;DELAY  
584 004656 005300 DEC RO ;TEST READY AGAIN  
585 004660 001373 BNE 1$ ;LOAD EXPECTED  
586 004662 012737 000204 001124 MOV #BIT7:BIT2,$GDDAT ;READ REG  
587 004670 017737 174550 001126 MOV ‡STAT,$BDDAT ;ERROR, READY FAILED TO SET  
588 004676 104011 ERROR 11 ; AFTER MODE 1 OPERATION  
589  
590
```



L03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T34

RA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 18  
TEST THAT MODE 2 (INTENSIFY ON DAC1) CLEARS THE READY FLAG

SEQ 0037

```

592      ;*****
(3)      ;*TEST 34      TEST THAT MODE 2 (INTENSIFY ON DAC1) CLEARS THE READY FLAG
(3)      ;*****
(2) 004700 000004
593      TST34: SCOPE
594      ; AND THEN SETS IT
594 004702 012700 001000      MOV      #1000,RO      ;SET UP DELAY
595 004706 012737 000210 001124      MOV      #BIT7!BIT3,$GDDAT      ;LOAD EXPECTED
596 004714 012777 000010 174522      MOV      #BIT3,$STAT      ;LOAD MODE 2
597 004722 017737 174516 001126      MOV      $STAT,$BDDAT      ;READ REG
598 004730 105737 001126      TSTB     $BDDAT      ;TEST READY
599 004734 100402      BMI      2$      ;;BR IF SET
600 004736 104011      ERROR    11      ;ERROR, IN MODE 2 READY SHOULD NOT CLEAR
601      ;UNTIL DAC1 IS LOADED
602 004740 000431      BR       TST35      ;;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
616 005022 104011      ERROR    11      ;ERROR, READY FAILED TO SET
617      ; AFTER MODE 2 OPERATION
618      ;*****
(3)      ;*TEST 35      TEST WHEN ERASE IS SET, READY BIT CLEARS AND SET AFTER DELAY      (SW12=1)
(3)      ;*****
(2) 005024 000004
(1) 005026 012737 000001 001166      TST35: SCOPE
619 005034 032777 010000 174076      MOV      #1,$TIMES      ;;DO 1 ITERATION
620 005042 001430      BIT      #BIT12,$SWR      ;TEST BIT 12
621 005044 012700 000010      BEQ      TST36      ;;BYPASS IF NO STORAGE SCOPE
622 005050 005037 020432      MOV      #10,RO
623 005054 012777 002000 174362      CLR      TEMP      ;CLEAR DELAY
624 005062 052777 010000 174354      MOV      #BIT10,$STAT      ;SET STORE MODE
625 005070 105777 174350      BIS      #BIT12,$STAT      ;SET ERASE BIT
626 005074 100002      TSTB     $STAT      ;TEST THAT READY CLEARS
627 005076 104011      BPL      1$      ;;BR IF CLEARED
628 005100 000411      ERROR    11      ;ERROR, READY FAILED TO RESET
629 005102 105777 174336 1$:      BR       TST36      ;;BR TO SCOPE
630 005106 100406      TSTB     $STAT      ;TEST FOR READY
631 005110 005337 020432      BMI      TST36      ;;BR IF SET
632 005114 001372      DEC      TEMP      ;DELAY
633 005116 005300      BNE      1$      ;;BR IF NOT READY
634 005120 001370      DEC      RO      ;DECREMENT COUNTER
635 005122 104011      BNE      1$      ;;BR IF NOT DONE
636      ;ERROR, ERASE CLEARED READY AND FAILED
637      ; TO SET READY AFTER A DELAY
;SWR 12 = 1 AND NO STORAGE SCOPE CONNECTED ??

```

# M03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T36

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 19  
TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED -1

SEQ 0038

```
639          ;:*****
(3)          ;:TEST 36      TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED -1
(3)          ;:*****
(2) 005124 000004          †TST36: SCOPE
(1) 005126 012737 000040 001166      MOV      #40,STIMES      ;;DO 40 ITERATIONS
640 005134 012737 000340 177776      MOV      #340,PSW
641 005142 012777 005204 174306      MOV      #1$,DIV      ;SET INTERRUPT VECTOR
642 005150 012700 000400          MOV      #400,RO      ;SET UP DELAY
643 005154 013737 020440 177776      MOV      BRLEV2,PSW      ;LOAD CPU PSW WITH DEVICE LEVEL -1
644 005162 012777 000101 174254      MOV      #BIT6!BIT0,@STAT ;START DISPLAY
645 005170 005300          DEC      RO      ;DELAY
646 005172 001376          BNE     .-2
647 005174 005077 174244          CLR     @STAT      ;DO NOT LET INTERRUPT ENABLE SET
648 005200 104006          ERROR  6      ;ERROR, FAILED TO INTERRUPT
649 005202 000401          BR     2$      ;;NEXT TEST
650 005204 022626          1$: CMP     (SP)+,(SP)+ ;POP THE STACK
651 005206 013777 001460 174242      2$: MOV     IVS,@IV ;RESET VECTOR
652 005214 005077 174240          CLR     @IVS
653
654          ;:*****
(3)          ;:TEST 37      TEST THAT THE DISPLAY DOES NOT INTERRUPT AT LEVEL INDICATED
(3)          ;:*****
(2) 005220 000004          †TST37: SCOPE
(1) 005222 012737 000040 001166      MOV      #40,STIMES      ;;DO 40 ITERATIONS
655 005230 012737 000340 177776      MOV      #340,PSW
656 005236 012777 005272 174212      MOV      #1$,DIV      ;SET INTERRUPT VECTOR
657 005244 012700 004000          MOV      #400,RO      ;SET UP DELAY
658 005250 013737 020436 177776      MOV      BRLEV1,PSW      ;LOAD CPU PSW WITH DEVICE LEVEL
659 005256 012777 000101 174160      MOV      #BIT6!BIT0,@STAT ;START DISPLAY
660 005264 005300          DEC      RO      ;DELAY
661 005266 001376          BNE     .-2
662 005270 000404          BR     2$      ;;
663 005272 005077 174146          1$: CLR     @STAT      ;DO NOT LET INTERRUPT ENABLE SET
664 005276 104007          ERROR  7      ;ERROR INTERRUPTED IN ERROR
665 005300 000417          BR     TST40 ;NEXT TEST
666 005302 012777 005324 174146      2$: MOV     #3$,DIV ;LOAD RETURN VECTOR
667 005310 005037 177776          CLR     PSW      ;LOWER PSW
668 005314 005077 174124          CLR     @STAT      ;CLEAR INT ENABLE
669 005320 104007          ERROR  7      ;LOWERING THE PRIORITY FAILED TO ALLOW INTERRUPT
670 005322 000401          BR     4$      ;;NEXT TEST
671 005324 022626          3$: CMP     (SP)+,(SP)+
672 005326 013777 001460 174122      4$: MOV     IVS,@IV ;RESET VECTOR
673 005334 005077 174120          CLR     @IVS
```



# N03

MAINDEC-11-DZAAC-A  
DZAACA.P11 T40

RA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 20  
TEST THAT RESET CLEARS MODE, EXT. DELAY AND FAST INTENSIFY BITS

SEQ 0039

```
675      ;:*****  
(3)      ;*TEST 40      TEST THAT RESET CLEARS MODE, EXT. DELAY AND FAST INTENSIFY BITS  
(3)      ;:*****  
(2) 005340 000004      †TST40: SCOPE  
(1) 005342 012737 000040 001166      MOV      #40,$TIMES      ;;DO 40 ITERATIONS  
676 005350 012777 000036 174066      MOV      #BIT4!BIT3!BIT2!BIT1,$STAT  
677 005356 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED  
678 005364 000005      RESET  
679 005366 017737 174052 001126      MOV      $STAT,$BDDAT      ;READ STATUS  
680 005374 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE  
681 005402 001401      BEQ      TST41      ;;BR IF EQUAL  
682 005404 104001      ERROR      1      ;ERROR, RESET FAILED TO CLEAR STATUS REG  
683      ;:*****  
(3)      ;*TEST 41      TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU  
(3)      ;:*****  
(2) 005406 000004      †TST41: SCOPE  
(1) 005410 012737 000040 001166      MOV      #40,$TIMES      ;;DO 40 ITERATIONS  
684 005416 012777 007100 174020      MOV      #BIT11!BIT10!BIT9!BIT6,$STAT  
685 005424 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED  
686 005432 000005      RESET  
687 005434 017737 174004 001126      MOV      $STAT,$BDDAT      ;READ STATUS  
688 005442 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE  
689 005450 001401      BEQ      TST42      ;;BR IF EQUAL  
690 005452 104001      ERROR      1      ;ERROR, RESET FAILED TO CLEAR STATUS  
691      ;:*****  
(3)      ;*TEST 42      TEST THAT RESET CLEARS DAC0 REGISTER      (SW09=0)  
(3)      ;:*****  
(2) 005454 000004      †TST42: SCOPE  
(1) 005456 012737 000040 001166      MOV      #40,$TIMES      ;;DO 40 ITERATIONS  
692 005464 032777 001000 173446      BIT      #SW09,$SWR      ;TEST IF BIT 9 IS SET  
(3) 005472 001016      BNE      TST43      ;;BR IF SET  
693 005474 012777 177777 173744      MOV      #-1,$DAC0  
694 005502 005037 001124      CLR      $GDDAT      ;LOAD EXPECTED  
695 005506 000005      RESET  
696 005510 017737 173732 001126      MOV      $DAC0,$BDDAT      ;READ REG  
697 005516 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE  
698 005524 001401      BEQ      TST43      ;;BR IF EQUAL  
699 005526 104002      ERROR      2      ;ERROR, RESET FAILED TO CLEAR DAC 0 REGISTER  
700      ;:*****  
(3)      ;*TEST 43      TEST THAT RESET CLEARS DAC1 REGISTER      (SW09=0)  
(3)      ;:*****  
(2) 005530 000004      †TST43: SCOPE  
(1) 005532 012737 000040 001166      MOV      #40,$TIMES      ;;DO 40 ITERATIONS  
701 005540 032777 001000 173372      BIT      #SW09,$SWR      ;TEST IF BIT 9 IS SET  
(3) 005546 001016      BNE      TST44      ;;BR IF SET  
702 005550 012777 177777 173672      MOV      #-1,$DAC1  
703 005556 005037 001124      CLR      $GDDAT      ;LOAD EXPECTED  
704 005562 000005      RESET  
705 005564 017737 173660 001126      MOV      $DAC1,$BDDAT      ;READ REG  
706 005572 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE  
707 005600 001401      BEQ      TST44      ;;BR IF EQUAL  
708 005602 104003      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 EXPECTED
(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 EXPECTED
(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)      ::*TEST 50      TEST THAT RESET SET DAC2 TO 4000 (SW09=1)

```

```

(4)
(3) 006054 000004
(2) 006056 012737 000040 001166
(1) 006064 032777 001000 173046
(3) 006072 001417
(1) 006074 012777 003777 173350
(1) 006102 012737 004000 001124
(1) 006110 000005
(1) 006112 017737 173334 001126
(1) 006120 023737 001124 001126
(3) 006126 001401
(1) 006130 104004
741
(4)
(4)
(3) 006132 000004
(2) 006134 012737 000040 001166
(1) 006142 032777 001000 172770
(3) 006150 001417
(1) 006152 012777 003777 173274
(1) 006160 012737 004000 001124
(1) 006166 000005
(1) 006170 017737 173260 001126
(1) 006176 023737 001124 001126
(3) 006204 001401
(1) 006206 104005
742
(3)
(3)
(2) 006210 000004
(1) 006212 012737 000010 001166
743 006220 032777 002000 172712
744 006226 001024
745 006230 012777 000020 173206
746 006236 105277 173202
747 006242 012700 000000
748 006246 105777 173172
749 006252 100403
750 006254 005300
751 006256 001373
752 006260 000407
753 006262 012737 000020 001124
754 006270 017737 173150 001126
755 006276 104011
756
757

```

```

*****
↑TST50: SCOPE
MOV #40,STIMES ;;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
*****
↑TST51: SCOPE
MOV #40,STIMES ;;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
*****
↑TST52: SCOPE
MOV #10,STIMES ;;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)          ;*TEST 53      TEST THE EXTERNAL DELAY DOES SET DISPLAY READY SW10=1
(3)          ;*****
(2) 006300 000004          †T53: SCOPE
(1) 006302 012737 000010 001166      MOV      #10,$TIMES          ;;DO 10 ITERATIONS
760 006310 032777 002000 172622      BIT      @SW10,@SWR          ;TEST SR BIT 10
761 006316 001424          BEQ      T54              ;;BR IF CLEARED
762 006320 012777 000020 173116      MOV      @BIT4,@STAT        ;ENABLE EXT. DELAY
763 006326 012700 000000          MOV      #0,R0              ;LOAD COUNTER
764 006332 105277 173106          INCB     @STAT              ;ENABLE DISPLAY
765 006336 105777 173102          TSTB    @STAT              ;WAIT FOR READY
766 006342 100411          BMI     2$                ;BR IF DONE
767 006344 005300          DEC     R0                  ;DELAY
768 006346 001373          BNE     1$                  ;BR IF NOT DONE
769 006350 012737 000220 001124      MOV      @BIT7!BIT4,$GDDAT  ;LOAD EXPECTED VALUE
770 006356 017737 173062 001126      MOV      @STAT,$BDDAT      ;READ ACTUAL
771 006364 104011          ERROR   11                  ;EXT. DELAY FAILED TO SET READY
772          ;WHEN THE OPERATOR (VIA SW 10=1) SAID AN EXT. DELAY WAS
773 006366 000005          2$:  RESET                ;ENSURE CLEARED AA11-K
774
775          ;*****
(3)          ;*TEST 54      DETERMINE IF MORE AA11-K'S REMAIN TO BE TESTED
(3)          ;*****
(2) 006370 000004          †T54: SCOPE
(1) 006372 012737 000001 001166      MOV      #1,$TIMES          ;;DO 1 ITERATION
776 006400 000005          RESET
777 006402 005237 001206          INC     $UNIT              ;MORE UNITS?
778 006406 123737 001206 020442      CMPB    $UNIT,EVER
779 006414 001431          BEQ     SEOP                ;;BR IF NONE
780 006416 063737 001436 001444      ADD     VADDR,STAT          ;UPDATE ADDRESS
781 006424 063737 001436 001446      ADD     VADDR,DAC0
782 006432 063737 001436 001450      ADD     VADDR,DAC1
783 006440 063737 001436 001452      ADD     VADDR,DAC2
784 006446 063737 001436 001454      ADD     VADDR,DAC3
785 006454 063737 001440 001456      ADD     VVECT,IV            ;UPDATE VECTOR
786 006462 063737 001440 001460      ADD     VVECT,IVS
787 006470 005037 001102          CLR     $STNM
788 006474 000137 002466          JMP     LTEST                ;TEST ANOTHER AA11-K

```





```

805 006622 012706 001100          VISUAL: MOV      #STACK,SP          ;LOAD SP
806 006626 005737 020444          TST      EVER1              ;TEST IF TYPED ONCE
807 006632 001006                   BNE      PICO              ;BR IF YES
808 006634 104400                   TYPE
809 006636 015457                   MES6                      ;HEADER ABOUT SCOPE ADJ.
810 006640 104400                   TYPE
811 006642 015507                   MES15                     ;TEXT ABOUT SWR
812 006644 005237 020444          INC      EVER1             ;SET FLAG
813                                     .SBTTL  DISPLAY HORIZONTAL LINE
814 006650 013700 001446          PICO:  MOV      DAC0,R0
815 006654 013701 001450          MOV      DAC1,R1
816 006660 032777 000010 172252  BIT      #SW03,2SWR        ;TEST SR BIT 3
817 006666 001404                   BEQ      2$                ;BR IF DAC #0 AND #1
818 006670 013700 001452          MOV      DAC2,R0          ;USE DAC #2 AND 3
819 006674 013701 001454          MOV      DAC3,R1
820 006700 012737 000030 020430 2$:  MOV      #30,TICKS        ;LOAD TIMER FOR CP TYPE
821 006706 004737 014324          JSR      PC,CHTIME        ;CHANGE TIMER FOR CP TYPE
822 006712 004737 007002          1$:  JSR      PC,PBB
823 006716 004737 014210          JSR      PC,TIMER        ;DONE ?
824 006722 000773                   BR      1$
825                                     .SBTTL  DISPLAY A VERTICAL LINE
826 006724 013700 001450          PIC1:  MOV      DAC1,R0
827 006730 013701 001446          MOV      DAC0,R1
828 006734 032777 000010 172176  BIT      #SW03,2SWR        ;TEST SR BIT 3
829 006742 001404                   BEQ      2$                ;BR IF DAC #0 AND 1
830 006744 013700 001454          MOV      DAC3,R0          ;USE DAC #2 AND 3
831 006750 013701 001452          MOV      DAC2,R1
832 006754 012737 000030 020430 2$:  MOV      #30,TICKS        ;LOAD TIME
833 006762 004737 014324          JSR      PC,CHTIME        ;CHANGE TIMER FOR CP TYPE
834 006766 004737 007002          1$:  JSR      PC,PBB
835 006772 004737 014210          JSR      PC,TIMER        ;DONE ?
836 006776 000773                   BR      1$
837 007000 000443                   BR      PIC3
838 007002 005077 172436          PBB:  CLR      2STAT
839 007006 032777 002000 172124  BIT      #SW10,2SWR        ;TEST IS SW10 =1
840 007014 001403                   BEQ      10$              ;BR IF NOT
841 007016 052777 000020 172420  BIS      #BIT4,2STAT        ;ENABLE EXT. DELAY
842 007024 013704 001444          10$:  MOV      STAT,R4
843 007030 012703 007777          MOV      #7777,R3        ;SET HIGH LIMIT
844 007034 012702 000001          MOV      #1,R2           ;INITIALIZE INCREMENTS BETWEEN POINTS
845 007040 032777 001000 172072  BIT      #SW09,2SWR        ;TEST SW 9 = 1
846 007046 001013                   BNE      2$                ;BR IF YES
847 007050 012711 004000          MOV      #4000,(1)
848 007054 012710 000000          MOV      #0,(0)
849 007060 060210 000000          1$:  ADD      R2,(0)          ;INCREMENT
850 007062 005214                   INC      (4)              ;INTENSIFY
851 007064 105714                   TSTB    (4)
852 007066 100376                   BPL     #-2
853 007070 021003                   CMP     (0),R3            ;DONE ALL POINTS?
854 007072 001372                   BNE     1$                ;NO
855 007074 000207                   RTS     PC
856 007076 012711 000000          2$:  MOV      #0,(R1)        ;LOAD TWO'S COMP. ORIGIN
857 007102 012710 000000          MOV      #0,(R0)
858 007106 000764                   BR      1$

```

```

      .SBTTL  PINCUSHION TEST (DISPLAY SQUARE)
860
861
862 007110 005077 172330          PIC3:  CLR      2STAT
863 007114 012737 000140 020430  MOV      2140,TICKS
864 007122 004737 014324          JSR      PC,CHTIME
865 007126 013701 001446          MOV      DAC0,R1
866 007132 013702 001450          MOV      DAC1,R2
867 007136 032777 000010 171774  BIT      2SW03,2SWR      ;TEST SR BIT 3
868 007144 001404          BEQ      1$              ;BR IF DAC #0 AND 1
869 007146 013701 001452          MOV      DAC2,R1      ;USE DAC #2 AND 3
870 007152 013702 001454          MOV      DAC3,R2
871 007156 013703 001444          1$:  MOV      STAT,R3
872 007162 012704 000020          MOV      20,R4
873 007166 032777 001000 171744  P3:  BIT      2SW09,2SWR      ;TEST BIT 9
874 007174 001405          BEQ      1$
875 007176 012711 004000          MOV      24000,(R1)    ;LOAD TWO'S COMP ORIGIN
876 007202 012712 004000          MOV      24000,(R2)
877 007206 000402          BR       2$
878 007210 005011          1$:  CLR      (R1)        ;CLEAR AXIS
879 007212 005012          CLR      (R2)
880          ;DRAW BOTTOM LINE
881 007214 012700 000377          2$:  MOV      2377,R0
882 007220 032777 002000 171712  BIT      2SW10,2SWR      ;TEST IF SW10 =1
883 007226 001403          BEQ      P3A            ;BR IF NOT
884 007230 052777 000020 172206  BIS      2BIT4,2STAT    ;ENABLE EXT. DELAY
885 007236 060411          P3A:  ADD      R4,(1)
886 007240 005213          INC      (R3)          ;INTENSIFY
887 007242 105713          TSTB    (3)
888 007244 100376          BPL     -2             ;WAIT FOR READY
889 007246 005300          DEC     R0
890 007250 001372          BNE     P3A            ;NO
891          ;DRAW RIGHT LINE
892 007252 012700 000377          MOV      2377,R0
893 007256 060412          P3B:  ADD      R4,(2)
894 007260 005213          INC      (R3)          ;INTENSIFY
895 007262 105713          TSTB    (3)
896 007264 100376          BPL     -2             ;WAIT FOR READY
897 007266 005300          DEC     R0
898 007270 001372          BNE     P3B            ;NO
899          ;DRAW TOP LINE
900 007272 012700 000377          MOV      2377,R0
901 007276 160411          P3C:  SUB      R4,(1)
902 007300 005213          INC      (R3)          ;INTENSIFY
903 007302 105713          TSTB    (3)
904 007304 100376          BPL     -2             ;WAIT FOR READY
905 007306 005300          DEC     R0
906 007310 001372          BNE     P3C            ;NO
907          ;DRAW LEFT LINE
908 007312 012700 000377          MOV      2377,R0
909 007316 160412          P3D:  SUB      R4,(2)
910 007320 005213          INC      (R3)          ;INTENSIFY
911 007322 105713          TSTB    (3)
912 007324 100376          BPL     -2             ;WAIT FOR READY
913 007326 005300          DEC     R0
914 007330 001372          BNE     P3D            ;NO
915 007332 004737 014210          JSR      PC,TIMER
    
```



# H04

```

916 007336 000713 BR P3
917 .SBTTL PLOT AN X
918
919 007340 005077 172100 PIC4: CLR @STAT
920 007344 012737 000100 020430 MOV #100,TICKS
921 007352 004737 014324 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
922 007356 013701 001446 PIC4B: MOV DAC0,R1
923 007362 013702 001450 MOV DAC1,R2
924 007366 032777 000010 171544 BIT #SW03,@SWR ;TEST SWR BIT 3
925 007374 001404 BEQ 1$ ;BR IF DAC #0 AND 1
926 007376 013701 001452 MOV DAC2,R1 ;USE DAC #2 AND 3
927 007402 013702 001454 MOV DAC3,R2
928 007406 013703 001444 1$: MOV STAT,R3
929 007412 012704 000004 MOV #4,R4
930 007416 032777 001000 171514 P4: BIT #SW09,@SWR ;TEST BIT 9
931 007424 001405 BEQ 1$
932 007426 012712 004000 MOV #4000,(R2)
933 007432 012711 004000 MOV #4000,(R1)
934 007436 000402 BR 2$
935 007440 005012 1$: CLR (R2)
936 007442 005011 CLR (R1)
937
938 ;PLOT LINE BEGINNING IN LOWER LEFT CORNER
939 007444 012700 001777 2$: MOV #1777,R0
940 007450 032777 002000 171462 BIT #SW10,@SWR ;TEST IF SW10 =1
941 007456 001403 BEQ P4A ;BR IF NOT
942 007460 052777 000020 171756 BIS #BIT4,@STAT ;ENABLE EXT. DELAY
943 007466 005213 P4A: INC (R3) ;INTENSIFY
944 007470 105713 TSTB (3)
945 007472 100376 BPL -2
946 007474 060412 ADD R4,(2) ;+4 TO Y
947 007476 060411 ADD R4,(1) ;+4 TO X
948 007500 005300 DEC R0
949 007502 001371 BNE P4A ;NO
950 007504 105713 TSTB (3)
951 007506 100376 BPL -2
952 ;PLOT LINE BEGINNING IN UPPER LEFT CORNER
953 007510 012712 007774 MOV #7774,(2)
954 007514 005011 CLR (1)
955 007516 012700 001777 P4B: MOV #1777,R0
956 007522 005213 INC (R3) ;INTENSIFY
957 007524 105713 TSTB (3)
958 007526 100376 BPL -2
959 007530 160412 SUB R4,(2) ;-4 TO Y
960 007532 060411 ADD R4,(1) ;+4 TO X
961 007534 005300 DEC R0
962 007536 001371 BNE P4B ;NO
963 007540 004737 014210 JSR PC,TIMER
964 007544 000724 BR P4
  
```

```

966
967
968
969 007546 012737 000060 020430 PIC5: MOV #60,TICKS ;LOAD TIMER
970 007554 004737 014324 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
971
972 007560 005077 171660 1S: CLR @STAT ;CLEAR STATUS
973 007564 005077 171656 CLR @DAC0 ;CLEAR X AXIS
974 007570 005077 171654 CLR @DAC1 ;CLEAR Y AXIS
975 007574 005077 171652 CLR @DAC2
976 007600 005077 171650 CLR @DAC3
977 007604 032777 002000 171326 BIT #SW10,@SWR ;TEST SW BIT 10
978 007612 001403 BEQ 10$ ;BR IF NOT
979 007614 052777 000020 171622 BIS #BIT4,@STAT ;ENABLE EXT. DELAY
980 007622 032777 000010 171310 10$: BIT #SW03,@SWR ;TEST SR BIT 3
981 007630 001405 BEQ 2$ ;BR IF DAC #0 AND 1
982 007632 013700 001454 MOV DAC3,R0 ;LOAD Y DAC
983 007636 013701 001452 MOV DAC2,R1 ;LOAD X DAC
984 007642 000404 BR 3$
985 007644 013700 001450 2$: MOV DAC1,R0 ;LOAD Y DAC
986 007650 013701 001446 MOV DAC0,R1 ;LOAD X DAC
987 007654 032777 000200 171256 3$: BIT #SW07,@SWR ;TEST X OR Y TEST BIT
988 007662 001403 BEQ 4$ ;NORMAL
989 007664 010003 MOV R0,R3 ;REVERSE THE AXIS
990 007666 010100 MOV R1,R0
991 007670 010301 MOV R3,R1
992 007672 013703 001444 4$: MOV STAT,R3 ;LOAD STATUS REG
993 007676 032777 001000 171234 5$: BIT #SW09,@SWR ;TEST TWO'S COMP SWITCH
994 007704 001007 BNE 6$
995 007706 004537 007754 JSR R5,LODPNT ;LOAD A LINE
996 007712 000000 O
997 007714 004537 007754 JSR R5,LODPNT ;LOAD A LINE
998 007720 007777 7777
999 007722 000406 BR 7$
1000 007724 004537 007754 6$: JSR R5,LODPNT ;LOAD A LINE
1001 007730 004000 4000
1002 007732 004537 007754 JSR R5,LODPNT
1003 007736 003777 3777
1004 007740 005710 7$: TST (R0) ;END
1005 007742 001355 BNE 5$ ;BR IF NOT
1006 007744 004737 014210 JSR PC,TIMER ;TEST TIME
1007 007750 000703 BR 1$
1008 007752 000414 BR PIC6
1009
1010 007754 012702 000200 LODPNT: MOV #200,R2
1011 007760 011511 1$: MOV (R5),(R1) ;LOAD AXIS
1012 007762 005213 INC (R3) ;INTENSIFY
1013 007764 105713 2$: TSTB (R3) ;DONE
1014 007766 100376 BPL 2$ ;WAIT
1015 007770 062710 000002 ADD #2,(R0) ;UPDATE AXIS
1016 007774 005302 DEC R2 ;DONE
1017 007776 001370 BNE 1$ ;BR IF NOT
1018 010000 005725 TST (R5)+ ;UPDATE
1019 010002 000205 RTS R5 ;EXIT

```







1116						
1117	010516	176	021	021	A:	.BYTE 176,21,21,21,176
	010521	021	176			
1118	010523	177	111	111	B:	.BYTE 177,111,111,111,66
	010526	111	066			
1119	010530	076	101	101	C:	.BYTE 76,101,101,101,42
	010533	101	042			
1120	010535	177	101	101	D:	.BYTE 177,101,101,101,76
	010540	101	076			
1121	010542	177	111	111	E:	.BYTE 177,111,111,111,101
	010545	111	101			
1122	010547	177	011	011	F:	.BYTE 177,11,11,11,1
	010552	011	001			
1123	010554	076	101	121	G:	.BYTE 76,101,121,121,62
	010557	121	062			
1124	010561	177	010	010	H:	.BYTE 177,10,10,10,177
	010564	010	177			
1125	010566	000	101	177	I:	.BYTE 0,101,177,101,0
	010571	101	000			
1126	010573	060	100	100	J:	.BYTE 60,100,100,100,77
	010576	100	077			
1127	010600	177	010	024	K:	.BYTE 177,10,24,42,101
	010603	042	101			
1128	010605	177	100	100	L:	.BYTE 177,100,100,100,100
	010610	100	100			
1129	010612	177	004	010	M:	.BYTE 177,4,10,4,177
	010615	004	177			
1130	010617	177	004	010	N:	.BYTE 177,4,10,20,177
	010622	020	177			
1131	010624	076	101	101	O:	.BYTE 76,101,101,101,76
	010627	101	076			
1132	010631	177	011	011	P:	.BYTE 177,11,11,11,6
	010634	011	006			
1133	010636	076	101	121	Q:	.BYTE 76,101,121,141,176
	010641	141	176			
1134	010643	177	011	031	R:	.BYTE 177,11,31,51,106
	010646	051	106			
1135	010650	046	111	111	S:	.BYTE 46,111,111,111,62
	010653	111	062			
1136	010655	001	001	177	T:	.BYTE 1,1,177,1,1
	010660	001	001			
1137	010662	077	100	100	U:	.BYTE 77,100,100,100,77
	010665	100	077			
1138	010667	037	040	100	V:	.BYTE 37,40,100,40,37
	010672	040	037			
1139	010674	177	020	010	W:	.BYTE 177,20,10,20,177
	010677	020	177			
1140	010701	143	024	010	X:	.BYTE 143,24,10,24,143
	010704	024	143			
1141	010706	003	004	170	Y:	.BYTE 3,4,170,4,3
	010711	004	003			
1142	010713	141	121	111	Z:	.BYTE 141,121,111,105,103
	010716	105	103			
1143	010720	076	121	111	NO:	.BYTE 76,121,111,105,76
	010723	105	076			
1144	010725	000	102	177	NI:	.BYTE 0,102,177,100,0

M04

MAINDEC-11-DZAC-A  
DZACA.P11

AA11-K DIAGNOSTIC  
PLOT CHARACTER SET

MACY11 27(732) 20-AUG-76 09:27 PAGE 29-1

SEQ 0051

1145	010730	100	000			
	010732	142	121	111	N2:	.BYTE 142,121,111,105,102
	010735	105	102			
1146	010737	042	101	111	N3:	.BYTE 42,101,111,111,66
	010742	111	066			
1147	010744	030	024	022	N4:	.BYTE 30,24,22,177,20
	010747	177	020			
1148	010751	047	105	105	N5:	.BYTE 47,105,105,105,71
	010754	105	071			
1149	010756	076	111	111	N6:	.BYTE 76,111,111,111,62
	010761	111	062			
1150	010763	101	041	021	N7:	.BYTE 101,41,21,11,7
	010766	011	007			
1151	010770	066	111	111	N8:	.BYTE 66,111,111,111,66
	010773	111	066			
1152	010775	046	111	111	N9:	.BYTE 46,111,111,111,76
	011000	111	076			
1153	011002	000	000	000	SPACEA:	.BYTE 0,0,0,0,0
	011005	000	000			
1154	011007	000	000	000		.BYTE 0,0,0,0,0
	011012	000	000			
1155	011014	000	000	000		.BYTE 0,0,0,0,0
	011017	000	000			
1156						.EVEN
1157	011022					.SBTTL
1158						CHANNEL 1 AND CHANNEL 2 TEST



1160	011022	012737	000200	020430	PIC7:	MOV	#200,TICKS	;SET UP A TIMER
1161	011030	004737	014324			JSR	PC,CHTIME	;CHANGE TIMER FOR CP TYPE
1162	011034	013737	001446	010502		MOV	DAC0,VCXTMP	;LOAD THE X AXIS
(1)	011042	013737	001450	010504		MOV	DAC1,VCYTMP	;LOAD THE Y AXIS
(1)	011050	032777	000010	170062		BIT	#SW03,ASWR	;TEST SR BIT 3
(1)	011056	001406				BEQ	1\$	;BR IF DAC #0 AND 1
(1)	011060	013737	001452	010502		MOV	DAC2,VCXTMP	;LOAD THE X AXIS
(1)	011066	013737	001454	010504		MOV	DAC3,VCYTMP	
(1)	011074	000240			1\$:	NOP		
1163	011076	013705	001444			MOV	STAT,R5	;SET UP R5 FOR STATUS REGISTER POINTER
1164	011102	012777	000000	170334	PIC7AA:	MOV	#0,ASAT	;SET UP SCOPE CONTROL
1165	011110	012737	002000	010510		MOV	#2000,XPOS	;LOAD X POSITION
1166	011116	012737	005000	010506		MOV	#5000,YPOS	;LOAD Y POSITION
1167	011124	032777	001000	170006		BIT	#SW09,ASWR	;TEST BIT 9
1168	011132	001406				BEQ	1\$	
1169	011134	012737	006000	010510		MOV	#6000,XPOS	;LOAD 2'S COMP ORIGIN
1170	011142	012737	001000	010506		MOV	#1000,YPOS	
1171	011150	012737	000011	020450	1\$:	MOV	#9,P7CNT	;SAVE THE NUMBER OF CHARACTERS
1172	011156	012737	020454	020452		MOV	#CH01,P7PNT	;SAVE CHANNEL 1 POINTER
1173	011164	017702	007262		PIC7A:	MOV	AP7PNT,R2	;MOVE MESSAGE POINTER INTO R2 FOR DISPLAY ROUTINE
1174	011170	004737	010324			JSR	PC,CHAR	;DISPLAY A CHARACTER
1175	011174	062737	000002	020452		ADD	#2,P7PNT	;ADD 2 TO THE MESSAGE POINTER
1176	011202	005337	020450			DEC	P7CNT	;DECREMENT CHARACTER COUNT
1177	011206	001366				BNE	PIC7A	;NOT FINISHED WITH ALL CHARACTERS
1178	011210	012777	001000	170226		MOV	#1000,ASAT	
1179	011216	012737	002000	010510		MOV	#2000,XPOS	;SET UP X POS FOR CHANNEL 2
1180	011224	012737	003000	010506		MOV	#3000,YPOS	;SET UP Y
1181	011232	032777	001000	167700		BIT	#SW09,ASWR	;TEST BIT 9
1182	011240	001406				BEQ	1\$	
1183	011242	012737	006000	010510		MOV	#6000,XPOS	;LOAD 2'S COMP ORIGIN
1184	011250	012737	007000	010506		MOV	#7000,YPOS	
1185	011256	012737	000011	020450	1\$:	MOV	#9,P7CNT	;SET UP CHARACTER COUNT
1186	011264	012737	020476	020452		MOV	#CH02,P7PNT	;SET UP CHANNEL 2 POINTER
1187	011272	017702	007154		PIC7B:	MOV	AP7PNT,R2	;SET UP
1188	011276	004737	010324			JSR	PC,CHAR	;DISPLAY A CHARACTER
1189	011302	062737	000002	020452		ADD	#2,P7PNT	;ADD 2 TO THE POINTER
1190	011310	005337	020450			DEC	P7CNT	;DECREMENT COUNT
1191	011314	001366				BNE	PIC7B	;NOT FINISHED
1192	011316	004737	014210			JSR	PC,TIMER	;CHECK THE RUNTIME OF THIS ROUTINE
1193	011322	000667				BR	PIC7AA	;NOT FINISHED
1194	011324	000400				BR	PIC12	

1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238

011326 012737 000002 020430  
011334 004737 013610  
011340 004737 013726  
011344 004737 014210  
011350 000771  
011352 004737 013610  
011356 000005  
011360 000137 006650  
  
011364 012706 001100  
011370 004737 001742  
011374 032777 002000 167536  
011402 001403  
011404 052777 000020 170032  
011412 013700 001446  
011416 004737 011454  
011422 013700 001450  
011426 004737 011454  
011432 013700 001452  
011436 004737 011454  
011442 013700 001454  
011446 004737 011454  
011452 000757  
  
011454 005010  
011456 062710 000010  
011462 005710  
011464 001374  
011466 004737 013332  
011472 000207  
011474 000240  
011476 000240

.SBTTL PHOSPHOR TEST  
PIC12: MOV #2,TICKS  
1\$: JSR PC,CLRVC  
JSR PC,LOADVC  
JSR PC,TIMER  
BR 1\$  
JSR PC,CLRVC  
RESET  
JMP PICO  
  
.SBTTL MANUAL DISPLAY ROUTINE  
FULRMP: MOV #STACK,SP  
JSR PC,LDTRAP  
BIT #SW10,JSWR  
BEQ 1\$  
BIS #BIT4,STAT  
1\$: MOV DAC0,R0  
JSR PC,10\$  
MOV DAC1,R0  
JSR PC,10\$  
MOV DAC2,R0  
JSR PC,10\$  
MOV DAC3,R0  
JSR PC,10\$  
BR 1\$  
  
10\$: CLR (R0)  
11\$: ADD #10,(R0)  
TST (R0)  
BNE 11\$  
JSR PC,ACTRLC  
RTS PC  
NOP  
NOP

;ERASE THE SCREEN  
;LOAD THE SCREEN  
;CHECK THE TIME  
  
;LOAD POINTER  
;LOAD BUS ADDRESS  
;TEST SR 10=1  
;BR IF NOT  
;ENABLE EXT. DELAY  
;GET BUS ADDRESS  
;LOAD THE RAMP ON DAC #1  
;GET BUS ADDRESS  
;LOAD THE RAMP ON DAC #1  
;GET BUS ADDRESS  
;LOAD THE RAMP ON DAC #2  
;GET THE BUS ADDRESS  
;LOAD THE RAMP ON DAC #3  
;BR BACK  
  
;CLEAR DAC  
;UPDATE THE DATA  
;TEST IF DONE  
;BR IF NOT  
;TEST FOR CTRL C  
;EXIT



```

1240 .SBTTL AUTO CALIBRATION
1241 011500 012706 001100 AUTCAL: MOV      #STACK, SP      ;LOAD STACK POINTER
1242 011504 004737 001742 JSR      PC, LDTRAP      ;LOAD TRAN AND ADDRESSES
1243 011510 104400 TYPE
1244 011512 017353 AUTOCL ;TELL OPERATOR
1245 011514 104400 TYPE
1246 011516 017061 SELD01
1247
1248 ;LOAD BUS TRAP FOR OPERTOR FALSE SELECTION
1249
1250 011520 012737 013150 000004 MOV      #AUTRAP, ERRVEC
1251 011526 012737 000054 001102 MOV      #STN-1, $STNM   ;LOAD TEST NUMBER
1252 011534 012737 011560 001110 MOV      #PSUPPLY, SLPERR ;LOAD LOOP RETURN
1253 011542 012737 011560 001106 MOV      #PSUPPLY, SLPADR ;LOAD LOOP ADDRESS
1254
1255 JSR      RS, CROSS
1256 011550 004537 014146 BIT13 ;CHANGE RELAYS
1257 011554 J20000 ;USING BIT 13 FOR CRISS-CROSS WRAP MODE
1258
1259 *****
1260 (3) ;*TEST 55 TEST THAT CH 3 IS AT +2.5 BIPOLAR
1261 (3) *****
1262 (2) †ST55: SCOPE
1263 011556 000004 PSUPPLY: MOV      V1754, $GDDAT ;LOAD EXPECTED VALUE
1264 011560 013737 014570 001124 JSR      RS, CONVRT ;CONVERT
1265 011566 004537 014444 CH3 ;CH 3
1266 011572 000003 MOV      ADEND, $BDDAT ;LOAD VALUE READ
1267 011574 013737 014602 001126 MOV      V24, $SPREAD ;LOAD + OR - COUNT SPREAD
1268 011602 013737 014572 014610 JSR      PC, COMPAR ;COMPARE $GDDAT AND $BDDAT
1269 011610 004737 014612 BR      NSUPPLY ;;BR IF WITHIN TOLERANCE
1270 011614 000402 ERROR 12 ;CH 3 FAILED TO EQUAL EXPECTED VALUE
1271 011616 104012
1272
1273 *****
1274 (3) ;*TEST 56 TEST THAT CH 4 IS AT -2.5 BIPOLAR
1275 (3) *****
1276 (2) †ST56: SCOPE
1277 011620 000004 NSUPPLY: MOV      V24, $GDDAT ;LOAD EXPECTED VALUE
1278 011622 013737 014572 001124 JSR      RS, CONVRT ;CONVERT
1279 011630 004537 014444 CH4 ;CH 4
1280 011634 000004 MOV      ADEND, $BDDAT ;LOAD VALUE READ
1281 011636 013737 014602 001126 MOV      V24, $SPREAD ;LOAD + OR - COUNT SPREAD
1282 011644 013737 014572 014610 JSR      PC, COMPAR ;COMPARE $GDDAT AND $BDDAT
1283 011652 004737 014612 BR      SUPOK ;;BR IF WITHIN TOLERANCE
1284 011656 000402 ERROR 12 ;CH 4 FAILED TO EQUAL EXPECTED VALUE
1285 011660 104012
1286 011662 000422
1287 011664 SUPOK: BR      TST57 ;;
1288 (1) 011664 004537 014146 JSR      RS, CROSS ;LOAD THE RELAYS WITH BIT13
1289 (1) 011670 020000 BIT13
1290 011672 104400 011700 TYPE 65$ ;;TYPE ASCIZ STRING
1291 (1) 011676 000414 BR      64$ ;;GET OVER THE ASCIZ
1292 (1) 011730 ;;65$: .ASCIZ <15><12>/ANALOG SUPPLIES OK/<15><12>
1293 64$:

```



D05

MAINDEC-11-DZAAC-A  
DZAACA.P11 TS7

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 33  
TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH

SEQ 0055

```

1320      ;*****
      (3) ;*TEST 57      TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
      (3) ;*****
      (2) 011730 000004
1321 011732 012777 000000 167504   TST57: SCOPE
1322 011740 012737 001146 001124   MOV      #0, @STAT      ;LOAD STATUS
1323 011746 004537 014444   MOV      #1146, $GDDAT   ;LOAD EXPECTED VALUE
1324 011752 000053   JSR      RS, CONVRT     ;CONVERT
1325 011754 013737 014602 001126   CH53     ;CH 53
1326 011762 023737 001124 001126   MOV      ADEND, $BDDAT   ;LOAD VALUE READ
1327 011770 003401   CMP      $GDDAT, $BDDAT ;COMPARE
1328 011772 104014   BLE     TST60           ;;BR IF LESS THAN OR EQUAL
1329                                     ;CH 53 FAILED TO EQUAL EXPECTED VALUE
1330      ;*****
      (3) ;*TEST 60      TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
      (3) ;*****
      (2) 011774 000004
1331 011776 005077 167442   TST60: SCOPE
1332 012002 012777 012000 167434   CLR      @STAT          ;CLEAR STATUS
1333 012010 012737 000120 001124   MOV      #BIT12!BIT10, @STAT ;SET ERASE
1334 012016 004537 014444   MOV      #120, $GDDAT    ;LOAD EXPECTED VALUE
1335 012022 000053   JSR      RS, CONVRT     ;CONVERT
1336 012024 013737 014602 001126   CH53     ;CH 53
1337 012032 023737 001124 001126   MOV      ADEND, $BDDAT   ;LOAD VALUE READ
1338 012040 002001   CMP      $GDDAT, $BDDAT ;COMPARE
1339 012042 104015   BGE     TST61           ;;BR IF GREATER THAN OR EQUAL
1340                                     ;CH 53 FAILED TO EQUAL EXPECTED VALUE

```

E05

MAINDEC-11-DZAAC-A  
DZAACA.P11 T61

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 34  
TEST THAT WRITE-THRU (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW

SEQ 0056

```

1342 (3) *****
(3) *TEST 61 TEST THAT WRITE-THRU (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
(2) C 2044 000004 *****
1343 012046 012777 006000 167370 †ST61: SCOPE ;SET WRITE-THRU
1344 012054 012737 000120 001124 MOV #BIT11!BIT10,STAT ;LOAD EXPECTED VALUE
1345 012062 004537 014444 JSR R5,CONVRT ;CONVERT
1346 012066 000053 CH53 ;CH 54
1347 012070 013737 014602 001126 MOV ADEND,SBDDAT ;LOAD VALUE READ
1348 012076 023737 001124 001126 CMP SGDDAT,SBDDAT ;COMPARE
1349 012104 002001 BGE TST62 ;;BR IF GREATER THAN OR EQUAL
1350 012106 104015 ERROR 15 ;CH 54 FAILED TO EQUAL EXPECTED VALUE
1351 (3) *****
(3) *TEST 62 TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
(2) 012110 000004 *****
1352 012112 005077 167326 †ST62: SCOPE ;CLEAR STORE MODE
1353 012116 012737 000120 001124 CLR STAT ;LOAD EXPECTED VALUE
1354 012124 004537 014444 JSR R5,CONVRT ;CONVERT
1355 012130 000055 CH55 ;CH 55
1356 012132 013737 014602 001126 MOV ADEND,SBDDAT ;LOAD VALUE READ
1357 012140 023737 001124 001126 CMP SGDDAT,SBDDAT ;COMPARE
1358 012146 002001 BGE TST63 ;;BR IF GREATER THAN OR EQUAL
1359 012150 104015 ERROR 15 ;CH 53 FAILED TO EQUAL EXPECTED VALUE
1360 (3) *****
(3) *TEST 63 TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
(2) 012152 000004 *****
1361 012154 012777 017000 167262 †ST63: SCOPE ;SET STORE MODE
1362 012162 012737 001146 001124 MOV #BIT12!BIT11!BIT10!BIT9,STAT ;LOAD EXPECTED VALUE
1363 012170 004537 014444 JSR R5,CONVRT ;CONVERT
1364 012174 000055 CH55 ;CH 55
1365 012176 013737 014602 001126 MOV ADEND,SBDDAT ;LOAD VALUE READ
1366 012204 023737 001124 001126 CMP SGDDAT,SBDDAT ;COMPARE
1367 012212 003401 BLE TST64 ;;BR IF LESS THAN OR EQUAL
1368 012214 104014 ERROR 14 ;CH 55 FAILED TO EQUAL EXPECTED VALUE
1369 (3) *****
(3) *TEST 64 TEST THAT DELAY RETURN SETS READY
(2) 012216 000004 *****
1370 012220 012777 000020 167216 †ST64: SCOPE ;LOAD EXT DELAY BIT
1371 012226 012737 000020 001124 MOV #BIT4,STAT ;LOAD EXPECTED
1372 012234 005277 167204 INC STAT ;MAKE READY GO AWAY
1373 012240 017737 167200 001126 MOV STAT,SBDDAT ;READ STAT
1374 012246 023737 001124 001126 CMP SGDDAT,SBDDAT ;COMPARE
1375 012254 001401 BEQ 15 ;;BR IF EXPECTED
1376 012256 104001 ERROR 1 ;READY FAILED TO CLEAR
1377 012260 052777 002000 167156 15: BIS #BIT10,STAT ;LOAD NON-STORE L
1378 012266 012737 002220 001124 MOV #BIT10!BIT7!BIT4,SGDDAT ;LOAD EXPECTED
1379 012274 017737 167144 001126 MOV STAT,SBDDAT ;READ THE REGISTER
1380 012302 023737 001124 001126 CMP SGDDAT,SBDDAT ;COMPARE
1381 012310 001401 BEQ 25 ;;BR IF SAME
1382 012312 104001 ERROR 1 ;EXT. DELAY RETURN FAILED TO SET READY
1383 012314 005077 167124 25: CLR STAT

```



# F05

MAINDEC-11-DZAAC-A  
DZAACA.P11 T65

RA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 35  
TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH

SEQ 0057

```

1385      ;:*****
(3)      ;:TEST 65      TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
(3)      ;:*****
(2) 012320 000004
1386 012322 012777 014000 167114
1387 012330 012737 001100 001124
1388 012336 004537 014444
1389 012342 000056
1390 012344 013737 014602 001126
1391 012352 023737 001124 001126
1392 012360 003401
1393 012362 104014
1394
1395
1396      ;:*****
(3)      ;:TEST 66      TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
(3)      ;:*****
(2) 012364 000004
1397 012366 012777 003000 167050
1398 012374 012737 000120 001124
1399 012402 004537 014444
1400 012406 000056
1401 012410 013737 014602 001126
1402 012416 023737 001124 001126
1403 012424 002001
1404 012426 104015
1405
1406 012430 005077 167010
1407

```

```

TST65: SCOPE
MOV      #BIT12:BIT11,@STAT      ;LOAD STATUS
MOV      #1100,$GDDAT           ;LOAD EXPECTED VALUE
JSR      R5,CONVRT              ;CONVERT
CH56                                ;CH 56
MOV      ADEND,$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

TST66: SCOPE
MOV      #BIT9:BIT10,@STAT      ;SET CHANNEL 2
MOV      #120,$GDDAT           ;LOAD EXPECTED VALUE
JSR      R5,CONVRT              ;CONVERT
CH56                                ;CH 56
MOV      ADEND,$BDDAT           ;LOAD VALUE READ
CMP      $GDDAT,$BDDAT          ;COMPARE
BGE      15                      ;;BR IF GREATER THAN
ERROR    15                      ;CH 56 FAILED TO EQUAL EXPECTED
; VALUE
15: CLR      @STAT              ;CLEAR BIT 9

```



# G05

MAINDEC-11-DZAAC-A  
DZAACA.P11 T67

AA11-K DIAGNOSTIC MACY11 27(732) 20-AUG-76 09:27 PAGE 36  
TEST THAT "ERASE" AND DONE CAN BE CLEARED AND SET

SEQ 0058

```

1409          ;*****
(3)          ;*TEST 67          TEST THAT "ERASE" AND DONE CAN BE CLEARED AND SET
(3)          ;*****
(2) 012434 000004          TST67: SCOPE
1410 012436 005077 167002 CLR          @STAT          ;CLEAR
1411 012442 005037 001124 CLR          $GDDAT          ;CLEAR EXPECTED
1412 012446 012777 010000 166770 MOV          #BIT12,@STAT          ;LOAD EXPECTED
1413 012454 017737 166764 001126 MOV          @STAT,$BDDAT          ;READ REG
1414 012462 023737 001124 001126 CMP          $GDDAT,$BDDAT          ;COMPARE
1415 012470 001401          BEQ          1$          ;;BR IF SET
1416 012472 104001          ERROR          1          ;ERASE FAILED TO CLEAR READY, CHECK G5036-
1417          ;BCOBR CONNECTION: A-VV, VV-A
1418 012474 052777 001000 002076 1$: BIS          #BIT9,@GSTAT          ;SET CH 2 <GENERATE ERASE RETURN ON GOOD>
1419 012502 005037 001124 CLR          $GDDAT          ;LOAD EXPECTED
1420 012506 017737 166732 001126 MOV          @STAT,$BDDAT          ;READ REG
1421 012514 023737 001124 001126 CMP          $GDDAT,$BDDAT          ;COMPARE
1422 012522 001401          BEQ          2$          ;;BR IF CLEARED
1423 012524 104001          ERROR          1          ;ERASE BIT FAILED TO CLEAR BY ERASE
1424          ;RETURN (SETTING OF THE CH 2 BIT)
1425
1426 012526 042777 001000 002044 2$: BIC          #BIT9,@GSTAT          ;CLEAR CH 2
1427 012534 012737 000200 001124 MOV          #BIT7,$GDDAT          ;LOAD EXPECTED
1428 012542 017737 166676 001126 MOV          @STAT,$BDDAT          ;READ REG
1429 012550 023737 001124 001126 CMP          $GDDAT,$BDDAT          ;COMPARE
1430 012556 001401          BEQ          3$          ;;BR IF EQUAL
1431 012560 104001          ERROR          1          ;READY FAILED TO SET ON THE END
1432          ; OF ERASE RETURN (CLEARING CH 2)
1433
1434 012562          3$: JSR          RS,CROSS          ;LOAD THE RELAYS WITH 0
(1) 012562 004537 014146          0
(1) 012566 000000

```

```

1436      ;*****
(3)      ;*TEST 70      ADJUSTMENT ROUTINES FOR DAC 0 - 1
(3)      ;*****
(2) 012570 000004
(1) 012572 012737 000001 001166
1437      ;ST70: SCOPE
1438 012600 004537 014054      ;DO 1 ITERATION
1439 012604 020235      ;OFFSET ADJUST FOR DAC 0
1440 012606 005077 166634      JSR      R5,SNDVLT      ;LOAD THE VOLTAGE
1441 012612 004537 014146      CLR      @DAC0      ;LOAD DAC REGISTER
(1) 012616 060000      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
1442 012620 104400      BIT14!BIT13
1443 012622 017565      TYPE
1444 012624 004737 013374      R38      ;TELL OPR. TO ADJUST R38
1445      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1446      ;OFFSET ADJUSTMENT FOR DAC 1
1447 012630 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 012634 020000      BIT13
1448 012636 005077 166606      CLR      @DAC1      ;LOAD THE DAC REGISTER
1449 012642 104400      TYPE
1450 012644 017632      R40      ;TELL OPR. TO ADJUST R40
1451 012646 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1452
1453      ;GAIN ADJUST FOR DAC 0
1454 012652 004537 014054      JSR      R5,SNDVLT
1455 012656 020250      P49976
1456 012660 012777 007777 166560      MOV      #7777,@DAC0      ;LOAD DAC 0
1457 012666 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
(1) 012672 060000      BIT14!BIT13
1458 012674 104400      TYPE
1459 012676 020011      R37      ;TELL OPR. TO ADJUST R37
1460 012700 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1461
1462      ;GAIN ADJUST FOR DAC 1
1463 012704 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 012710 020000      BIT13
1464 012712 012777 007777 166530      MOV      #7777,@DAC1      ;LOAD DAC 1
1465 012720 104400      TYPE
1466 012722 020056      R39      ;TELL OPR. TO ADJUST R39
1467 012724 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1468
1469      ;DAC 0 D/A FUNCTION CHECK
1470 012730 004537 014352      JSR      R5,FUNDAC      ;GO TO DAC FUNCTION SUB.
1471 012734 001446      DAC0
1472 012736 000051      CH51
1473
1474      ;DAC 1 D/A FUNCTION CHECK
1475 012740 004537 014352      JSR      R5,FUNDAC
1476 012744 001450      DAC1
1477 012746 000052      CH52
1478

```

```

1480      ;*****
(3)      ;*TEST 71      ADJUSTMENT ROUTINES FOR DAC 2 - 3
(3)      ;*****
(2) 012750 000004
(1) 012752 012737 000001 001166
1481      ;NOW DO DAC 2 AND 3      ;;DO 1 ITERATION
1482 012760 104400
1483 012762 017126
1484      ;OFFSET ADJUST TOR DAC 2
1485 012764 004537 014054      JSR      R5,SNDVLT      ;LOAD E.D.C
1486 012770 020235      NS0000
1487 012772 005077 166454      CLR      @DAC2      ;LOAD DAC VALUE
1488 012776 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
(1) 013002 060000      BIT14!BIT13
1489
1490 013004 104400      TYPE
1491 013006 000042      42      ;TELL OPR. TO ADJUST R42
1492 013010 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1493
1494      ;OFFSET ADJUST FOR DAC 3
1495 013014 005077 166434      CLR      @DAC3
1496 013020 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 013024 020000      BIT13
1497 013026 005077 166422      CLR      @DAC3      ;LOAD THE DAC REGISTER
1498 013032 104400      TYPE
1499 013034 017744      R44      ;TELL OPR. TO ADJUST R44
1500 013036 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1501
1502      ;GAIN ADJUST FOR DAC 2
1503 013042 004537 014054      JSR      R5,SNDVLT
1504 013046 020250      P49976
1505 013050 012777 007777 166374      MOV      #7777,@DAC2      ;LOAD DAC 2
1506 013056 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT14!BIT13
(1) 013062 060000      BIT14!BIT13
1507
1508 013064 104400      TYPE
1509 013066 020123      R41      ;TELL OPR. TO ADJUST R41
1510 013070 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1511
1512      ;GAIN ADJUST FOR DAC 3
1513 013074 012777 007777 166352      MOV      #7777,@DAC3      ;LOAD DAC 3
1514 013102 004537 014146      JSR      R5,CROSS      ;LOAD THE RELAYS WITH BIT13
(1) 013106 020000      BIT13
1515
1516 013110 104400      TYPE
1517 013112 020170      R43      ;TELL OPR. TO ADJUST R43
1518 013114 004737 013374      JSR      PC,CSPACE      ;WAIT FOR OPR. TO TYPE "SPACE"
1519

```



```

1521                ;DAC 2 D/A FUNCTION CHECK
1522
1523 013120 004537 014352        JSR    R5,FUNDAC
1524 013124 001452              DAC2
1525 013126 000051              CHS1
1526
1527                ;DAC 3 D/A FUNCTION CHECK
1528
1529 013130 004537 014352        JSR    R5,FUNDAC
1530 013134 001454              DAC3
1531 013136 000052              CHS2
1532
1533 013140 104400              TYPE
1534 013142 017173              CALDON
1535 013144 000137 011500      JMP    AUTCAL
1536
1537 013150 022626              AUTRAP: CMP    (SP)+,(SP)+
1538 013152 012737 000006 000004  MOV    #6,ERRVEC
1539 013160 104400              TYPE
1540 013162 017437              AUTOER
1541 013164 000137 002312      JMP    CTRLC

```

```

1543                                     .SBTTL  MANUAL LOGIC TEST
1544
1545 013170 012706 001100      MANUL:  MOV    #STACK,SP      ;LOAD STACK
1546 013174 004737 001742      JSR    PC,LDTRAP      ;LOAD BUS ADDRESS
1547 013200 104400
1548 013202 017232
1549 013204 004737 013332      1S:    JSR    PC,ACTRLC      ;TEST FOR CTRL C
1550 013210 104406
1551 013212 017700 165722      MOV    #SWR,R0        ;READ THE SWITCHES
1552 013216 010001      MOV    R0,R1          ;COPY
1553 013220 042700 017777      BIC    #17777,R0      ;MASK OFF BITS
1554 013224 001003      BNE    2S              ;BR IF NOT DAC 0
1555 013226 013702 001446      MOV    DAC0,R2        ;LOAD DAC 0 BUS ADDRESS
1556 013232 000424      BR     10S
1557 013234 022700 020000      2S:    CMP    #BIT13,R0      ;TEST FOR DAC 1
1558 013240 001003      BNE    3S              ;BR IF NOT
1559 013242 013702 001450      MOV    DAC1,R2        ;LOAD DAC 1 BUS ADDRESS
1560 013246 000416      BR     10S
1561 013250 022700 040000      3S:    CMP    #BIT14,R0      ;TEST FOR DAC 2
1562 013254 001003      BNE    4S              ;BR IF NOT
1563 013256 013702 001452      MOV    DAC2,R2        ;LOAD DAC 2 BUS ADDRESS
1564 013262 000410      BR     10S
1565 013264 022700 060000      4S:    CMP    #BIT14:BIT13,R0 ;TEST FOR DAC 3
1566 013270 001003      BNE    5S              ;BR IF NOT
1567 013272 013702 001454      MOV    DAC3,R2        ;LOAD DAC 3 BUS ADDRESS
1568 013276 000402      BR     10S
1569 013300 013702 001444      5S:    MOV    STAT,R2        ;LOAD STATUS REG. BUS ADDRESS
1570
1571 013304 010112      10S:   MOV    R1,(R2)        ;LOAD THE SWITCHES INTO THE SELECTED ADDRESS
1572 013306 013703 001442      MOV    DELAY,R3       ;PRESET THE DELAY
1573 013312 005303      11S:   DEC    R3              ;DELAY
1574 013314 001376      BNE    11S
1575 013316 005012      CLR    (R2)           ;CLEAR THE REGISTER
1576 013320 013703 001442      MOV    DELAY,R3       ;LOAD THE PRESET
1577 013324 005303      12S:   DEC    R3              ;DELAY
1578 013326 001376      BNE    12S
1579 013330 000725      BR     1S

```

```

1581
1582           ;TEST FOR CTRL C  -- DONT WAIT
1583
1584 013332 105777 165606   ACTRLC: TSTB   @STKS           ;INPUT FLAG ?
1585 013336 100014           BPL   1$           ;NO
1586 013340 017737 165602 013372   MOV   @STKB,10$   ;READ THE CHARACTER
1587 013346 042737 177600 013372   BIC   @177600,10$
1588 013354 022737 000003 013372   CMP   #3,10$     ;TEST FOR CTRL C
1589 013362 001002           BNE   1$         ;BR IF NOT
1590 013364 000137 002312   JMP   CTRLC
1591 013370 000207           1$:   RTS   PC           ;EXIT
1592 013372 000000           10$:  0
1593
1594           ;WAIT FOR A "SPACE" CHARACTER
1595
1596 013374 105777 165544   CSPACE: TSTB   @STKS           ;WAIT FOR CHAR
1597 013400 100375           BPL   CSPACE
1598 013402 017737 165540 013452   MOV   @STKB,10$   ;READ CHAR
1599 013410 042737 177600 013452   BIC   @177600,10$ ;MASK
1600 013416 022737 000003 013452   CMP   #3,10$     ;TEST FOR CTRL C
1601 013424 001002           BNE   1$
1602 013426 000137 002312   JMP   CTRLC
1603 013432 022737 000040 013452   1$:   CMP   #40,10$ ;TEST FOR SPACE
1604 013440 001001           BNE   2$
1605 013442 000207           RTS   PC           ;EXIT
1606 013444 104400           2$:   TYPE
1607 013446 016774           QMARK
1608 013450 000751           BR   CSPACE
1609
1610 013452 000000           10$:  0

```



# MOS

MAINDEC-11-DZAAC-A  
DZAACA.P11

AA11-K DIAGNOSTIC  
MANUAL LOGIC TEST

MACY11 27(732) 20-AUG-76 09:27 PAGE 42

SEQ 0064

```

1612
1613
1614
1615 013454 012706 001100
1616 013460 004737 001742
1617 013464 104406
1618 013466 004737 013332
1619 013472 017700 165442
1620 013476 010077 165744
1621 013502 010077 165742
1622 013506 010077 165740
1623 013512 010077 165736
1624 013516 000762
1625
1626
1627
1628 013520 012706 001100
1629 013524 004737 001742
1630 013530 104406
1631 013532 004737 013332
1632 013536 017700 165376
1633 013542 004737 013556
1634 013546 005000
1635 013550 004737 013556
1636 013554 000765
1637
1638 013556 010077 165664
1639 013562 010077 165662
1640 013566 010077 165660
1641 013572 010077 165656
1642 013576 012700 000020
1643 013602 005300
1644 013604 100376
1645 013606 000207
1646
1652

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

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

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
    
```

```

1654          .SBTTL  SUBROUTINE TO ERASE STORAGE SCOPE SCREEN
1655 013610 104406          CLRVC: CKSWR
1656 013612 032777 000040 165320 BIT      #BITS, @SWR      ;TEST SR BIT 5
1657 013620 001441          BEQ      3$              ;BR IF NOT A STORAGE SCOPE
1658 013622 012777 002000 165614 MOV      #BIT10, @STAT ;ERASE THE SCREEN
1659 013630 052777 010000 165606 BIS      #BIT12, @STAT
1660 013636 012700 000020          MOV      #20, R0      ;SET UP DELAY
1661 013642 005001          CLR      R1
1662 013644 032777 000040 165266 1$: BIT      #BITS, @SWR      ;TEST IF NOT STORAGE SCOPE
1663 013652 001424          BEQ      3$
1664 013654 105777 165564          TSTB     @STAT      ;TEST FOR READY
1665 013660 100421          BMI      3$              ;BRANCH IF SET
1666 013662 005301          DEC      R1              ;DELAY
1667 013664 001367          BNE      1$
1668 013666 004737 013332          JSR     PC, ACTRLC    ;TEST FOR CTRL C
1669 013672 005300          DEC      R0              ;DELAY
1670 013674 001363          BNE      1$
1671 013676 037727 165236 010000 BIT      @SWR, #SW12   ;TEST INHIBIT PRINTOUT
1672 013704 001002          BNE      2$
1673 013706 104400          TYPE
1674 013710 015412          MES3
1675 013712 104406          CKSWR      ;TEST FOR CTRL C
1676 013714 005777 165220          TST      @SWR      ;TEST @SWR
1677 013720 100001          BPL      3$
1578 013722 000000          HALT
1679 013724 000207          3$:  RTS      PC      ;ERASE RETURN FAILED TO SET READY
1680
1681          .SBTTL  SUBROUTINE TO DRAW A HORIZONTAL LINE
1682 013726 005077 165512          LOADVC: CLR      @STAT      ;CLEAR STATUS
1683 013732 012737 007777 020434 MOV      #7777, TEMP1
1684 013740 013700 001444          MOV      STAT, R0
1685 013744 032777 000010 165166 BIT      #SW03, @SWR   ;TEST SR BIT 3
1686 013752 001405          BEQ      4$              ;BR IF DAC #0 AND 1
1687 013754 013701 001452          MOV      DAC2, R1 ;LOAD X AXIS
1688 013760 013702 001454          MOV      DAC3, R2 ;LOAD Y AXIS
1689 013764 000404          BR      5$
1690 013766 013701 001446          4$: MOV      DAC0, R1
1691 013772 013702 001450          MOV      DAC1, R2
1692 013776 012710 002000          5$: MOV      #BIT10, (0) ;SET STORE MODE
1693 014002 013712 020434          MOV      TEMP1, (2)
1694 014006 012711 007777          1$: MOV      #7777, (1)
1695 014012 000402          BR      3$
1696 014014 162711 000010          2$: SUB      #10, (1)
1697 014020 005210          3$: INC      (0)
1698 014022 105710          TSTB     (0)
1699 014024 100376          BPL      -2
1700 014026 022711 000007          CMP      #7, (1)
1701 014032 001370          BNE      2$
1702 014034 104406          CKSWR
1703 014036 004737 013332          JSR     PC, ACTRLC    ;TEST FOR CTRL G
1704 014042 162712 000010          SUB      #10, (2) ;TEST FOR CTRL C
1705 014046 100357          BPL      1$
1706 014050 000207          RTS      PC
1707

```

```

1709 014052 167772          FILZ: 167772          ;DR11-C OUTPUT CONTROL REGISTER
1710
1711          ;SUBROUTINE TO LOAD THE EDC VOLTAGE
1712          ;DATA FORMAT IS:          STX
1713          :                          P OR N
1714          :                          N VOLTS
1715          :                          0 TENTHS VOLTS
1716          :                          0
1717          :                          0
1718          :                          0
1719          :                          0
1720          :                          0
1721          :                          0
1722          :                          0
1723          :                          0
1723 014054 012500          SNOVLT: MOV      (R5)+,P          ;LOAD POINTER
1724 014056 112001          2$:  MOVB     (R0)+,R1          ;GET A BYTE
1725 014060 001422          BEQ      3$                    ;BR IF TERM.
1726 014062 005101          COM     R1                    ;CONVERT IT
1727 014064 110177 177762  MOVB     R1,2FILZ              ;LOAD FUNNY DATA BYTE
1728 014070 012701 001000  MOV     #1000,R1              ;LOAD DELAY
1729 014074 005301          5$:  DEC     R1                    ;DELAY
1730 014076 001376          BNE     5$
1731 014100 042777 000200 177744  BIC     #BIT7,2FILZ          ;CLEAR BIT 7
1732 014106 012701 001000  MOV     #1000,R1              ;LOAD DELAY COUNT
1733 014112 005301          1$:  DEC     R1                    ;DELAY
1734 014114 001376          BNE     1$
1735 014116 052777 000200 177726  BIS     #BIT7,2FILZ          ;SET BIT 7
1736 014124 000754          BR      2$                    ;DO MORE DATA
1737
1738 014126 012701 000000 3$:  MOV     #0,R1                    ;LOAD DELAY
1739 014132 152777 000177 177712  BISB   #177,2FILZ          ;SET BITS 0-6
1740 014140 005301          4$:  DEC     R1                    ;DELAY
1741 014142 001376          BNE     4$
1742 014144 000205          RTS     R5                    ;EXIT
1743
1744 014146 012537 014206  CROSS:  MOV     (R5)+,10$          ;READ REG.
1745 014152 052737 000177 014206  BIS     #177,10$              ;LOAD LOW 7 BITS
1746 014160 013777 014206 177664  MOV     10$,2FILZ            ;LOAD RELAYS
1747 014166 012700 000001  MOV     #1,R0                ;LOAD DELAY
1748 014172 05001          CLR     R1
1749 014174 005301          1$:  DEC     R1                    ;DELAY
1750 014176 001376          BNE     1$
1751 014200 005300          DEC     R0                    ;DELAY
1752 014202 001374          BNE     1$                    ;DONE ?
1753 014204 000205          RTS     R5                    ;EXIT
1754 014206 000000          10$: 0

```



```

1756 .SBTTL TIMER ROUTINE FOR VISUAL TEST PATTERNS
1757 ; ENTER VIA JSR PC,TIMER
1758
1759 014210 104406 TIMER: CKSWR
1760 014212 004737 013332 JSR PC,ACTRLC ;TEST FOR CTRL C
1761 014216 017737 164716 020426 MOV JSWR,TIMSV
1762
1763 014224 032737 000400 020426 TIMERA: BIT #BIT8,TIMSV
1764 014232 001006 BNE TIMER2 ;BIT 8 SET ?
1765 014234 005337 020430 DEC TICKS ;NO, DECREMENT TICKS
1766 014240 001002 BNE TIMER1
1767 014242 062716 000002 ADD #2,(6) ;ADD 2 TO STACK POINTER
1768 014246 000207 TIMER1: RTS PC ;RETURN
1769
1770 ; SWR 8=1 SELECT TEST TO LOCK ON
1771 ; SWR 2-0= TEST NUMBER
1772
1773 014250 042737 177770 020426 TIMER2: BIC #177770,TIMSV
1774 014256 006337 020426 ASL TIMSV
1775 014262 062737 014304 020426 ADD #ROUTPT,TIMSV
1776 014270 017737 004132 020426 MOV @TIMSV,TIMSV
1777 014276 022600 CMP (SP)+,R0
1778 014300 000177 004122 TIMER4: JMP @TIMSV
1779
1780 014304 006650 ROUTPT: PICO ;DISPLAY A HORIZONTAL LINE
1781 014306 006724 PIC1 ;DISPLAY A VERTICAL LINE
1782 014310 007110 PIC3 ;DISPLAY A SQUARE
1783 014312 007340 PIC4 ;DISPALY A "X"
1784 014314 007546 PIC5 ;DISPLAY SETTLING TIME
1785 014316 010004 PIC6 ;DISPLAY CHARACTER SET
1786 014320 011022 PIC7 ;DISPLAY CHANNEL TEST
1787 014322 011326 PIC12 ;DISPLAY ERASE AND PHOSPHOR TEST
1788
1789 014324 013737 014350 020436 CHTIME: MOV CPTYPE,BRLEV1
1790 014332 005337 020436 1S: DEC BRLEV1
1791 014336 001403 BEQ 2S
1792 014340 006337 020430 ASL TICKS
1793 014344 000772 BR 1S
1794 014346 000207 2S: RTS PC
1795
1796 014350 000001 CPTYPE: 1

```

```

1798
1799 ;DO 32 CONVERSIONS ON EACH PRESET VALUE OF THE DAC'S
1800 ;AND SUBTRACT THE FIRST RESULT FROM EACH OF THE RESULTS
1801
1802 014352 013537 014442 FUNDAC: MOV @ (R5)+, XDACUT ;GET BUS ADDRESS OF ARJUT DAC
1803 014356 012537 014402 MOV (R5)+, 60$ ;GET CHANNEL INPUT FOR ARKG
1804 014362 012777 007600 000052 MOV #7600, @XDACUT ;PRESET THE DAC
1805 014370 012737 007600 001124 MOV #7600, $GDDAT ;LOAD EXPECTED
1806
1807 014376 004537 014444 1$: JSR R5, CONVRT ;CONVERT USING ARKG
1808 014402 000051 60$: CHS1 ;
1809
1810 014404 012737 000010 014610 MOV #10, SPREAD ;LOAD LIMIT
1811 014412 004737 014612 JSR PC, COMPAR ;TEST IF WITHIN LIMIT
1812 014416 000401 BR 2$ ;;BR IF WITHIN
1813 014420 104016 ERROR 16 ;SELECTED DAC HAS A LINEARITY ERROR
1814
1815 014422 162777 000400 000012 2$: SUB #400, @XDACUT ;UPDATE DAC
1816 014430 162737 000400 001124 SUB #400, $GDDAT ;UPDATE GOOD VALUE
1817 014436 001357 BNE 1$ ;;BR IF NOT FINISHED
1818
1819 014440 000000 10$: 0
1820 014442 000000 XDACUT: 0

```

```

1822                                     ;SUBROUTINE TO GET AVERAGE OF 32 CONVERSIONS FROM THE KNOWN GOOD AR 11
1823
1824 014444 012537 014564 CONVRT: MOV (R5)+,10$
1825 014450 013737 014564 014606 MOV 10$,CHANL
1826 014456 000337 014564 SWAB 10$
1827 014462 012737 000020 014566 MOV #16.,11$
1828 014470 005037 014602 CLR ADEND
1829 014474 013777 014564 000072 MOV 10$,GADCS
1830 014502 005277 000066 2$: INC GADCS
1831 014506 105777 000062 1$: TSTB GADCS
1832 014512 100375 BPL 1$
1833 014514 067737 000056 014602 ADD GADDBR,ADEND
1834 014522 005337 014566 DEC 11$
1835 014526 001365 BNE 2$
1836 014530 006237 014602 ASR ADEND
1837 014534 006237 014602 ASR ADEND
1838 014540 006237 014602 ASR ADEND
1839 014544 006237 014602 ASR ADEND
1840 014550 005537 014602 ADC ADEND ;ROUND UP
1841 014554 013737 014602 001126 MOV ADEND,$BDDAT
1842 014562 000205 RTS RS
1843
1844 014564 000000 10$: 0
1845 014566 000000 11$: 0
1846 014570 001754 V1754: 1754
1847 014572 000024 V24: 24
1848 014574 17J460 GADCS: 170460
1849 014576 170462 GADDBR: 170462
1850 014600 170470 GSTAT: 170470
1851 014602 000000 ADEND: 0
1852 014604 000000 FCHANL: 0
1853 014606 000000 CHANL: 0
1854 014610 000000 SPREAD: 0
1855
1856 014612 010046 COMPAR: MOV RO,-(SP)
1857 014614 010146 MOV R1,-(SP)
1858 014616 013700 001124 MOV $GDDAT,RO ;LOAD RO
1859 014622 013701 001126 MOV $BDDAT,R1 ;LOAD R1
1860 014626 160100 SUB R1,RO ;SUBTRACT
1861 014630 100001 BPL 8$ ;
1862 014632 005400 NEG RO
1863 014634 020037 014610 8$: CMP RO,SPREAD ;MAGNITUDE OF DIFF. IN RO
1864 014640 003405 BLE 10$
1865 014642 012601 9$: MOV (SP)+,R1
1866 014644 012600 MOV (SP)+,RO
1867 014646 062716 000002 ADD #2,(SP)
1868 014652 000207 RTS PC
1869
1870 014654 012601 10$: MOV (SP)+,R1
1871 014656 012600 MOV (SP)+,RO
1872 014660 000207 RTS PC
1873

```



1875  
1876  
1877 014662 005015 040412 030501  
014670 026461 020113 044504  
014676 043501 047516 052123  
014704 041511 052040 051505  
014712 026124 024040 040515  
014720 047111 042504 026503  
014726 030461 042055 040532  
014734 041501 040455 024460  
014742 005015  
1878 014744 042523 042514 052103  
014752 052040 051505 051524  
014760 041040 020131 054524  
014766 044520 043516 040440  
014774 052040 047527 046040  
015002 052105 042524 020122  
015010 027111 027104 005015  
1879 015016 046101 035011 052501  
015024 047524 046040 043517  
015032 041511 052040 051505  
015040 006524 012  
1880 015043 101 004504 040472  
015050 052125 020117 044504  
015056 050123 040514 020131  
015064 042524 052123 024040  
015072 044504 050123 040514  
015100 020131 041523 050117  
015106 020105 047503 047116  
015114 041505 042524 024504  
015122 005015  
1881 015124 041501 035011 052501  
015132 047524 041440 046101  
015140 041111 040522 044524  
015146 047117 024040 040501  
015154 030461 045455 047440  
015162 052120 047511 020116  
015170 042524 052123 040440  
015176 042522 020101 047117  
015204 054514 006451 012  
1882 015211 115 004514 046472  
015216 047101 040525 020114  
015224 047514 044507 020103  
015232 042524 052123 024040  
015240 053523 020122 032461  
015246 030455 020063 020075  
015254 042522 044507 052123  
015262 051105 020040 031061  
015270 030055 036440 042040  
015276 052101 024501 005015  
1883 015304 042115 035011 040515  
015312 052516 046101 042040  
015320 051511 046120 054501  
015326 024040 047516 042040  
015334 051511 046120 054501  
015342 051440 047503 042520

.SBTTL ASCII MESSAGES

TITLE: .ASCII <15><12><12>'AA11-K DIAGNOSTIC TEST, (MAINDEC-11-DZAAC-A0)'<15><12>

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

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

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

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

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

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

1884	015350	006451	012			
	015353	115	004503	046472	.ASCII /MC	:MANUAL CALIBRATION LOOP/<15><12>
	015360	047101	040525	020114		
	015366	040503	044514	051102		
	015374	052101	047511	020116		
	015402	047514	050117	005015		
1885	015410	000056			.ASCIZ /./	
1886	015412	005015	051105	051501	MES3: .ASCIZ	<15><12>"ERASE RETURN FAILED TO SET READY"<15><12>
	015420	020105	042522	052524		
	015426	047122	043040	044501		
	015434	042514	020104	047524		
	015442	051440	052105	051040		
	015450	040505	054504	005015		
	015456	000				
1887	015457	015	051412	047503	MES6: .ASCIZ	<15><12>'SCOPE ADJUSTMENT TEST'
	015464	042520	040440	045104		
	015472	051525	046524	047105		
	015500	020124	042524	052123		
	015506	000				
1888	015507	015	051412	051127	MES15: .ASCII	<15><12>/SWRB AND 0 THRU 2 CONTROL PATTERN/<15><12>
	015514	020070	047101	020104		
	015522	020060	044124	052522		
	015530	031040	041440	047117		
	015536	051124	046117	050040		
	015544	052101	042524	047122		
	015552	005015				
1889	015554	053523	020122	044502	.ASCII	'SWR BIT 3 SELECTS DAC 0+1 OR DAC 2+3'<15><12>
	015562	020124	020063	042523		
	015570	042514	052103	020123		
	015576	040504	020103	025460		
	015604	020061	051117	042040		
	015612	041501	031040	031453		
	015620	005015				
1890	015622	053523	020122	044502	.ASCII	/SWR BIT 5 SELECTS STORAGE SCOPE MODE/<15><12>
	015630	020124	020065	042523		
	015636	042514	052103	020123		
	015644	052123	051117	043501		
	015652	020105	041523	050117		
	015660	020105	047515	042504		
	015666	005015				
1891	015670	053523	020122	044502	.ASCII	/SWR BIT 7 ENABLES VERTICAL SETTling TEST/<15><12>
	015676	020124	020067	047105		
	015704	041101	042514	020123		
	015712	042526	052122	041511		
	015720	046101	051440	052105		
	015726	046124	047111	020107		
	015734	042524	052123	005015		
1892	015742	053523	020122	044502	.ASCIZ	/SWR BIT 9 SELECTS TWO'S COMPLEMENT DISPLAY MODE/<15><12>
	015750	020124	020071	042523		
	015756	042514	052103	020123		
	015764	053524	023517	020123		
	015772	047503	050115	042514		
	016000	042515	052116	042040		
	016006	051511	046120	054501		
	016014	046440	042117	006505		
	016022	000012				

1893	016024	052123	052101	051525	EM1:	.ASCIZ /STATUS REGISTER IN ERROR/
	016032	051040	043505	051511		
	016040	042524	020122	047111		
	01604E	042440	051122	051117		
	016054	000				
1894	016055	104	041501	020060	EM2:	.ASCIZ /DAC0 REGISTER IN ERROR/
	016062	042522	044507	052123		
	016070	051105	044440	020116		
	016076	051105	047522	000122		
1895	016104	040504	030503	051040	EM3:	.ASCIZ /DAC1 REGISTER IN ERROR/
	016112	043505	051511	042524		
	016120	020122	047111	042440		
	016126	051122	051117	000		
1896	016133	104	041501	020062	EM4:	.ASCIZ /DAC2 REGISTER IN ERROR/
	016140	042522	044507	052123		
	016146	051105	044440	020116		
	016154	051105	047522	000122		
1897	016162	040504	031503	051040	EM5:	.ASCIZ /DAC3 REGISTER IN ERROR/
	016170	043505	051511	042524		
	016176	020122	047111	042440		
	016204	051122	051117	000		
1898	016211	101	030501	026461	EM6:	.ASCIZ /AA11-K FAILED TO INTERRUPT/
	016216	020113	040506	046111		
	016224	042105	052040	020117		
	016232	047111	042524	051122		
	016240	050125	000124			
1899	016244	040501	030461	045455	EM7:	.ASCIZ /AA11-K INTERRUPTED IN ERROR/
	016252	044440	052116	051105		
	016260	052522	052120	042105		
	016266	044440	020116	051105		
	016274	047522	000122			
1900	016300	052502	020123	044524	EM10:	.ASCIZ /BUS TIME-OUT WHEN REFERENCING THE AA11-K/
	016306	042515	047455	052125		
	016314	053440	042510	020116		
	016322	042522	042506	042522		
	016330	041516	047111	020107		
	016336	044124	020105	040501		
	016344	030461	045455	000		
1901	016351	101	030501	026461	EM11:	.ASCIZ /AA11-K READY BIT ERROR/
	016356	020113	042522	042101		
	016364	020131	044502	020124		
	016372	051105	047522	000122		
1902	016400	047520	042527	020122	EM12:	.ASCIZ /POWER SUPPLY VOLTAGE WAS INCORRECT/
	016406	052523	050120	054514		
	016414	053040	046117	040524		
	016422	042507	053440	051501		
	016430	044440	041516	051117		
	016436	042522	052103	000		
1903	016443	117	042516	041040	EM13:	.ASCIZ /ONE BIT THE DUST/
	016450	052111	052040	042510		
	016456	042040	051525	000124		
1904	016464	040501	030461	045455	EM14:	.ASCIZ /AA11-K LOGIC SIGNAL HIGH OUTPUT TO LOW/
	016472	046040	043517	041511		
	016500	051440	043511	040516		
	016506	020114	044510	044107		
	016514	047440	052125	052520		



1905	016522	020124	047524	046040			
	016530	053517	000				
	016533	101	030501	026461	EM15:	.ASCIZ	/AA11-K LOGIC SIGNAL LOW OUTPUT TO HIGH/
	016540	020113	047514	044507			
	016546	020103	044523	047107			
	016554	046101	046040	053517			
	016562	047440	052125	052520			
	016570	020124	047524	044040			
1906	016576	043511	000110				
	016602	042523	042514	052103	EM16:	.ASCIZ	/SELECTED DAC HAS A LINEARITY ERROR/
	016610	042105	042040	041501			
	016616	044040	051501	040440			
	016624	046040	047111	040505			
	016632	044522	054524	042440			
	016640	051122	051117	000			
1907	016645	105	051122	041520	DH1:	.ASCIZ	/ERRPC IBASE EXPECT BAD/
	016652	044411	040502	042523			
	016660	020040	042440	050130			
	016666	041505	020124	020040			
	016674	041040	042101	000			
1908	016701	105	051122	041520	DH6:	.ASCIZ	/ERRPC IBASE/
	016706	044411	040502	042523			
	016714	000					
1909	016715	105	051122	041520	DH13:	.ASCIZ	/ERRPC IBASE # FIRST # NOW/
	016722	044411	040502	042523			
	016730	021411	043040	051111			
	016736	052123	021411	047040			
	016744	053517	000				
1910	016747	105	051122	041520	DH14:	.ASCIZ	/ERRPC IBASE GOOD BAD/
	016754	044411	040502	042523			
	016762	043411	047517	004504			
	016770	040502	000104				
1911							
1912	016774	015	012	077	QMARK:	.BYTE	15,12,77,15,12,0
	016777	015	012	000			
1913	017002	136	103	015	CONTC:	.BYTE	136,103,15,12,56,0
	017005	012	056	000			
1914	017010	015	012		FOUND1:	.BYTE	15,12
1915	017012	051120	043517	040522		.ASCIZ	/PROGRAM DETECTED /
	017020	020115	042504	042524			
	017026	052103	042105	000040			
1916	017034	034050	020051	020040	FOUND2:	.ASCIZ	/ 8) AA11-K(S) /
	017042	040501	030461	045455			
	017050	051450	020051	000040			
1917	017056	015	012	000		.BYTE	15,12,0
1918	017061	015	012		SEL001:	.BYTE	15,12
1919	017063	123	052105	051440		.ASCIZ	/SET SWITCH TO SELECT DAC 0 AND 1/<15><12>
	017070	044527	041524	020110			
	017076	047524	051440	046105			
	017104	041505	020124	040504			
	017112	020103	020060	047101			
	017120	020104	006461	000012			
1920	017126	015	012		SEL023:	.BYTE	15,12
1921	017130	042523	020124	053523		.ASCIZ	/SET SWITCH TO SELECT DAC 2 AND 3/<15><12>
	017136	052111	044103	052040			
	017144	020117	042523	042514			

	017152	052103	042040	041501
	017160	031040	040440	042116
	017166	031440	005015	000
1922	017173	015	012	
1923	017175	101	052125	020117
	017202	040503	044514	051102
	017210	052101	047511	020116
	017216	047503	050115	042514
	017224	042524	006504	000012
1924	017232	015	012	
1925	017234	053523	030440	026465
	017242	031461	051440	046105
	017250	041505	020124	044124
	017256	020105	042522	044507
	017264	052123	051105	005015
1926	017272	053523	030440	026462
	017300	030060	040440	042522
	017306	046040	040517	042504
	017314	020104	047111	047524
	017322	052040	042510	051440
	017330	046105	041505	042524
	017336	020104	042522	044507
	017344	052123	051105	005015
1927	017352	000		
1928	017353	015	012	
1929	017355	101	052125	020117
	017362	040503	044514	051102
	017370	052101	047511	020116
	017376	040450	030501	026461
	017404	020113	050117	044524
	017412	047117	052040	051505
	017420	020124	051101	040505
	017426	047440	046116	024531
1930	017434	015	012	000
1931	017437	015	041012	051525
	017444	052040	040522	020120
	017452	044127	047105	046040
	017460	040517	044504	043516
	017466	053040	046117	040524
	017474	042507	047440	020122
	017502	044103	047101	044507
	017510	043516	040440	051040
	017516	046105	054501	005015
1932	017524	037477	040411	030501
	017532	026461	020113	050117
	017540	044524	047117	052040
	017546	051505	020124	051101
	017554	040505	037440	004477
	017562	005015	000	
1933				
1937	017565	015	040412	045104
(1)	017572	051525	020124	031522
(1)	017600	020070	047506	020122
(1)	017606	020101	052516	046114
(1)	017614	047440	020116	044124
(1)	017622	020105	042515	042524

CALDON: .BYTE 15,12  
.ASCII /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>

.ASCIZ /?? AA11-K OPTION TEST AREA ?? /<15><12>

R38: .ASCIZ <15><12>/ADJUST R38 FOR A NULL ON THE METER/

(1)	017630	000122				
1938	017632	005015	042101	052512	R40:	.ASCIZ <15><12>/ADJUST R40 FOR A NULL ON THE METER/
(1)	017640	052123	051040	030064		
(1)	017646	043040	051117	040440		
(1)	017654	047040	046125	020114		
(1)	017662	047117	052040	042510		
(1)	017670	046440	052105	051105		
(1)	017676	000				
1939	017677	015	040412	045104	R22:	.ASCIZ <15><12>/ADJUST R22 FOR A NULL ON THE METER/
(1)	017704	051525	020124	031122		
(1)	017712	023062	047506	020122		
(1)	017720	020101	052516	046114		
(1)	017726	047440	020116	044124		
(1)	017734	020105	042515	042524		
(1)	017742	000122				
1940	017744	005015	042101	052512	R44:	.ASCIZ <15><12>/ADJUST R44 FOR A NULL ON THE METER/
(1)	017752	052123	051040	032064		
(1)	017760	043040	051117	040440		
(1)	017766	047040	046125	020114		
(1)	017774	047117	052040	042510		
(1)	020002	046440	052105	051105		
(1)	020010	000				
1941	020011	015	040412	045104	R37:	.ASCIZ <15><12>/ADJUST R37 FOR A NULL ON THE METER/
(1)	020016	051525	020124	031522		
(1)	020024	020067	047506	020122		
(1)	020032	020101	052516	046114		
(1)	020040	047440	020116	044124		
(1)	020046	020105	042515	042524		
(1)	020054	000122				
1942	020056	005015	042101	052512	R39:	.ASCIZ <15><12>/ADJUST R39 FOR A NULL ON THE METER/
(1)	020064	052123	051040	034463		
(1)	020072	043040	051117	040440		
(1)	020100	047040	046125	020114		
(1)	020106	047117	052040	042510		
(1)	020114	046440	052105	051105		
(1)	020122	000				
1943	020123	015	040412	045104	R41:	.ASCIZ <15><12>/ADJUST R41 FOR A NULL ON THE METER/
(1)	020130	051525	020124	032122		
(1)	020136	020061	047506	020122		
(1)	020144	020101	052516	046114		
(1)	020152	047440	020116	044124		
(1)	020160	020105	042515	042524		
(1)	020166	000122				
1944	020170	005015	042101	052512	R43:	.ASCIZ <15><12>/ADJUST R43 FOR A NULL ON THE METER/
(1)	020176	052123	051040	031464		
(1)	020204	043040	051117	040440		
(1)	020212	047040	046125	020114		
(1)	020220	047117	052040	042510		
(1)	020226	046440	052105	051105		
(1)	020234	000				
1945		000001				STX=1
1946		000003				ETX=3
1947	020235	001			N50000:	.BYTE STX
1948	020236	032516	030060	030060		.ASCII /N500000V/
	020244	053060				
1949	020246	003	000			.BYTE ETX,0



1950	020250	001			P49976:	.BYTE	STX
1951	020251	120	034464	033471		.ASCII	/P49976OV/
	020256	030066	126				
1952	020261	003	000			.BYTE	ETX,0
1953		020264				.EVEN	
1954	020264	001116	001444	001124	DT1:	SERRPC,STAT,SGDDAT,SBDDAT,0	
	020272	001126	000000				
1955	020276	001116	001446	001124	DT2:	SERRPC,DACO,SGDDAT,SBDDAT,0	
	020304	001126	000000				
1956	020310	001116	001450	001124	DT3:	SERRPC,DAC1,SGDDAT,SBDDAT,0	
	020316	001126	000000				
1957	020322	001116	001452	001124	DT4:	SERRPC,DAC2,SGDDAT,SBDDAT,0	
	020330	001126	000000				
1958	020334	001116	001454	001124	DT5:	SERRPC,DAC3,SGDDAT,SBDDAT,0	
	020342	001126	000000				
1959	020346	001116	001444	000000	DT6:	SERRPC,STAT,0	
1960	020354	001116	001444	020442	DT13:	SERRPC,STAT,EVER,SUNIT,0	
	020362	001206	000000				
1961	020366	001116	001444	001124	DT14:	SERRPC,STAT,SGDDAT,SBDDAT,0	
	020374	001126	000000				
1962	020400	000000	000000	000000	DF0:	0,0,0,0,0,0,0,0	
	020406	000000	000000	000000			
	020414	000000	000000				
1963	020420	000000			LOW:	0	
1964	020422	000000			HIGH:	0	
1965	020424	000010			INCR:	10	
1966	020426	000000			TIMSV:	0	
1967	020430	000000			TICKS:	0	
1968	020432	000000			TEMP:	0	
1969	020434	000000			TEMP1:	0	
1970	020436	000000			BRLEV1:	0	
1971	020440	000000			BRLEV2:	0	
1972	020442	000000			EVER:	0	
1973	020444	000000			EVER1:	0	
1974	020446	000000			OPRIN:	0	
1975	020450	000000			P7CNT:	0	
1976	020452	000000			P7PNT:	0	
1977	020454	010530			CH01:	0	
1978	020456	010561				0	
1979	020460	010516				0	
1980	020462	010617				0	
1981	020464	010617				0	
1982	020466	010542				0	
1983	020470	010605				0	
1984	020472	011002				0	
1985	020474	010725				0	
1986	020476	010530			CH02:	0	
1987	020500	010561				0	
1988	020502	010516				0	
1989	020504	010617				0	
1990	020506	010617				0	
1991	020510	010542				0	
1992	020512	010605				0	
1993	020514	011002				0	
1994	020516	010732				0	
1995						0	

;TEMPORARY STORAGE

CHANNEL  
SPACEA  
N1  
CHANNEL  
SPACEA  
N2

1996

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(2) 020520  
(3) 020520 010046  
(3) 020522 010146  
(3) 020524 010246  
(3) 020526 010346  
(3) 020530 010546  
(1) 020532 012746 020200  
(1) 020536 016605 000020  
(1) 020542 100004  
(1) 020544 005405  
(1) 020546 112766 000055 000001  
(1) 020554 005000  
(1) 020556 012703 020734  
(1) 020562 112723 000040  
(1) 020566 005002  
(1) 020570 016001 020724  
(1) 020574 160105  
(1) 020576 002402  
(1) 020600 005202  
(1) 020602 000774  
(1) 020604 060105  
(1) 020606 005702  
(1) 020610 001002  
(1) 020612 105716  
(1) 020614 100407  
(1) 020616 106316  
(1) 020620 103003  
(1) 020622 116663 000001 177777  
(1) 020630 052702 000060  
(1) 020634 052702 000040  
(1) 020640 110223  
(1) 020642 005720  
(1) 020644 020027 000010  
(1) 020650 002746  
(1) 020652 003002  
(1) 020654 010502  
(1) 020656 000764  
(1) 020660 105726  
(1) 020662 100003  
(1) 020664 116663 177777 177776  
(1) 020672 105013  
(3) 020674 012605  
(3) 020676 012603  
(3) 020700 012602

```

*****
*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      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB    #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3     ;;SETUP THE OUTPUT POINTER
MOVB    #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
CLR      R2            ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
SUB      R1,R5         ;;FORM THIS BCD DIGIT
BLT     4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR      3$

4$:
ADD      R1,R5         ;;ADD BACK THE CONSTANT
TST     R2            ;;CHECK IF BCD DIGIT=0
BNE     5$            ;;FALL THROUGH IF 0
TSTB   (SP)          ;;STILL DOING LEADING 0'S?
BMI     7$            ;;BR IF YES
ASLB   (SP)          ;;MSD?
BCC     6$            ;;BR IF NO
MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST    (R0)+        ;;JUST INCREMENTING
CMP    R0,#10       ;;CHECK THE TABLE INDEX
BLT    2$            ;;GO DO THE NEXT DIGIT
BGT    8$            ;;GO TO EXIT
MOV    R5,R2        ;;GET THE LSD
BR     6$            ;;GO CHANGE TO ASCII
8$:
TSTB   (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
BPL    9$            ;;BR IF NO
MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
CLRB   (R3)         ;;SET THE TERMINATOR
MOV    (SP)+,R5     ;;POP STACK INTO R5
MOV    (SP)+,R3     ;;POP STACK INTO R3
MOV    (SP)+,R2     ;;POP STACK INTO R2
    
```



```

(3) 020702 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
(3) 020704 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
(1) 020706 104400      TYPE     $DBLK        ;; NOW TYPE THE NUMBER
(1) 020712 016666      MOV      2(SP),4(SP)  ;; ADJUST THE STACK
(1) 020720 012616      MOV      (SP)+,(SP)
(1) 020722 000002      RTI                    ;; RETURN TO USER
(1) 020724 023420      SDTBL:  10000.
(1) 020726 001750      1000.
(1) 020730 000144      100.
(1) 020732 000012      10.
(1) 020734 000004      SDBLK:  .BLKW 4
1997 .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)
(1) *****
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW14=1      LOOP ON TEST
(1) *SW11=1      INHIBIT ITERATIONS
(1) *SW08=1      LOOP ON TEST IN SWR<7:0>
(1) *CALL
(1) *          SCOPE          ;;SCOPE=IOT
(1)
(1) 020744          SSCOPE:
(1) 020744 104406      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(2) 020746 004737      JSR          PC,ACTRLC
(1) 020752 032777      BIT          $BIT14,$SWR          ;; LOOP ON PRESENT TEST?
(1) 020760 001070      BNE          $OVER          ;; YES IF SW14=1
(1) *****START OF CODE FOR THE XOR TESTER*****
(1) 020762 000416      $XTSTR: BR          6$          ;; IF RUNNING ON THE "XOR" TESTER CHANGE
(1) *****THIS INSTRUCTION TO A "NOP" (NOP=240)
(1) *****SAVE THE CONTENTS OF THE ERROR VECTOR
(1) *****SET FOR TIMEOUT
(1) *****TIME OUT ON XOR?
(1) 020764 013746      MOV          @ERRVEC,-(SP)      ;; RESTORE THE ERROR VECTOR
(1) 020770 012737      MOV          @SS,@ERRVEC      ;; GO TO THE NEXT TEST
(1) 020776 005737      TST         @177060          ;; CLEAR THE STACK AFTER A TIME OUT
(1) 021002 012637      MOV          (SP)+,@ERRVEC      ;; RESTORE THE ERROR VECTOR
(1) 021006 000446      BR          $SVLAD          ;; LOOP ON THE PRESENT TEST
(1) 021010 022626      5$: CMP      (SP)+,(SP)+      ;; CLEAR THE STACK AFTER A TIME OUT
(1) 021012 012637      MOV          (SP)+,@ERRVEC      ;; RESTORE THE ERROR VECTOR
(1) 021016 000451      BR          $OVER          ;; LOOP ON THE PRESENT TEST
(1) 021020          6$: *****END OF CODE FOR THE XOR TESTER*****
(1) 021020 032777      BIT          $BIT08,$SWR      ;; LOOP ON SPEC. TEST?
(1) 021026 001404      BEQ          2$          ;; BR IF NO
(1) 021030 127737      CMPB        $SWR,$TSTNM      ;; ON THE RIGHT TEST? SWR<7:0>
(1) 021036 001441      BEQ          $OVER          ;; BR IF YES
(1) 021040 105737      2$: TSTB     $ERFLG          ;; HAS AN ERROR OCCURRED?
(1) 021044 001404      BEQ          3$          ;; BR IF NO
(1) 021046 105037      4$: CLRB     $ERFLG          ;; ZERO THE ERROR FLAG
(1) 021052 005037      CLR          $TIMES          ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 021056 032777      3$: BIT          $BIT11,$SWR      ;; INHIBIT ITERATIONS?
(1) 021064 001011      BNE          1$          ;; BR IF YES
(1) 021066 005737      TST         $PASS          ;; IF FIRST PASS OF PROGRAM
(1) 021072 001406      BEQ          1$          ;; INHIBIT ITERATIONS
(1) 021074 005237      INC          $ICNT          ;; INCREMENT ITERATION COUNT
(1) 021100 023737      CMP         $TIMES,$ICNT      ;; CHECK THE NUMBER OF ITERATIONS MADE
(1) 021106 002015      BGE          $OVER          ;; BR IF MORE ITERATION REQUIRED

```



```

( ) 021110 012737 000001 001104 1S:  MOV  #1,SICNT      ;;REINITIALIZE THE ITERATION COUNTER
( ) 021116 013737 021156 001166      MOV  SMXCNT,STIMES  ;;SET NUMBER OF ITERATIONS TO DO
( ) 021124 105237 001102                SSVLAD: INCB  STSTNM      ;;COUNT TEST NUMBERS
( ) 021130 113737 001102 001200      MOV  STSTNM,STESTN  ;;SET TEST NUMBER IN APT MAILBOX
(1) 021136 011637 001106                MOV  (SP),SLPADR   ;;SAVE SCOPE LOOP ADDRESS
(1) 021142 011777 001102 157772  SOVER: MOV  STSTNM,SDISPLAY ;;DISPLAY TEST NUMBER
(1) 021150 013716 001106                MOV  SLPADR,(SP)  ;;FUDGE RETURN ADDRESS
(1) 021154 000002                RTI                ;;FIXES PS
(1) 021156 003720                SMXCNT: 2000      ;;MAX. NUMBER OF ITERATIONS
1998 .SBTTL  ERROR HANDLER ROUTINE

```

```

(1) (2) *****
(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 SERRTYP 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      ;;EP OR=EMT AND N=ERROR ITEM NUMBER

```

```

(1) 021160                SERROR:
(1) 021160 104406                CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(2) 021162 004737 013332                JSR  PC,ACTRLC
(1) 021166 105237 001103                7S:  INCB  SEARFLG  ;;SET THE ERROR FLAG
(1) 021172 001775                BEQ  7S          ;;DON'T LET THE FLAG GO TO ZERO
(1) 021174 013777 001102 157740      MOV  STSTNM,SDISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 021202 005237 001112                INC  SERTTL      ;;INC THE ERROR COUNT
(1) 021206 011637 001116                MOV  (SP),SERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 021212 162737 000002 001116      SUB  #2,SERRPC
(1) 021220 117737 157672 001114      MOV  SDERRPC,SITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 021226 032777 020000 157704      BIT  #BIT13,SWR   ;;SKIP TYPEOUT IF SET
(1) 021234 001004                BNE  20S        ;;SKIP TYPEOUTS
(1) 021236 004737 021310                JSR  PC,SERRTYP  ;;GO TO USER ERROR ROUTINE
(1) 021242 104400 001171                TYPE  ,SCLF
(1) 021246                20S:
(1) 021246 122737 000001 001214      CMPB #APTENV,SENV ;;RUNNING IN APT MODE
(1) 021254 001007                BNE  2S          ;;NO SKIP APT ERROR REPORT
(1) 021256 113737 001114 021270      MOV  SITEMB,21S  ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 021264 004737 023270                JSR  PC,SATY4    ;;REPORT FATAL ERROR TO APT
(1) 021270 000                21S:  .BYTE  0
(1) 021271 000                .BYTE  0
(1) 021272 000777                22S:  BR  22S          ;;APT ERROR LOOP
(1) 021274 005777 157640                2S:  TST  SWR        ;;HALT ON ERROR
(1) 021300 100002                BPL  3S          ;;SKIP IF CONTINUE
(1) 021302 000000                HALT
(1) 021304 104406                CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 021306 000002                3S:
(1) 021306 000002                RTI          ;;RETURN
1999 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

```

```

(1) (2) *****
(1) *THIS ROUTINE USES THE "ITEM CONTROL BYTE" (SITEMB) TO DETERMINE WHICH
(1) *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (SERRTB),
(1) *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)

```

```

(1) 021310                                SERRTYP:
(1) 021310 104400 001171                TYPE      $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 021314 010046                        MOV        RO,-(SP)      ;; SAVE RO
(1) 021316 005000                        CLR        RO           ;; PICKUP THE ITEM INDEX
(1) 021320 153700 001114                BISB      @SITEMB,RO
(1) 021324 001004                        BNE       1$           ;; IF ITEM NUMBER IS ZERO, JUST
(2) 021326 013746 001116                MOV        SERRPC,-(SP) ;; TYPE THE PC OF THE ERROR
(2) 021332 104401                        TYPOC
(1) 021334 000426                        BR         6$          ;; SAVE SERRPC FOR TYPEOUT
(1) 021336 005300                        1$: DEC      RO        ;; ERROR ADDRESS
(1) 021340 006300                        ASL      RO          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 021342 006300                        ASL      RO          ;; GET OUT
(1) 021344 006300                        ASL      RO          ;; ADJUST THE INDEX SO THAT IT WILL
(1) 021346 067700 001256                ADD      @SERRTB,RO    ;; WORK FOR THE ERROR TABLE
(1) 021352 012037 021362                MOV      (RO)+,2$
(1) 021356 001404                        BEQ      3$
(1) 021360 104400                        TYPE
(1) 021362 000000                        2$: .WORD 0          ;; FORM TABLE POINTER
(1) 021364 104400 001171                TYPE      $CRLF          ;; PICKUP "ERROR MESSAGE" POINTER
(1) 021370 012037 021400                3$: MOV      (RO)+,4$    ;; SKIP TYPEOUT IF NO POINTER
(1) 021374 001404                        BEQ      5$          ;; TYPE THE "ERROR MESSAGE"
(1) 021376 104400                        TYPE      $CRLF          ;; "ERROR MESSAGE" POINTER GOES HERE
(1) 021400 000000                        4$: .WORD 0          ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 021402 104400 001171                TYPE      $CRLF          ;; PICKUP "DATA HEADER" POINTER
(1) 021406 011000                        5$: MOV      (RO),RO    ;; SKIP TYPEOUT IF 0
(1) 021410 001004                        BNE      7$          ;; TYPE THE "DATA HEADER"
(1) 021412 012600                        6$: MOV      (SP)+,RO    ;; "DATA HEADER" POINTER GOES HERE
(1) 021414 104400 001171                TYPE      $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 021420 000207                        RTS       PC          ;; PICKUP "DATA TABLE" POINTER
(2) 021422                                7$: MOV      @ (RO)+,-(SP) ;; GO TYPE THE DATA
(2) 021422 013046                        TYPOC              ;; RESTORE RO
(2) 021424 104401                        TST      (RO)        ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 021426 005710                        BEQ      6$          ;; RETURN
(1) 021430 001770                        TYPE      $CRLF          ;; SAVE @ (RO)+ FOR TYPEOUT
(1) 021432 104400 021440                BR         6$          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 021436 000771                        TYPE      8$          ;; IS THERE ANOTHER NUMBER?
(1) 021440 020040 000                BR         7$          ;; BR IF NO
(1) 021444 021444                        .ASCIZ  "/ /"        ;; TYPE TWO(2) SPACES
(1)                                .EVEN              ;; LOOP
(1)                                .SBTTL POWER DOWN AND UP ROUTINES ;; TWO(2) SPACES

(1)
(2)
(1)
(1) 021444 012737 021610 000024 $PWRDN: MOV      @SILLUP,@PWRVEC ;; SET FOR FAST UP
(1) 021452 012737 000340 000026    MOV      @340,@PWRVEC+2 ;; PRIO:7
(3) 021460 010046                        MOV      RO,-(SP)    ;; PUSH RO ON STACK
(3) 021462 010146                        MOV      R1,-(SP)    ;; PUSH R1 ON STACK
(3) 021464 010246                        MOV      R2,-(SP)    ;; PUSH R2 ON STACK
(3) 021466 010346                        MOV      R3,-(SP)    ;; PUSH R3 ON STACK
(3) 021470 010446                        MOV      R4,-(SP)    ;; PUSH R4 ON STACK
(3) 021472 010546                        MOV      R5,-(SP)    ;; PUSH R5 ON STACK
(3) 021474 017746 157440                MOV      @SWR,-(SP)  ;; PUSH @SWR ON STACK
(1) 021500 010637 021614                MOV      SP,$SAVR6  ;; SAVE SP
(1) 021504 012737 021516 000024    MOV      @SPWRUP,@PWRVEC ;; SET UP VECTOR
(1) 021512 000000                        HALT

```













```

(1) 022276 000770 BR 7S ;;LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 022300 112716 000040 8S: MOVB #' (SP) ;;REPLACE TAB WITH SPACE
(1) 022304 004737 022324 9S: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 022310 132737 000007 022370 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
(1) 022316 001372 BNE 9S ;;TAB STOP
(1) 022320 005726 TST (SP)+ ;;POP SPACE OFF STACK
(1) 022322 000724 BR 2S ;;GET NEXT CHARACTER
(1) 022324 105777 156620 $TYPEC: TSTB $STPS ;;WAIT UNTIL PRINTER IS READY
(1) 022330 100375 BPL $TYPEC
(1) 022332 116677 000002 156612 MOVB 2(SP),$STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 022340 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 022346 001003 BNE 1S ;;BRANCH IF NO
(1) 022350 105037 022370 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 022354 000406 BR $TYPEX ;;EXIT
(1) 022356 122766 000012 000002 1S: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 022364 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 022366 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 022370 000000 $CHARCNT: WORD 0 ;;CHARACTER COUNT STORAGE
(1) 022372 000207 $TYPEX: RTS PC
(1)
2006 .SBTTL TTY INPUT ROUTINE
(1)
(1) ;*****
(1) .ENABL LSB
(1)
(1) ;*****
(1) ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;*SERVICE THE TEST JR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) ;*WHEN OPERATING IN TTY FLAG MODE.
(1) 022374 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 022402 001074 BNE 1S ;;BRANCH IF NO
(1) 022404 105777 156534 TSTB $STKS ;;CHAR THERE?
(1) 022410 100071 BPL 1S ;;IF NO, DON'T WAIT AROUND
(1) 022412 117746 156530 MOVB $STKB,-(SP) ;;SAVE THE CHAR
(1) 022416 042716 177600 BIC #C177,(SP) ;;STRIP-OFF THE ASCII
(1) 022422 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
(1) 022426 001062 BNE 1S ;;NO, RETURN TO USER
(1) 022430 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 022436 001456 BEQ 1S ;;BRANCH IF YES
(1)
(1) 022440 104400 023121 $GTSWR: TYPE ,SCNTLG ;;ECHO THE CONTROL-G (↑G)
(1) 022444 104400 023126 TYPE $MSWR ;;TYPE CURRENT CONTENTS
(2) 022450 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 022454 104401 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 022456 104400 023137 TYPE ,SMNEW ;;PROMPT FOR NEW SWR
(1) 022462 005046 19S: CLR -(SP) ;;CLEAR COUNTER
(1) 022464 005046 CLR -(SP) ;;THE NEW SWR
(1) 022466 105777 156452 7S: TSTB $STKS ;;CHAR THERE?
(1) 022472 100375 BPL 7S ;;IF NOT TRY AGAIN
(1)
(1) 022474 117746 156446 MOVB $STKB,-(SP) ;;PICK UP CHAR
(1) 022500 042716 177600 BIC #C177,(SP) ;;MAKE IT 7-BIT ASCII

```





```

(1) 022720 105777 156220 2$: TSTB 2$TKS ;: WAIT FOR A CHARACTER
(1) 022724 100375 BPL 2$ ;: LOOP UNTIL ITS THERE
(1) 022726 117746 156214 MOVB 2$TKB, -(SP) ;: GET CHARACTER
(1) 022732 042716 177600 BIC #1C177, (SP) ;: MAKE IT 7-BIT ASCII
(1) 022736 022627 000021 CMP (SP)+, #21 ;: IS IT A CONTROL-Q?
(1) 022742 001366 BNE 2$ ;: IF NOT DISCARD IT
(1) 022744 000750 BR 1$ ;: YES, RESUME
(1) 022746 026627 000004 000140 3$: CMP 4(SP), #140 ;: IS IT UPPER CASE?
(1) 022754 002407 BLT 4$ ;: BRANCH IF YES
(1) 022756 026627 000004 000175 CMP 4(SP), #175 ;: IS IT A SPECIAL CHAR?
(1) 022764 003003 BGT 4$ ;: BRANCH IF YES
(1) 022766 042766 000040 000004 BIC #40, 4(SP) ;: MAKE IT UPPER CASE
(1) 022774 000002 4$: RTI ;: GO BACK TO USER
(2) ;: *****
(1) ;: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) ;: *CALL:
(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) 022776 010346 $RDLIN: MOV R3, -(SP) ;: SAVE R3
(1) 023000 012703 023104 1$: MOV #STTYIN, R3 ;: GET ADDRESS
(1) 023004 022703 023114 2$: CMP #STTYIN+8., R3 ;: BUFFER FULL?
(1) 023010 101405 BLOS 4$ ;: BR IF YES
(1) 023012 104407 RDCHR ;: GO READ ONE CHARACTER FROM THE TTY
(1) 023014 112613 MOVB (SP)+, (R3) ;: GET CHARACTER
(1) 023016 122713 000177 10$: CMPB #177, (R3) ;: IS IT A RUBOUT
(1) 023022 001003 BNE 3$ ;: SKIP IF NOT
(1) 023024 104400 001170 4$: TYPE $QUES ;: TYPE A '?'
(1) 023030 000763 BR 1$ ;: CLEAR THE BUFFER AND LOOP
(1) 023032 111337 023102 3$: MOVB (R3), 9$ ;: ECHO THE CHARACTER
(1) 023036 104400 023102 TYPE 9$
(1) 023042 122723 000015 CMPB #15, (R3)+ ;: CHECK FOR RETURN
(1) 023046 001356 BNE 2$ ;: LOOP IF NOT RETURN
(1) 023050 105063 177777 CLRB -1(R3) ;: CLEAR RETURN (THE 15)
(1) 023054 104400 001172 TYPE $LF ;: TYPE A LINE FEED
(1) 023060 012603 MOV (SP)+, R3 ;: RESTORE R3
(1) 023062 011646 MOV (SP), -(SP) ;: ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 023064 016666 000004 000002 MOV 4(SP), 2(SP) ;: FIRST ASCII CHARACTER ON IT
(1) 023072 012766 023104 000004 MOV #STTYIN, 4(SP)
(1) 023100 000002 RTI ;: RETURN
(1) 023102 000 9$: .BYTE 0 ;: STORAGE FOR ASCII CHAR. TO TYPE
(1) 023103 000 .BYTE 0 ;: TERMINATOR
(1) 023104 000010 $TTYIN: .BLKB 8. ;: RESERVE 8 BYTES FOR TTY INPUT
(1) 023114 052536 005015 000 $CNTLU: .ASCIZ /?U/<15><12> ;: CONTROL "U"
(1) 023121 136 006507 000012 $CNTLG: .ASCIZ /?G/<15><12> ;: CONTROL "G"
(1) 023126 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 023134 020075 000 $MNEW: .ASCIZ / NEW = /
(1) 023137 040 047040 053505 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1) 023144 036440 000040
2007
(1)
(2) ;: *****
(1) ;: *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)          $RDOCT: MOV      (SP), -(SP)  ;; PROVIDE SPACE FOR THE
(1) 023150 011646 000004 000002 MOV      4(SP), 2(SP)  ;; INPUT NUMBER
(1) 023152 016666 000004 000002 MOV      RO, -(SP)    ;; PUSH RO ON STACK
(3) 023160 010046 000004 000002 MOV      R1, -(SP)    ;; PUSH R1 ON STACK
(3) 023162 010146 000004 000002 MOV      R2, -(SP)    ;; PUSH R2 ON STACK
(3) 023164 010246 000004 000002 1$: RDLIN  ;; READ AN ASCII LINE
(1) 023166 104410 000004 000002 MOV      (SP)+, RO    ;; GET ADDRESS OF 1ST CHARACTER
(1) 023170 012600 000004 000002 CLR      R1          ;; CLEAR DATA WORD
(1) 023172 005001 000004 000002 CLR      R2
(1) 023174 005002 000004 000002 2$: MOVB   (RO)+, -(SP)  ;; PICKUP THIS CHARACTER
(1) 023176 112046 000004 000002 BEQ      3$          ;; IF ZERO GET OUT
(1) 023200 001412 000004 000002 ASL     R1          ;; *2
(1) 023202 006301 000004 000002 ROL     R2
(1) 023204 006102 000004 000002 ASL     R1          ;; *4
(1) 023206 006301 000004 000002 ROL     R2
(1) 023210 006102 000004 000002 ASL     R1          ;; *8
(1) 023212 006301 000004 000002 ROL     R2
(1) 023214 006102 000004 000002 BIC     #1C7, (SP)  ;; STRIP THE ASCII JUNK
(1) 023216 042716 177770 000002 ADD     (SP)+, R1    ;; ADD IN THIS DIGIT
(1) 023222 062601 000004 000002 BR      2$          ;; LOOP
(1) 023224 000764 000004 000002 3$: TST   (SP)+    ;; CLEAN TERMINATOR FROM STACK
(1) 023226 005726 000012 000012 MOV     R1, 12(SP)  ;; SAVE THE RESULT
(1) 023230 010166 000012 023250 MOV     R2, $HIOCT
(1) 023234 010237 023250 000012 MOV     (SP)+, R2    ;; POP STACK INTO R2
(3) 023240 012602 000012 000012 MOV     (SP)+, R1    ;; POP STACK INTO R1
(3) 023242 012601 000012 000012 MOV     (SP)+, RO    ;; POP STACK INTO RO
(3) 023244 012600 000012 000012 RTI          ;; RETURN
(1) 023246 000002 000012 000012 $HIOCT: .WORD 0  ;; HIGH ORDER BITS GO HERE
(1) 023250 000000 000012 000012 .SBTTL  APT COMMUNICATIONS ROUTINE

2008
(1)          ;; *****
(2)          $ATY1: MOVB   #1, $FFLG  ;; TO REPORT FATAL ERROR
(1) 023252 112737 000001 023516 $ATY3: MOVB   #1, $MFLG  ;; TO TYPE A MESSAGE
(1) 023260 112737 000001 023514 BR      $ATYC
(1) 023266 000403 000001 023516 $ATY4: MOVB   #1, $FFLG  ;; TO ONLY REPORT FATAL ERROR
(2) 023276 000403 000001 023516 $ATYC:
(3) 023276 010046 000001 023516 MOV     RO, -(SP)  ;; PUSH RO ON STACK
(3) 023300 010146 000001 023516 MOV     R1, -(SP)  ;; PUSH R1 ON STACK
(1) 023302 105737 023514 TSTB   $MFLG  ;; SHOULD TYPE A MESSAGE?
(1) 023306 001450 000001 001214 BEQ     5$          ;; IF NOT: BR
(1) 023310 122737 000001 001214 CMPB   #APTENV, $ENV  ;; OPERATING UNDER APT?
(1) 023316 001031 000001 001214 BNE    3$          ;; IF NOT: BR
(1) 023320 132737 000100 001215 BITB   #APTPOOL, $ENVM  ;; SHOULD SPOOL MESSAGES?
(1) 023326 001425 000100 001215 BEQ     3$          ;; IF NOT: BR
(1) 023330 017600 000004 000004 MOV     #4(SP), RO    ;; GET MESSAGE ADDR.
(1) 023334 062766 000002 000004 ADD     #2, 4(SP)    ;; BUMP RETURN ADDR.
(1) 023342 005737 001174 000004 1$: TST   $MSGTYPE  ;; SEE IF DONE W/ LAST XMISSION?
(1) 023346 001375 001210 000004 BNE    1$          ;; IF NOT: WAIT
(1) 023350 010037 001210 000004 MOV     RO, $MSGAD  ;; PUT ADDR IN MAILBOX
(1) 023354 105720 001210 000004 2$: TSTB   (RO)+    ;; FIND END OF MESSAGE
(1) 023356 001376 001210 000004 BNE    2$
(1) 023360 163700 001210 000004 SUB     $MSGAD, RO  ;; SUB START OF MESSAGE

```



```

(1) 023364 006200 ASR RO ;;GET MESSAGE LNGTH IN WORDS
(1) 023366 010037 001212 MOV RO,SMSGLGT ;;PUT LENGTH IN MAILBOX
(1) 023372 012737 000004 001174 MOV #4,SMSGTYPE ;;TELL APT TO TAKE MSG.
(1) 023400 000413 BR 5$
(1) 023402 017637 000004 023426 3$: MOV 24(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 023410 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 023416 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 023422 004737 022112 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 023426 000000 4$: .WORD 0
(1) 023430 5$:
(1) 023430 105737 023516 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 023434 001416 BEQ 12$ ;;IF NOT: BR
(1) 023436 005737 001214 TST $ENV ;;RUNNING UNDER APT?
(1) 023442 001413 BEQ 12$ ;;IF NOT: BR
(1) 023444 005737 001174 11$: TST SMSGTYPE ;;FINISHED LAST MESSAGE?
(1) 023450 001375 BNE 11$ ;;IF NOT: WAIT
(1) 023452 017637 000004 001176 MOV 24(SP),$FATAL ;;GET ERROR #
(1) 023460 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 023466 005237 001174 INC SMSGTYPE ;;TELL APT TO TAKE ERROR
(1) 023472 105037 023516 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) 023476 105037 023515 CLRB $LFLG ;;CLEAR LOG FLAG
(1) 023502 105037 023514 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) 023506 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 023510 012600 MOV (SP)+,RO ;;POP STACK INTO RO
(1) 023512 000207 RTS PC ;;RETURN
(1) 023514 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 023515 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 023516 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 023520 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPool=100
(1) 000040 APTCSUP=040

```

2009  
2010

.SBTTL TRAP DECODER

```

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

```

```

(1) 023520 010046 STRAP: MOV RO,-(SP) ;;SAVE RO
(1) 023522 016600 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS
(1) 023526 005740 TST -(RO) ;;BACKUP BY 2
(1) 023530 111000 MOV#B (RO),RO ;;GET RIGHT BYTE OF TRAP
(1) 023532 006300 ASL RO ;;POSITION FOR INDEXING
(1) 023534 016000 023542 MOV STRPAD(RO),RO ;;INDEX TO TABLE
(1) 023540 000200 RTS RO ;;GO TO ROUTINE

```

.SBTTL TRAP TABLE

```

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

```

```

(3) ; ROUTINE

```

```

(3)
(3) 023542
(3) 023542 022112
(3) 023544 021710
(3) 023546 021664
(3) 023550 021724
(3) 023552 020520
(1)
(3) 023554 022444
(1)
(3) 023556 022374
(3) 023560 022656
(3) 023562 022776
(3) 023564 023150
2011
2012 023566 000074
2013 023756 000074
2014 024146 000074
2015
2016 024150 000240
2017 000001

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

          SGTSMR ;;CALL=GTSMR     TRAP+5(104405) GET SOFT-SMR SETTING

          SCKSMR ;;CALL=CKSMR     TRAP+6(104406) TEST FOR CHANGE IN SOFT-SMR
          SRDCHR  ;;CALL=RDCHR     TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
          SRDLIN  ;;CALL=RDLIN     TRAP+10(104410) TTY TYPEIN STRING ROUTINE
          SRDOCT  ;;CALL=RDOCT     TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY

          BUF2:  .BLKW  60.
          BUF3:  .BLKW  60.
          BUFFER: 60.

          NOP
          .END

```













































DEC	555	584	612	631	633	645	660	750	767	791	889	897	905	913	948
	961	1016	1176	1190	1573	1577	1643	1666	1669	1729	1733	1740	1749	1751	1765
	1790	1834	1999												
DECB	2004	2005													
EMT	14														
HALT	23	1678	1998	2000	2005										
INC	234	536	777	791	812	850	886	894	902	910	943	956	1012	1070	1095
	1099	1103	1372	1697	1830	1996	1997	1998	2000	2004	2006	2008			
INCB	746	764	1997	1998	2005										
IOT	14														
JMP	23	24	25	138	214	215	216	217	218	219	788	791	1066	1205	1535
	1541	1590	1602	1778											
JSR	185	258	791	821	822	823	833	834	835	864	915	921	963	970	995
	997	1000	1002	1006	1035	1045	1054	1063	1064	1069	1161	1174	1188	1192	1199
	1200	1201	1203	1216	1221	1223	1225	1227	1234	1242	1294	1299	1303	1309	1313
	1317	1323	1334	1345	1354	1363	1388	1399	1434	1438	1441	1444	1447	1451	1454
	1457	1460	1463	1467	1470	1475	1485	1488	1492	1496	1500	1503	1506	1510	1514
	1518	1523	1529	1546	1549	1616	1618	1629	1631	1633	1635	1668	1703	1760	1807
	1811	1997	1998	2005	2006	2008									
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	618	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	814	815	818	819	820	826	827
	830	831	832	842	843	844	847	848	856	857	863	865	866	869	870
	871	872	875	876	881	892	900	908	920	922	923	926	927	928	929
	932	933	939	953	955	969	982	983	985	986	989	990	991	992	1010
	1011	1034	1036	1037	1038	1041	1042	1043	1046	1047	1050	1051	1052	1055	1056
	1059	1060	1061	1067	1068	1077	1079	1080	1087	1088	1089	1093	1094	1101	1160
	1162	1163	1164	1165	1166	1169	1170	1171	1172	1173	1178	1179	1180	1183	1184
	1185	1186	1187	1198	1215	1220	1222	1224	1226	1241	1250	1251	1252	1253	1298
	1301	1302	1308	1311	1312	1321	1322	1325	1332	1333	1336	1343	1344	1347	1353
	1356	1351	1362	1365	1370	1371	1373	1378	1379	1386	1387	1390	1397	1398	1401
	1412	1413	1420	1427	1428	1436	1456	1464	1480	1505	1513	1538	1545	1551	1552
	1555	1559	1563	1567	1569	1571	1572	1576	1586	1598	1615	1619	1620	1621	1622
	1623	1628	1632	1638	1639	1640	1641	1642	1658	1660	1683	1684	1687	1688	1690
	1691	1692	1693	1694	1723	1728	1732	1738	1744	1746	1747	1761	1776	1789	1802
	1803	1804	1805	1810	1824	1825	1827	1829	1841	1856	1857	1858	1859	1865	1866
	1870	1871	1996	1997	1998	1999	2000	2004	2005	2006	2007	2008	2010		
MOV B	164	1090	1724	1727	1996	1997	1999	2004	2005	2006	2007	2008	2010		
NEG	1862	1996	2004												
NOP	260	577	791	1036	1162	1236	1237	2016							
RESET	134	424	678	686	695	704	714	722	738	739	740	741	773	776	791
	1204														
ROL	2004	2007													
ROLB	1091														
RTI	136	1996	1997	1998	2000	2004	2005	2006	2007						
RTS	168	855	1019	1072	1106	1235	1591	1605	1645	1679	1706	1742	1753	1768	1794

SUB	1842	1868	1872	1999	2005	2008	2010	1816	1860	1996	1998	2008			
SWAB	167	901	909	959	1696	1704	1815								
TRAP	1826														
TST	2010														
TSTB	146	157	186	188	232	237	241	275	276	277	278	279	806	1004	1018
	1232	1676	1996	1997	1998	1999	2004	2005	2006	2007	2008	2010			
	549	553	567	576	582	598	606	610	625	629	748	765	851	887	895
	903	911	944	950	957	1013	1084	1096	1584	1596	1664	1698	1831	1996	1997
	2005	2006	2008												
.ASCII	30	178	179	180	181	182	183	1877	1878	1879	1880	1881	1882	1883	1884
.ASCIZ	1888	1889	1890	1891	1925	1926	1929	1931	1948	1951					
	30	791	1318	1885	1886	1887	1892	1893	1894	1895	1896	1897	1898	1899	1900
	1901	1902	1903	1904	1905	1906	1907	1908	1909	1910	1915	1916	1919	1921	1923
	1932	1937	1938	1939	1940	1941	1942	1943	1944	1999	2001	2006			
.BLKB	2006														
.BLKW	1996	2012	2013												
.BYTE	30	247	248	791	1117	1118	1119	1120	1121	1122	1123	1124	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	1912	1913
	1914	1917	1918	1920	1922	1924	1927	1928	1930	1947	1949	1950	1952	1998	2004
	2006	2008													
.DSABL	2006														
.ENABL	4	2006													
.END	2017														
.ENDC	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	1297	1304	1307	1314	1316
	1318	1320	1327	1330	1338	1342	1349	1351	1358	1360	1367	1369	1375	1381	1385
	1392	1396	1403	1409	1415	1422	1430	1436	1480	1812	1817	1996	1997	1998	1999
	2000	2004	2005	2006	2007	2008	2010								
.EQUIV	14														
.EVEN	30	1157	1318	1953	1999	2002	2008								
.IF	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	1297	1304	1307	1314	1316
	1318	1320	1327	1330	1338	1342	1349	1351	1358	1360	1367	1369	1375	1381	1385
	1392	1396	1403	1409	1415	1422	1430	1436	1480	1812	1817	1996	1997	1998	1999
	2000	2004	2005	2006	2007	2008	2010								
.IFF	14	21	27	29	30	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	1297	1304	1307	1314	1316	1320	1327	1330	1338
	1342	1349	1351	1358	1360	1367	1369	1375	1381	1385	1392	1396	1403	1409	1415
	1422	1430	1436	1480	1812	1817	1996	1997	1998	1999	2000	2004	2005	2006	2007
.IFT	1318	1997	1998	2006	2007										
.IFTF	1319	1997	1998	2006	2007										
.IIF	13	21	23	30	136	791	1997	1998	1999	2005	2006	2010			
.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	1297	1307	1320	1330	1342	1351	1360	1369	1385	1396	1409	1436	1480	1996
.LIST	1997	1998	2000	2007	2008										
	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	1031	1212	1292	1297
	1307	1318	1320	1330	1342	1351	1360	1369	1385	1396	1409	1436	1480	1651	1997
.MACRO	1998	2006	2010												
.MCALL	21	30	136	261	264	268	726	1022	1934	2010					
.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	1021	1207	1254	1297
	1307	1318	1320	1330	1342	1351	1360	1369	1385	1396	1409	1436	1480	1647	1997
.PAGE	1998	2006	2010												
.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	813	825	860	917	967
	1032	1158	1196	1208	1209	1210	1211	1213	1240	1297	1307	1320	1330	1342	1351
	1360	1369	1385	1396	1409	1436	1480	1543	1613	1626	1648	1649	1650	1654	1681
.TITLE	1756	1875	1996	1997	1998	1999	2000	2004	2005	2006	2007	2008	2010		
.WORD	13	27	29	30	791	1999	2000	2004	2005	2007	2008				

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

\*DZAACA, DZAACA.ASS/CRF/PAGNUM=DSKZ:DZAACA P11  
RUN-TIME: 76 43 8 SECONDS  
RUN-TIME RATIO: 416/129=3.2  
CORE USED: 27K (53 PAGES)





	THE SOFT	...	B1	1205	011360	000137	...	B5
	OVER THE PROGRAM	...	C1	1249			...	C5
	1. TYPE	...	D1	1326	011762	023737	...	D5
	AND	...	E1	1348	012076	023737	...	E5
	SWITCH	...	F1	1391	012352	023737	...	F5
	-----	...	G1	1415	012470	001401	...	G5
	1. TYPI	...	H1	1441	012612	004537	...	H5
	IS INCREMENTED W	...	I1	1485	012764	004537	...	I5
	PHOSPHOR AND ERA	...	J1	1530	013134	001454	...	J5
	WHICH TIME THE P	...	K1	1552	013216	010001	...	K5
136	INITIALI	...	L1	1590	013364	000137	...	L5
796	END OF P	...	M1	1621	013502	010077	...	M5
(3)	TRAP TAB	...	N1	1663	013652	001424	...	N5
(1)		...	B2	1718			...	B6
(1)	000004	...	C2	1765	014234	005337	...	C6
16	170416	...	D2	1807	014376	004537	...	D6
(1)		...	E2	1831	014506	105777	...	E6
(1)	000214	...	F2		014734	041501	...	F6
(1)	001102	000	G2		015426	047122	...	G6
(2)		...	H2	1895	016104	040504	...	H6
(1)		...	I2	1906	016602	042523	...	I6
80	001352	020346	J2	1924	017232	015	...	J6
(1)		...	K2	(1)	017704	051525	...	K6
149		...	L2	1956	020310	001116	...	L6
186	002132	005737	M2	(1)			...	M6
236	002364	022626	N2	(1)	020732	000012	...	N6
291	002600	104002	B3	1998			...	B7
317	002744	104003	C3	(2)	021332	104401	...	C7
343	003106	001401	D3	(3)	021542	012677	...	D7
370	003244	023737	E3	(1)	021710	112737	...	E7
398	003424	012737	F3	(1)			...	F7
427	003610	001401	G3	(1)	022322	000724	...	G7
465	004026	001401	H3	(1)	022522	000757	...	H7
495	004210	104001	I3	(1)	022756	026627	...	I7
528	004402	001401	J3	(1)	023166	104410	...	J7
549	004510	105737	K3	(1)	023430		...	K7
598	004730	105737	L3	(1)	023554	022444	...	L7
644	005162	012777	M3	ADDW1 =	000000		...	M7
680	005374	023737	N3	BEGIN	001462		...	N7
714	005636	000005	B4	CH12 =	000012		...	B8
(1)	006120	023737	C4	DH1	016645		...	C8
764	006332	105277	D4	IOTVEC=	000020		...	D8
(1)	006500		E4	PIC7	011022		...	E8
814	006650	013700	F4				...	F8
869	007146	013701	G4	SW1 =	000002		...	G8
925	007374	001404	H4	TST36	005124		...	H8
975	007574	005077	I4				...	I8
(1)	010050	013737	J4	\$FFLG	023516		...	J8
1084	010370	105777	K4	\$OMODE	022110		...	K8
1121	010542	177	L4	\$TSTNM	001102		...	L8
1149	010756	076	M4	ESCAPE	14#	1315	...	M8
1163	011076	013705	N4		620	628	...	N8

IOT	351	359	...	B9
.ASCII	14	764	...	C9
.IRP	30	178	...	D9
	32	135	...	E9
**END**	USER DAVIES, TOM		...	F9