

NCV11

NCV-11 DIAG
CZNCCCO

AH-E772C-MC
FICHE 1 OF 1

FEB 1981
COPYRIGHT © 78-80
MADE IN USA



IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-E771C-MC
DIAGNOSTIC CODE: MAINDEC-11-CZNCC-C
PRODUCT NAME: CZNCCC NCV11 DIAGNOSTIC TEST
DATE: AUG. 4, 1980
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1978, 1979, 1980
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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

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

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

TABLE OF CONTENTS

0.0	HISTORY
1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	PROGRAM START
5.0	SOFTWARE SWITCH REGISTER
5.1	LOGIC TEST OPTIONS
5.2	FIELD ADJUSTMENT OPTIONS
5.3	CONTROL
6.0	ERROR REPORTING
7.0	MISCELLANEOUS
7.1	NCV11 BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE NCV11 INTERFACE TESTING
7.5	RESTRICTIONS
7.6	ADDRESS MAKER TABLE
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
10.0	LISTING

0.0 HISTORY

VERSION "C" WAS CREATED TO:

- #1 INSTALL A 3 LOCATION PATCH TO LOGIC TEST 134.
- #2 CORRECT A BUG IN LOGIC TEST 151.

VERSION "B" WAS CREATED TO:

- #1 INSTALL A 2 LOCATION PATCH TO THE LOGIC TEST.
- #2 INSTALL A 3 BYTE PATCH TO THE M8036 MICRO-CODE.
- #3 CHANGE THE OUTPUT REPORT OF DIFFERENTIAL LINEARITY.
- #4 CORRECT A DEFECT IN ACCOUNTING ENORMOUS ERRORS IN TOTAL ERROR COUNTER.
- #5 CORRECT ERROR CODES FOR TWO TESTS.

1.0 ABSTRACT

ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE ON THE M8026 AND M8036. THE M7952 DIAGNOSTIC SHOULD HAVE BEEN EXECUTED. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA KEYBOARD TEST SELECTION AND THE PROVISIONS OF SECTION 5. THE PROGRAM CAN BE EXECUTED ON AN LSI-11 OR PDP-11 COMPUTER.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11 FAMILY OR LSI-11 FAMILY COMPUTER
2. I/O TERMINAL (IE: LA36)
3. NCV11 OPTION INCLUDING KVV11 REAL TIME CLOCK (M7952)
4. A017 SELF-TEST CONNECTOR (70-12894) (REQUIRED FOR SECTION 9.2 + 9.3)
5. H3060 JOYSTICK (OPTIONAL)
6. HARDWARE TESTER AND PROM BLASTER (OPTIONAL)

2.2 STORAGE

THE PROGRAM REQUIRES 16K OF MEMORY. IF MEMORY MANAGEMENT OR ADDITIONAL MEMORY ARE SENSED, THE PROGRAM WILL USE THOSE HARDWARE OPTIONS.

2.3 PRELIMINARY PROGRAMS

THE KVV11 DIAGNOSTIC (MD-11-CVKWA) SHOULD HAVE BEEN RUN.

3.0 LOADING PROCEDURE

NORMAL PROCEDURE FOR LOADING A BINARY PROGRAM INTO MEMORY SHOULD BE FOLLOWED.

4.0 STARTING PROCEDURE

1. MAKE SURE THE NCV11 DEVICE BUS ADDRESS, INTERRUPT VECTOR ON THE M8026 AGREE WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1.
2. MAKE SURE THE NCV11 CLOCK BUS ADDRESS ON THE M7952 AGREE WITH THE DEFAULT VALUE DEFINED IN SECTION 7.1.
3. MAKE SURE THE NCV11 INTERRUPT PRIORITY LEVEL ON THE M8217 AGREE WITH THE DEFAULT VALUE DEFINED IN SECTION 7.1.
4. INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
- LSI11: 5. TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
6. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.
- LSI11: 7. THE PROGRAM WILL REQUEST SWITCH REGISTER VALUE.
8. THE PROGRAM WILL TYPE THE TEST SELECTION MENU FOR THE OPERATOR.
9. THE OPERATOR SELECT'S THE APPROPRIATE TEST BY TYPING THE TEST LETTER AND "CR"

IF THE PROGRAM IS EXECUTED ON A CPU WITH A SWITCH REGISTER AND THE OPERATOR SETS ALL SWITCHES TO A 1, THE SOFTWARE S.R. WILL BE USED. IF THE HARDWARE SWITCH REGISTER IS USED, THE SOFTWARE S.R. HAS NO EFFECT.

4.1 PROGRAM START

200
204
210

STARTING ADDRESS OF THE PROGRAM
RESTART ADDRESS OF THE PROGRAM
STARTING ADDRESS FOR THE FACTORY OPTION CHECK-OUT AREA.
(INFORMS THE PROGRAM THE TESTER AND BLASTER ARE CONNECTED)

5.0 SOFTWARE SWITCH REGISTER

5.1 LOGIC TEST OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <7-0>

5.2 FIELD ADJUSTMENT LOOP

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON "INPUT VOLTAGE OUT OF RANGE" ERROR
SW13=1	020000	INHIBIT ERROR OR CONVERSION TYPEOUT
SW09=1	001000	INHIBIT CONVERSIONS ON CAMERA #3 GAIN = 1
SW08=1	000400	INHIBIT CONVERSIONS ON CAMERA #2 GAIN = 1
SW07=1	000200	INHIBIT CONVERSIONS ON CAMERA #1 GAIN = 1
SW06=1	000100	INHIBIT CONVERSIONS ON CAMERA #0 GAIN = 1
SW04=1	000020	INHIBIT CONVERSIONS ON JOYSTICK
SW03=1	000010	INHIBIT CONVERSIONS ON CAMERA #3 GAIN = 2
SW02=1	000004	INHIBIT CONVERSIONS ON CAMERA #2 GAIN = 2
SW01=1	000002	INHIBIT CONVERSIONS ON CAMERA #1 GAIN = 2
SW00=1	000001	INHIBIT CONVERSIONS ON CAMERA #0 GAIN = 2

5.3 CONTROL

1. THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
- LSI-11: 3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & C' OR 'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.
5. TESTING CAN BE ABORTED BY TYPING THE 'CONTROL & C' KEYS.

6.0 ERROR REPORTING

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

7.0 MISCELLANEOUS

7.1 NCV11 BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1250) IF BASE NCV11 BUS ADDRESS IS NOT 772760.
MODIFY THE LOW NINE BITS OF LOCATION '\$VECT1' (LOC. 1244) IF BASE NCV11 INTERRUPT VECTOR IS NOT 370.
MODIFY THE HIGH THREE BITS OF LOCATION '\$VECT1' (LOC. 1244) IF THE INTERRUPT PRIORITY LEVEL IS NOT LEVEL 6 ON A UNIBUS CPU.
MODIFY LOCATION '\$CDW1' (LOC. 1254) IF BASE NCV11 CLOCK BUS ADDRESS IS NOT 770420.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REQUIRES 16K OR MORE).
IF RUNNING UNDER XXDP/ACT/APT, THE LOGIC AND DIFLIN TESTS WILL BE RUN.

7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

7.4 MULTIPLE NCV11'S INTERFACE TESTING.

THIS PROGRAM DOES NOT 'AUTO-SIZE' THE NUMBER OF NCV11'S CONNECTED. FOR EACH ADDITIONAL NCV11, THE OPERATOR MUST MODIFY LOCATIONS \$BASE, \$VECT1 AND \$CDW1 (REF. 7.1).

7.5 RESTRICTIONS

KWV11 REAL TIME CLOCK (M7952) MUST NOT BE CONNECTED TO M8026 'FAST-ON' TABS. UNLESS OTHERWISE STATED, NO EXTERNAL CONNECTIONS TO M8036 OR A017.

7.6

ADDRESS MAKER TABLE

THE M8036 CONTAINS LOGIC TO CONVERT A/D DATA INTO A RELATIVE ADDRESS. BELOW IS A TABLE WHICH SHOWS THE ADDRESS MAKER OUTPUT WITH THE "TEST CONTROL" MAINT. BIT SET. SETTING THIS BIT ENABLES THE STATES OF THE "GAIN, ZB ENABLE AND RESOLUTION" BITS SIMULATE THE CONVERTED A/D DATA.

RESOLUTION ADDRESS MAKER OUTPUT BITS
RES0 RES1 WORD*16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

1	1	1	B	G	Z	G	Z	G	1	1	1	G	Z	G	Z	G	1	1	1
1	1	0	0	B	G	Z	G	Z	G	1	1	G	Z	G	Z	G	1	1	0
1	0	1	0	0	B	G	Z	G	Z	G	1	0	G	Z	G	Z	G	1	0
1	0	0	0	0	0	B	G	Z	G	Z	G	1	G	Z	G	Z	G	1	0
0	1	1	0	0	0	0	B	G	Z	G	Z	G	0	G	Z	G	Z	G	0
0	1	0	0	0	0	0	B	G	Z	G	Z	G	Z	G	Z	G	Z	G	0
0	0	1	0	0	0	0	0	B	G	Z	G	Z	G	Z	G	Z	G	Z	G

0 = ALWAYS A 0
1 = ALWAYS A 1
G = GAIN FLOP STATE
Z = ZB ENABLE FLOP STATE
B = THE "AND" OF ZB ENABLE WITH ZB MAINT. INPUT

8.0

EXECUTION TIME

LOGIC: EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS TO ABOUT 70 SECONDS WITH ITERATIONS ENABLED. AN END PASS MESSAGE INDICATES ALL SUB-TESTS HAVE COMPLETED. EACH ADDITIONAL 4K OF MEMORY WILL ADD ABOUT 10 SECONDS TO THE LOGIC TEST RUNTIME. DIFFERENTIAL LINEARITY: EXECUTION TIME IS ABOUT 90 SECONDS.

9.0 PROGRAM TEST DESCRIPTIONS

SEQ 0007

THE PROGRAM CONSISTS OF FOUR MAIN SECTIONS PLUS THREE AUXILIARY AND THREE FACTORY OPTION CHECK-OUT SECTIONS. THE OPERATOR SELECTS EACH SECTION VIA THE CONSOLE KEYBOARD. BELOW IS A BRIEF DESCRIPTION OF EACH SECTION AND THE KEYBOARD CHARACTER.

MAIN SECTIONS

9.1 L LOGIC TEST (M8026 + M8036)

<NO INTERVENTION> <SETUP ONLY IF SA=210>

THE TEST CONTAINS A SERIES OF INDEPENDENT SUB-TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE NCV11 GAMMA CAMERA INTERFACE. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH SUB-TEST TITLE CAN BE BENEFICIAL TO UNDERSTANDING THE LOGIC TESTED. ADDITIONAL LOGIC TESTS OF THE A017 WILL BE EXECUTED IF THE PROGRAM WAS STARTED AT LOC. 210.

9.2 D (M) DIFFERENTIAL LINEARITY (A017)

<REQUIRES SELF-TEST CONNECTOR IN J1 ON THE A017> <SETUP INTERVENTION ONLY>
<A017 SWITCH SET TO 'MAINT.' POSITION>

A FINITE VARIATION OF INPUT VOLTAGE EQUALS A GIVEN CONVERTED VALUE OR STATE. THE TEST IS TO VERIFY THE WIDTH OF EACH POSSIBLE STATE. AN INPUT VOLTAGE IS RAMPED AND THE NCV11 SAMPLES THE RESULT. THE NCV11 INCREMENTS THE RESPECTIVE MEMORY LOCATION CORRESPONDING TO THE INPUT VALUE. THE PROGRAM THEN DETERMINES IF ANY STATE HAS BEEN SKIPPED, IS EXCESSIVELY WIDE OR NARROW. WHEN 'M' IS USED TO SELECT THE SECTION, THE PROGRAM OUTPUT AN EXPANDED REPORT OF THE 254. STATES. TIGHTER ERROR TOLERANCES ARE USED IF RUNNING IN THE OPTION CHECKOUT AREA (SA 210).

9.3 F FINAL ACCEPTANCE

<REQUIRES SELF-TEST CONNECTOR IN J1 ON THE A017> <SETUP INTERVENTION ONLY>
<A017 SWITCH SET TO 'MAINT.' POSITION>

THE SUB-SECTION RUNS SECTION 'L' AND 'D' TO VERIFY PROPER OPERATION.

9.4 A ADJUSTMENT OF A017 AT FIELD SITE

<MANUAL INTERVENTION> <SETUP INTERVENTION>
 <A017 SWITCH SET TO 'OPER.' POSITION>

COARSE ADJUSTMENT OF THE A017 CAN BE PERFORMED USING THIS SUB-SECTION.
 THE OPERATOR MAY SELECT THE CAMERA AND GAIN TO BE SAMPLED USING THE
 SWITCH REGISTER. A SERIES OF CONVERSIONS ARE TAKEN ON EACH CAMERA
 AND THE OPERATOR IS INFORMED OF THE RESULTS.

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON "INPUT VOLTAGE OUT OF RANGE" ERROR
SW13=1	020000	INHIBIT ERROR OR CONVERTED VALUE TYPE-OUT
SW09=1	001000	INHIBIT CONVERSIONS ON CAMERA #3 GAIN = 1
SW08=1	000400	INHIBIT CONVERSIONS ON CAMERA #2 GAIN = 1
SW07=1	000200	INHIBIT CONVERSIONS ON CAMERA #1 GAIN = 1
SW06=1	000100	INHIBIT CONVERSIONS ON CAMERA #0 GAIN = 1
SW04=1	000020	INHIBIT CONVERSIONS ON JOYSTICK
SW03=1	000010	INHIBIT CONVERSIONS ON CAMERA #3 GAIN = 2
SW02=1	000004	INHIBIT CONVERSIONS ON CAMERA #2 GAIN = 2
SW01=1	000002	INHIBIT CONVERSIONS ON CAMERA #1 GAIN = 2
SW00=1	000001	INHIBIT CONVERSIONS ON CAMERA #0 GAIN = 2

AUXILIARY SECTION

9.5 O OTHER TERMINAL ADDRESS

IN SOME CUSTOMER SITES, THE CONSOLE TERMINAL MAY NOT BE LOCATED
 WITHIN EASY ACCESS TO THE NCV11. THIS SECTION ALLOWS THE OPERATOR
 TO CHANGE THE PROGRAMS CONTROL TO ANOTHER TERMINAL.

9.6 H HELP THE OPERATOR

A LIST OF THE COMMANDS WILL BE TYPED TO REFRESH THE OPERATOR'S MEMORY.

9.7 S SOFTWARE SWITCH REGISTER

ENABLES THE OPERATOR TO CHANGE THE SOFTWARE SWITCH REGISTER WHEN
 NOT RUNNING ANOTHER SUB-SECTION.

FACTORY OPTION CHECK-OUT SECTION

9.8 I INITIAL ADJUSTMENT OF A017

<MANUAL INTERVENTION> <SETUP INTERVENTION> <HARDWARE TESTER>
 <A017 SWITCH SET TO 'OPER.' POSITION>

A KNOWN GOOD DIGITAL TO ANALOG (D/A) SUPPLIES A REFERENCE INPUT
 VOLTAGE. THE OPERATOR IS INFORMED WHICH ADJUSTMENT TO PERFORM.
 THE PURPOSE OF THE TEST IS TO VERIFY THAT THE ADJUSTMENT TO THE PRE-AMP
 SECTION CAN BE PERFORMED. THE TEST DOES NOT ATTEMPT TO CALIBRATE
 THE PRE-AMP, BUT ONLY TO FUNCTIONALLY VERIFY THE ADJUSTMENTS.

9.9 B BLASTING THE LINEARITY CORRECTION PROM

SEQ 0009

<MANUAL INTERVENTION> <SETUP INTERVENTION> <HARDWARE TESTER>
<A017 SWITCH SET TO 'MAINT.' POSITION>

THE TEST IS DESIGNED TO CREATE THE LINEARITY CORRECTION PROM (LCP). THE TESTER HARDWARE CONTAINS RAM STORAGE TO EMULATE THE LCP. THE PROGRAM WILL RUN DIFFERENTIAL LINEARITY. THE PROGRAM WILL THEN USE THE RESULTS TO DETERMINE THE CORRECT VALUE TO BE PLACED IN THE LCP. THE PROGRAM WILL THEN INFORM THE PROM BLASTER TO GENERATE A CORRECTED LINEARITY PROM FOR THE A017 MODULE. UPON COMPLETION OF GENERATION AND VERIFICATION STAGE, THE OPERATOR IS INSTRUCTED TO REMOVE THE EMULATOR CABLE FROM THE A017 AND INSERT THE NEWLY CREATED LCP. THE PROGRAM WILL RE-EXECUTE DIFFERENTIAL LINEARITY.

9.10 C CONTROL PROGRAM PROM

<MANUAL INTERVENTION> <SETUP INTERVENTION> <HARDWARE TESTER>

THE ROUTINE WILL CREATE THE M8036 CONTROL PROM. THE OPERATOR MAY CREATE SEVERAL COPIES BY REPEATELY EXECUTING THIS SECTION. THE PROGRAM WILL INFORM THE OPERATOR THE STEPS TO PERFORM.

LOCATION	OCTAL	BINARY
-----	-----	-----
00	215	10001101
01	244	10100100
02	116	01001110
03	144	01100100
04	215	10001101
05	244	10100100
06	116	01001110
07	144	01100100
10	215	10001101
11	244	10100100
12	116	01001110
13	144	01100100
14	215	10001101
15	244	10100100
16	116	01001110
17	144	01100100
20	106 (116)	01000110 (01001110)
21	144	01100100
22	314 (216)	11001100 (10001110)
23	304 (244)	11000100 (10100100)

NOTE: () INDICATES OLD CONTENTS OF M8036 ROM.

10.0 LISTING

13	BASIC DEFINITIONS
14	MEMORY MANAGEMENT DEFINITIONS
21	OPERATIONAL SWITCH SETTINGS
22	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
29	ACT11 HOOKS
31	APT PARAMETER BLOCK
32	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
147	INITIALIZE THE COMMON TAGS
149	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
150	PRIME THE NCV11 ADDRESSES FROM THE DEFAULT VALUES
179	DIAGNOSTIC IDENTIFICATION TYPEOUT
192	KEYBOARD COMMAND DECODER
345	
346	TEST
347	-----
348	
350	T1 VERIFY THE 8 NCV11 AND 2 NCV11 CLOCK BUS ADDRESSES RESPOND
381	T2 FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.
397	T3 VERIFY THAT "INIT" CLEARS THE CSR REGISTER
406	T4 VERIFY THAT "CLEAR ALL" CLEARS THE CSR REGISTER
416	T5 VERIFY LOW BYTE OPERATION OF THE "CSR" REGISTER
431	T6 FLOAT A 1 ACROSS 4 BITS OF THE SPECIAL FUNCTION REGISTER
445	T7 VERIFY THAT CLEARING HIGH BYTE OF SFR DOES NOT CLEAR LOW BYTE
455	T10 VERIFY THAT "INIT" CLEARS THE SFR REGISTER
463	T11 VERIFY THAT "CLEAR ALL" CLEARS THE SFR REGISTER
472	T12 FLOAT A 1 ACROSS THE WORD COUNT REGISTER
485	T13 FLOAT A 1 ACROSS THE BUS ADDRESS REGISTER
498	T14 FLOAT A 1 ACROSS THE OFFSET REGISTER
512	T15 VERIFY NO DUAL REGISTER SELECTION
548	T16 VERIFY "CLR ALL" CLEARS THE EXTENDED OFFSET BITS
557	T17 TEST THE "ACTIVE" FLOP CAN SET AND CLEAR
589	
592	T20 VERIFY Z INPUTS CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN MATRIX MODE
613	T21 VERIFY Z INPUTS CAUSE THE LOW 16 BITS OF 32 BIT COUNTER TO CHANGE
637	T22 VERIFY Z INPUTS CAUSE THE LOW 24 BITS OF 32 BIT COUNTER TO CHANGE
655	T23 VERIFY Z INPUTS CAUSE THE HIGH 8 BITS OF THE 32 BIT COUNTER TO CHANGE
669	T24 VERIFY Z INPUTS DO NOT CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN LIST MODE
680	T25 TEST THAT "WCA OVFL" SETS Z/WC OVERFLOW FLOP AND "CLR ALL" CLEARS IT
711	T26 TEST THAT "WCA OVFL" SETS Z/WC OVERFLOW FLOP AND "CLR WC OVFL" CLEARS IT
745	T27 TEST THAT "WCA OVFL" GENERATES AN INTERRUPT
791	T30 VERIFY "WCA OVFL" CLEARS "ACTIVE"
814	
817	T31 VERIFY JOYSTICK DONE FLOP SETS
836	T32 VERIFY THAT "RESET" INSTRUCTION CLEARS THE JOY READY FLOP
874	T33 JOYSTICK DATA PATH = GAIN =0 ZB ENABLE =0 RES. = 000
879	T34 JOYSTICK DATA PATH = GAIN =1
884	T35 JOYSTICK DATA PATH = ZB ENABLE =1
889	T36 JOYSTICK DATA PATH RES. = 001
894	T37 JOYSTICK DATA PATH RES. = 010
899	T40 JOYSTICK DATA PATH RES. = 100
905	
907	T41 VERIFY THE DATA INCREMENT FUNCTION

912	T42	VERIFY THE DATA INCREMENT CARRY BIT			
918	T43	VERIFY THE DATA INCREMENT FUNCTION IS INHIBITED			
924	T44	VERIFY THE DATA DECREMENT FUNCTION			
930	T45	VERIFY THE DATA DECREMENT BORROW			
935	T46	VERIFY THE DATA DECREMENT FUNCTION IS INHIBITED			
956	T47	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 0			
957	T50	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 1 ZB ENABLE = 0			
958	T51	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 1			
960	T52	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 0			
961	T53	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 1 ZB ENABLE = 0			
962	T54	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 1			
964	T55	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 0			
965	T56	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 1 ZB ENABLE = 0			
966	T57	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 1			
968	T60	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 0			
969	T61	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 1 ZB ENABLE = 0			
970	T62	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 1			
972	T63	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 0			
973	T64	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 1 ZB ENABLE = 0			
974	T65	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 1			
976	T66	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 0			
977	T67	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 1 ZB ENABLE = 0			
978	T70	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 1			
980	T71	TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 0			
981	T72	TEST ADDRESS MAKFR - MATRIX MODE - RES = 1 GAIN = 1 ZB ENABLE = 0			
982	T73	TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 1			
984	T74	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 0			
998	T75	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 1 GAIN = 0			
1012	T76	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 1			
1027					
1029	T77	ENABLE A ONE WORD TRANSFER SECTION LIST MODE			
1077	T100	ENABLE A 512 WORD TRANSFER SECTION LIST MODE			
1125	T101	VERIFY "TIMEOUT" FLOP SETS AND "CLR ALL" CLEARS IT			
1155	T102	VERIFY "TIMEOUT" FLOP SETS AND "CLR TIMEOUT" CLEARS IT			
1184	T103	VERIFY "TIMEOUT" INTERRUPT			
1217	T104	VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 16			
1238	T105	VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 17			
1262	T106	VERIFY "SET EVENT" DATA GENERATES A 177777 DATA WORD			
1283	T107	VERIFY "SET TIME" DATA GENERATES A 000000 DATA WORD			
1305	T110	VERIFY "CLOCK ST1" GENERATES A EVENT (177777) DATA WORD			
1326	T111	VERIFY "CLOCK OVERFLOW" GENERATES A TIME (000000) DATA WORD			
1350	T112	VERIFY "SET TIME" OVERRIDES "SET EVENT" DATA			
1370	T113	DO A ONE WORD MATRIX MODE TRANSFER -CHECK FOR INCREMENT FUNCTION			
1393	T114	VERIFY EACH BIT OF THE INCREMENT REG. DATA PATH			
1429	T115	CHECK FOR LOW BYTE "INC OVFL" TO SET CELL OVERFLOW AND "CLR CELL" TO CLEAR IT			
1472	T116	CHECK FOR WORD "INC OVFL" TO SET CELL OVERFLOW AND "CLR ALL" TO CLEAR IT			
1508	T117	CHECK FOR "CELL OVERFLOW" INTERRUPT			
1541	T120	VERIFY CORRECT BR LEVEL THE NCV-11			
1610					
1612	T121	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 13			
1613	T122	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 12			
1615	T123	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 11			
1616	T124	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 10			
1618	T125	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 09			
1619	T126	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 08			
1621	T127	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 07			

1622	T130	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 06
1624	T131	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 05
1625	T132	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 04
1627	T133	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 03
1628	T134	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 02
1631		
1633	T135	VERIFY THE ADM INPUT TO THE MATRIX MUX. USING GAIN ENABLE
1664	T136	VERIFY THE ADM INPUT TO THE MATRIX MUX. ADDER USING ZB ENABLE
1684	T137	VERIFY LOW BYTE OPERATION OF THE "TESTX" FLOP
1710	T140	VERIFY BIT 12 ADDER INPUT TO MATRIX MODE MUX
1726	T141	VERIFY BIT 13 ADDER INPUT TO MATRIX MODE MUX
1743	T142	VERIFY BIT 14 ADDER INPUT TO MATRIX MODE MUX
1765	T143	CHECK FOR HIGH BYTE "INC OVFL" TO SET CELL OVERFLOW
1795	T144	VERIFY EACH BIT OF THE LOWER 16 BIT Z COUNTER (TESTER ONLY)
1818	T145	VERIFY EACH BIT OF THE HIGHER 16 BIT Z COUNTER (TESTER ONLY)
1842	T146	VERIFY THAT CAMERA 01 CHANNEL IS OPERATIONAL (TESTER ONLY)
1862	T147	VERIFY THAT CAMERA 10 CHANNEL IS OPERATIONAL (TESTER ONLY)
1882	T150	VERIFY THAT CAMERA 11 CHANNEL IS OPERATIONAL (TESTER ONLY)
1902	T151	DYNAMIC MATRIX MODE ADDRESS
1974	T152	DYNAMIC LIST MODE ADDRESS
2042	T153	DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K
2084	T154	ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY
2132	T155	ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY
2179	T156	VERIFY BIT 15 MATRIX ADDER INPUT
2214	T157	VERIFY HIGH BYTE OPERATION OF THE "TEST X"
2237	T160	VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER
2274	T161	DETERMINE IF DIFLIN IS TO BE RUN (F)
2278		END OF PASS ROUTINE
2280		ERROR ASCII MESSAGES
2336		TTY INPUT ROUTINE
2338		READ AN OCTAL NUMBER FROM THE TTY
2340		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2341		SCOPE HANDLER ROUTINE
2342		TYPE ROUTINE
2344		BINARY TO OCTAL (ASCII) AND TYPE
2346		ERROR HANDLER ROUTINE
2348		ERROR MESSAGE TIMEOUT ROUTINE
2350		APT COMMUNICATIONS ROUTINE
2351		POWER DOWN AND UP ROUTINES
2353		ROUTINE TO SIZE MEMORY
2357		SAVE AND RESTORE R0-R5 ROUTINES
2359		INTEGER MULTIPLY ROUTINE
2361		INTEGER DIVIDE ROUTINE
2363		DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
2364		SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
2366		TRAP DECODER
(3)		TRAP TABLE
2372		A TO D FIELD SITE AND ADJUSTMENT LOOP

```
1      ;DEVELOPED USING SYSMAC.C3
2      .LIST ME
3      .NLIST MD,MC,CND
4      .ENABL ABS,AMA
5      $SWR=167400
6      $TN=1
12     .TITLE CZNCCC NCV11 DIAGNOSTIC
(1)    ;*COPYRIGHT (C) 1980
(1)    ;*DIGITAL EQUIPMENT CORP.
(1)    ;*MAYNARD, MASS. 01754
(1)    ;*
(1)    ;*PROGRAM BY R. SHOOP
(1)    ;*
(1)    ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)    ;*
13     .SBTTL BASIC DEFINITIONS
(1)    ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)    001100 STACK= 1100
(1)    .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)    .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
(1)    ;*MISCELLANEOUS DEFINITIONS
(1)    000011 HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)    000012 LF= 12      ;;CODE FOR LINE FEED
(1)    000015 CR= 15      ;;CODE FOR CARRIAGE RETURN
(1)    000200 CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1)    .EQUIV PS,PSW
(1)    177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)    177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1)    177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)    ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000 R0= %0      ;;GENERAL REGISTER
(1)    000001 R1= %1      ;;GENERAL REGISTER
(1)    000002 R2= %2      ;;GENERAL REGISTER
(1)    000003 R3= %3      ;;GENERAL REGISTER
(1)    000004 R4= %4      ;;GENERAL REGISTER
(1)    000005 R5= %5      ;;GENERAL REGISTER
(1)    000006 R6= %6      ;;GENERAL REGISTER
(1)    000007 R7= %7      ;;GENERAL REGISTER
(1)    000006 SP= %6      ;;STACK POINTER
(1)    000007 PC= %7      ;;PROGRAM COUNTER
(1)    ;*PRIORITY LEVEL DEFINITIONS
(1)    000000 PR0= 0      ;;PRIORITY LEVEL 0
(1)    000040 PR1= 40     ;;PRIORITY LEVEL 1
(1)    000100 PR2= 100    ;;PRIORITY LEVEL 2
(1)    000140 PR3= 140    ;;PRIORITY LEVEL 3
(1)    000200 PR4= 200    ;;PRIORITY LEVEL 4
(1)    000240 PR5= 240    ;;PRIORITY LEVEL 5
(1)    000300 PR6= 300    ;;PRIORITY LEVEL 6
(1)    000340 PR7= 340    ;;PRIORITY LEVEL 7
```



```
(1)
(1)
(1) 000004      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000010      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
(1) 000014      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014      TBITVEC=14        ;; "T" BIT
(1) 000014      TRTVEC= 14         ;; TRACE TRAP
(1) 000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
(1) 000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024      PWRVEC= 24         ;; POWER FAIL
(1) 000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
(1) 000034      TRAPVEC=34        ;; "TRAP" TRAP
(1) 000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
(1) 000064      TPVEC= 64          ;; TTY PRINTER VECTOR
(1) 000240      PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
  14          .SBTTL MEMORY MANAGEMENT DEFINITIONS
(1)
(1)          ;*KT11 VECTOR ADDRESS
(1)
(1) 000250      MMVEC= 250
(1)
(1)          ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1) 177572      SR0= 177572
(1) 177574      SR1= 177574
(1) 177576      SR2= 177576
(1) 172516      SR3= 172516
(1)
(1)          ;*USER "I" PAGE DESCRIPTOR REGISTERS
(1)
(1) 177600      UIPDR0= 177600
(1) 177602      UIPDR1= 177602
(1) 177604      UIPDR2= 177604
(1) 177606      UIPDR3= 177606
(1) 177610      UIPDR4= 177610
(1) 177612      UIPDR5= 177612
(1) 177614      UIPDR6= 177614
(1) 177616      UIPDR7= 177616
(1)
(1)          ;*USER "D" PAGE DESCRIPTOR REGISTORS
(1)
(1) 177620      UDPDR0= 177620
(1) 177622      UDPDR1= 177622
(1) 177624      UDPDR2= 177624
(1) 177626      UDPDR3= 177626
(1) 177630      UDPDR4= 177630
(1) 177632      UDPDR5= 177632
(1) 177634      UDPDR6= 177634
(1) 177636      UDPDR7= 177636
(1)
(1)          ;*USER "I" PAGE ADDRESS REGISTERS
(1)
(1) 177640      UIPAR0= 177640
(1) 177642      UIPAR1= 177642
(1) 177644      UIPAR2= 177644
(1) 177646      UIPAR3= 177646
(1) 177650      UIPAR4= 177650
```


(1)	177652	UIPAR5= 177652
(1)	177654	UIPAR6= 177654
(1)	177656	UIPAR7= 177656
(1)		
(1)		;*USER 'D' PAGE ADDRESS REGISTERS
(1)	177660	UDPAR0= 177660
(1)	177662	UDPAR1= 177662
(1)	177664	UDPAR2= 177664
(1)	177666	UDPAR3= 177666
(1)	177670	UDPAR4= 177670
(1)	177672	UDPAR5= 177672
(1)	177674	UDPAR6= 177674
(1)	177676	UDPAR7= 177676
(1)		
(1)		;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
(1)	172200	SIPDR0= 172200
(1)	172202	SIPDR1= 172202
(1)	172204	SIPDR2= 172204
(1)	172206	SIPDR3= 172206
(1)	172210	SIPDR4= 172210
(1)	172212	SIPDR5= 172212
(1)	172214	SIPDR6= 172214
(1)	172216	SIPDR7= 172216
(1)		
(1)		;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
(1)	172220	SDPDR0= 172220
(1)	172222	SDPDR1= 172222
(1)	172224	SDPDR2= 172224
(1)	172226	SDPDR3= 172226
(1)	172230	SDPDR4= 172230
(1)	172232	SDPDR5= 172232
(1)	172234	SDPDR6= 172234
(1)	172236	SDPDR7= 172236
(1)		
(1)		;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
(1)	172240	SIPAR0= 172240
(1)	172242	SIPAR1= 172242
(1)	172244	SIPAR2= 172244
(1)	172246	SIPAR3= 172246
(1)	172250	SIPAR4= 172250
(1)	172252	SIPAR5= 172252
(1)	172254	SIPAR6= 172254
(1)	172256	SIPAR7= 172256
(1)		
(1)		;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
(1)	172260	SDPAR0= 172260
(1)	172262	SDPAR1= 172262
(1)	172264	SDPAR2= 172264
(1)	172266	SDPAR3= 172266
(1)	172270	SDPAR4= 172270
(1)	172272	SDPAR5= 172272

```
(1)          172274          SDPAR6= 172274
(1)          172276          SDPAR7= 172276
(1)
(1)          ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
(1)
(1)          172300          KIPDR0= 172300
(1)          172302          KIPDR1= 172302
(1)          172304          KIPDR2= 172304
(1)          172306          KIPDR3= 172306
(1)          172310          KIPDR4= 172310
(1)          172312          KIPDR5= 172312
(1)          172314          KIPDR6= 172314
(1)          172316          KIPDR7= 172316
(1)
(1)          ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
(1)
(1)          172320          KDPDR0= 172320
(1)          172322          KDPDR1= 172322
(1)          172324          KDPDR2= 172324
(1)          172326          KDPDR3= 172326
(1)          172330          KDPDR4= 172330
(1)          172332          KDPDR5= 172332
(1)          172334          KDPDR6= 172334
(1)          172336          KDPDR7= 172336
(1)
(1)          ;*KERNEL 'I' PAGE ADDRESS REGISTERS
(1)
(1)          172340          KIPAR0= 172340
(1)          172342          KIPAR1= 172342
(1)          172344          KIPAR2= 172344
(1)          172346          KIPAR3= 172346
(1)          172350          KIPAR4= 172350
(1)          172352          KIPAR5= 172352
(1)          172354          KIPAR6= 172354
(1)          172356          KIPAR7= 172356
(1)
(1)          ;*KERNEL 'D' PAGE ADDRESS REGISTERS
(1)
(1)          172360          KDPAR0= 172360
(1)          172362          KDPAR1= 172362
(1)          172364          KDPAR2= 172364
(1)          172366          KDPAR3= 172366
(1)          172370          KDPAR4= 172370
(1)          172372          KDPAR5= 172372
(1)          172374          KDPAR6= 172374
(1)          172376          KDPAR7= 172376
(1)
15
16          172760          ABASE=172760          ;NCV11 BASE ADDRESS
17          140370          AVECT1=140370        ;BR LEVEL 6 VECTOR TO 370
18          170420          ACDW1=170420        ;NCV11 CLOCK ADDRESS
```

```

20
21      .SBTTL  OPERATIONAL SWITCH SETTINGS
(1)      : *
(1)      : *      SWITCH              USE
(1)      : *      -----
(1)      : *      15              HALT ON ERROR
(1)      : *      14              LOOP ON TEST
(1)      : *      13              INHIBIT ERROR TYPEOUTS
(1)      : *      11              INHIBIT ITERATIONS
(1)      : *      10              BELL ON ERROR
(1)      : *      9              LOOP ON ERROR
(1)      : *      8              LOOP ON TEST IN SWR<7:0>
22      .SBTTL  TRAP CATCHER
(1)
(1)      000000      .=0
(1)      : *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)      : *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)      : *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)      000174      .=174
(1) 000174 000000  DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000  SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
(1)      .SBTTL  STARTING ADDRESS(ES)
(1) 000200 000137 002034  JMP      @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
23 000204 000137 002014  JMP      RTRT   ;RESTART ADDRESS
24 000210 000137 002022  JMP      TESTER ;STARTING ADDRESS WHEN IN THE OPTION AREA
25
26      000100      .=100
27 000100 000104 000200 000002  104,200,2      ;B EVENT SAFE GUARD
  
```

```
29      .SBTTL ACT11 HOOKS
(1)
(2)      ;*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      $SVPC=.          ;SAVE PC
(1)      .=46
(1)      000046 030170  $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)      000052 000052  .=52
(1)      000052 000000  .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      000106 000106  .=$SVPC        ;; RESTORE PC
30      .=1000
31      .SBTTL APT PARAMETER BLOCK
(1)
(2)      ;*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ;*****
(1)      001000 001000  .$X=.          ;;SAVE CURRENT LOCATION
(1)      000024 000024  .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)      000024 000200  200          ;;FOR APT START UP
(1)      000044 000044  .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)      000044 001000  $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)      001000 001000  .=$X          ;;RESET LOCATION COUNTER
(2)      ;*****
(1)      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      ;INTERFACE SPEC.
(1)
(1)      $APTHD:
(1)      001000 000000  $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1)      001002 001174  $MBADR: .WORD $MAIL     ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1)      001004 000030  $STMT: .WORD 30        ;;RUN TIM OF LONGEST TEST
(1)      001006 000010  $PASTM: .WORD 10       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1)      001010 000030  $UNITM: .WORD 30       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1)      001012 000031  .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```



```
(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 140370 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001246 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001250 172760 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001252 000000 $DEVM: .WORD ADEVM ;;DEVICE MAP
(2) 001254 170420 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
(2) 001256 $TEND:
.MEXIT
```

.SBTTL ERROR POINTER TABLE

:(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) : * 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

\$ERRTB:

33	001256	030224	032504	032734	EM1,DH2,DT2,DF0	:M8026 NCV11 TIMEOUT
34	001264	030264	032504	032734	EM2,DH2,DT2,DF0	:M8026 COMMAND-STATUS REGISTER ERROR
35	001274	030330	032504	032734	EM3,DH2,DT2,DF0	:M8026 SPECIAL FUNCTION REGISTER ERROR
36	001304	030376	032504	032734	EM4,DH2,DT2,DF0	:M8026 WORD COUNT REGISTER ERROR
37	001314	030436	032504	032734	EM5,DH2,DT2,DF0	:M8026 BUS ADDRESS REGISTER ERROR
38	001324	030477	032504	032734	EM6,DH2,DT2,DF0	:M8026 OFFSET REGISTER ERROR
39	001334	030533	032504	032734	EM7,DH2,DT2,DF0	:M8026 DUAL REGISTER SELECTION ERROR
40	001344	030577	032504	032734	EM10,DH2,DT2,DF0	:M8026-M8036 LOW 16 BIT Z COUNT ERROR
41	001354	030644	032504	032734	EM11,DH2,DT2,DF0	:M8026-M8036 HIGH 16 BIT Z COUNT ERROR
42	001364	030712	032504	032734	EM12,DH2,DT2,DF0	:M8026 Z COUNT STATUS ERROR
43	001374	030745	032504	032734	EM13,DH2,DT2,DF0	:M8026 Z COUNT INTERRUPT ERROR
44	001404	031003	032504	032734	EM14,DH2,DT2,DF0	:M8036 JOYSTICK STATUS ERROR
45	001414	031037	032504	032734	EM15,DH2,DT2,DF0	:M8036 JOYSTICK DATA ERROR
46	001424	031071	032504	032734	EM16,DH2,DT2,DF0	:M8036 DATA INCREMENT ERROR
47	001434	031124	032504	032734	EM17,DH2,DT2,DF0	:M8036 DATA DECREMENT ERROR
48	001444	031157	032504	032734	EM20,DH2,DT2,DF0	:M8026-M8036 MATRIX MODE ADDRESS MAKER DATA ERROR
49	001454	031240	032504	032734	EM21,DH2,DT2,DF0	:M8026 LIST MODE ADDRESS MAKER DATA ERROR
50	001464	031311	032504	032734	EM22,DH2,DT2,DF0	:M8026 LIST MODE TRANSFER BUS ADDRESS ERROR
51	001474	031364	032504	032734	EM23,DH2,DT2,DF0	:M8026 LIST MODE TRANSFER WORD COUNT ERROR
52	001504	031436	032504	032734	EM24,DH2,DT2,DF0	:M8026 LIST MODE TRANSFER OFFSET ERROR
53	001514	031504	032504	032734	EM25,DH2,DT2,DF0	:M8026 TIMEOUT STATUS ERROR

54	001524	033006					
	001526	031537	032504	032734	EM26,DH2,DT2,DF0	:M8026	TIMEOUT INTERRUPT ERROR
	001534	033006					
55	001536	031575	032504	032734	EM27,DH2,DT2,DF0	:M8026	SET 'EVENT' OR 'TIME' DATA ERROR
	001544	033006					
56	001546	031644	032504	032734	EM30,DH2,DT2,DF0	:M8026	CELL INCREMENT DATA ERROR
	001554	033006					
57	001556	031704	032504	032734	EM31,DH2,DT2,DF0	:M8026	CELL OVERFLOW STATUS ERROR
	001564	033006					
58	001566	031745	032504	032734	EM32,DH2,DT2,DF0	:M8026	CELL OVERFLOW INTERRUPT ERROR
	001574	033006					
59	001576	032011	032530	032746	EM33,DH3,DT3,DF0	:M8026	MATRIX MODE ADDRESS MUX ERROR
	001604	033006					
60	001606	032055	032504	032734	EM34,DH2,DT2,DF0	:M8026	'TESTX' FUNCTION ERROR
	001614	033006					
61	001616	032112	032563	032762	EM35,DH4,DT4,DF0	:M8026	DATA ERROR WHEN TRANSFERING TO EXTENDED MEMORY
	001624	033006					
62	001626	032177	032504	032734	EM36,DH2,DT2,DF0	:M8026	LIST MODE TRANSFER STATUS ERROR
	001634	033006					
63	001636	032245	032530	032746	EM37,DH3,DT3,DF0	:M8026	LIST MODE TRANSFER DATA ERROR
	001644	033006					
64	001646	032311	032504	032734	EM40,DH2,DT2,DF0	:JUMPER-M8026-M7952	'EVENT' OR 'TIME' MARK ERROR
	001654	033006					
65	001656	032371	032471	033000	EM41,DH1,DT5,DF0	:M7952	CLOCK TIMEOUT
	001664	033006					
66	001666	032431	032471	032726	EM42,DH1,DT1,DF0	:M8217	INTERRUPT LEVEL ERROR
	001674	033006					

:DL11 BLASTER COMM. ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)

DLICSR: 176510 ;INPUT STATUS REG.
 DLIBD: 176512 ;INPUT DATA
 DLOCSR: 176514 ;OUTPUT STATUS
 DLODB: 176516 ;OUTPUT DATA

:VSV01 ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)

VTCHAR: 172600 ;CHAR. STATUS
 VTYPOS: 172602
 VTXPOS: 172603
 VTCXY: 172604
 VTCSR: 172620 ;MAP STATUS
 VTMAP: 172622
 VTPX: 172624
 VTINT: 172630

:TESTER ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)

DACSR: 176416 ;KNOWN GOOD D/A STATUS
 DAC0: 176420
 DAC1: 176422
 DAC2: 176424
 DAC3: 176426

67							
68	001676	176510					
69	001700	176512					
70	001702	176514					
71	001704	176516					
72							
73							
74	001706	172600					
75	001710	172602					
76	001712	172603					
77	001714	172604					
78	001716	172620					
79	001720	172622					
80	001722	172624					
81	001724	172630					
82							
83							
84	001726	176416					
85	001730	176420					
86	001732	176422					
87	001734	176424					
88	001736	176426					


```
90 ;KVV11 PROGRAM GENERATED ADDRESSES (THE OPER. ONLY HAS TO CHANGE '$CDW1')
91
92 001740 170420 KWCSR: ACDW1 ;KVV11 STATUS REGISTER
93 001742 170421 KWCSR1: ACDW1+1 ;KVV11 STATUS REG. HIGH BYTE
94 001744 170422 KWPSR: ACDW1+2 ;KVV11 PRESET REGISTER
95
96 ;NCV11 PROGRAM GENERATER ADDRESSES (THE OPER. ONLY HAS TO CHANGE '$BASE')
97
98 001746 172760 CSR: ABASE
99 001750 172762 OFF: ABASE+2
100 001752 172764 WCR: ABASE+4
101 001754 172766 BAR: ABASE+6
102 001756 172770 SFR: ABASE+10
103 001760 172772 ADM: ABASE+12
104 001762 172774 JOY: ABASE+14
105 001764 172776 BAR1: ABASE+16
106 001766 172761 CSRHB: ABASE+1
107 001770 172771 SFRHB: ABASE+11
108
109 ;NCV11 PROGRAM GENERATED VECTORS (THE OPER. ONLY HAS TO CHANGE LOW 9 BITS OF '$VECT1')
110
111 001772 140370 VECTA0: AVECT1
112 001774 140372 VECTA1: AVECT1+2
113 001776 140374 VECTB0: AVECT1+4
114 002000 140376 VECTB1: AVECT1+6
115
116 ;NCV11 PROGRAM GENERATED BR LEVEL (THE OPER. ONLY HAS TO CHANGE TOP 3 BITS OF '$VECT1')
117
118 002002 000300 BRLEV: 300 ;BR LEVEL 6
119
120 002004 000000 $TEMP: 0
121 002006 000001 CPUDLO: 1
122 002010 000020 CPUDL1: 20
123 002012 000004 CPUDL2: 4
124
125 040000 CLRWCO=BIT14
126 004000 CLRALL=BIT11
127 000400 ENDDMA=BIT8
128 000010 TSTDMA=BIT3
129 000004 TSTCON=BIT2
130 000002 TESTZ=BIT1
131 000001 REDJOY=BIT0
132 000000 MORTST=0 ;DEFINE TO RESTORE TEST CODE
133
```

```

140 002014 005237 050034 RTRT: INC AGAN ;SET RESTART FLAG
141 002020 000411 BR RTRTA
142 002022 005237 050026 TESTER: INC WFMODE ;SET FLAG INDICATING OPTION AREA MODE
143 002026 005037 050034 CLR AGAN
144 002032 000404 BR RTRTA
145 002034 005037 050034 BEGIN: CLR AGAN
146 002040 005037 050026 CLR WFMODE ;CLEAR FLAG INDIC. OPTION TEST AREA MODE
147 002044 RTRTA:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
(1) 002044 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 002050 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 002052 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 002056 001374 BNE -6 ;;LOOP BACK IF NO
(1) 002060 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 002064 012737 034112 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002072 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
(1) 002100 012737 035106 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002106 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
(1) 002114 012737 037272 000034 MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002122 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
(1) 002130 012737 035710 000024 MOV #PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
(1) 002136 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
(1) 002144 013737 030136 030130 MOV SENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 002152 005037 001160 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002156 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002162 112737 000001 001115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
(1) 002170 012737 002170 001106 MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002176 012737 002176 001110 MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
(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) 002204 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002210 012737 002244 000004 MOV #64$,@ERRVEC ;;SET UP ERROR VECTOR
(2) 002216 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002224 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002232 022777 177777 176700 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 002240 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT = -1
(2) 002242 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 002244 012716 002252 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
(2) 002250 000002 RTI
(2) 002252 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
(2) 002260 012737 000174 001142 MOV #DISPREG,DISPLAY
(2) 002266 012637 000004 66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002272 005037 001202 CLR $PASS ;;CLEAR PASS COUNT
(2) 002276 132737 000200 001215 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002304 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
(2) 002306 012737 001216 001140 MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 002314 67$:

```



```

179          .SBTTL  DIAGNOSTIC IDENTIFICATION TYPEOUT
180 002602 005737 050034      TST      AGAN          ;TEST IF RESTART
181 002606 001017          BNE      RBEGO        ;BR IF YES
182 002610 105737 001134      TSTB    $AUTOB       ;TEST IF ACT/APT
183 002614 001405          BEQ      3$           ;BR IF NOT
184 002616 012737 177777 050024  MOV     #-1,RUNDIF    ;INDICATE TO RUN DIFLIN IN CHAIN MODE
185 002624 000137 003360          JMP     LOGIC         ;YES; RUN LOGIC AND DIFLIN TESTS
186 002630 005737 050026      3$:    TST      WFMODE    ;TEST IF OPTIC% CHECKOUT MODE
187 002634 001402          BEQ      4$           ;BR IF YES
188 002636 104401          TYPE
189 002640 046174          LISTO          ;TELL OPERATOR ABOUT OPTION AREA TESTS
190 002642 104401      4$:    TYPE
191 002644 046404          LIST          ;TELL OPERATOR ABOUT THE TEST
192          .SBTTL  KEYBOARD COMMAND DECODER
193 002646 104401 047017      RBEGO:  TYPE, LIST1    ;PRINT 'DOT'
194 002652 104411      1$:    RDLIN          ;READ THE RESPONSE
195 002654 013637 052412      MOV     @ (SP)+,RUNIT ;GET THE CHARACTER
196 002660 142737 000040 052412  BICB   #40,RUNIT     ;ENSURE UPPER CASE
197 002666 112737 000042 052413  MOVB   #' ',RUNIT+1  ;FIX ASCII MESSAGE
198 002674 122737 000101 052412  CMPB   #'A',RUNIT    ;DETERMINE IF AN A
199 002702 001002          BNE     2$           ;BR IF NOT
200 002704 000137 003300          JMP     TFSITE       ;RUN SITE ADJUSTMENT LOOP
201 002710 005737 050026      2$:    TST      WFMODE    ;TEST IF ON THE TESTER
202 002714 001406          BEQ     3$           ;BR IF NOT
203 002716 122737 000102 052412  CMPB   #'B',RUNIT    ;DETERMINE IF AN B
204 002724 001002          BNE     3$           ;BR IF NOT
205 002726 000137 042274          JMP     BLAST        ;RUN THE BLASTER SECTION
206 002732 122737 000104 052412  3$:    CMPB   #'D',RUNIT ;DETERMINE IF AN D
207 002740 001004          BNE     4$           ;BR IF NOT
208 002742 005037 050024          CLR     RUNDIF      ;INDICATE DIFLIN IS NOT TO BE RUN
209 002746 000137 043550          JMP     DIFLIN      ;AND RUN IT
210 002752 122737 000106 052412  4$:    CMPB   #'F',RUNIT ;DETERMINE IF AN F
211 002760 001005          BNE     5$           ;BR IF NOT
212 002762 012737 177777 050024  MOV     #-1,RUNDIF    ;INDICATE RUN DIFLIN AFTER THE LOGIC TEST
213 002770 000137 003360          JMP     LOGIC        ;AND RUN LOGIC TEST
214 002774 005737 050026      5$:    TST      WFMODE    ;TEST IF OPTION CHECKOUT MODE
215 003000 001406          BEQ     6$           ;BR IF NOT
216 003002 122737 000111 052412  CMPB   #'I',RUNIT    ;DETERMIN IF AN I
217 003010 001002          BNE     6$           ;BR IF NOT
218 003012 000137 047024          JMP     POTIME       ;RUN IN-HOUSE ADJUSTMENT LOOP
219 003016 122737 000114 052412  6$:    CMPB   #'L',RUNIT ;DETERMINE IF AN L
220 003024 001004          BNE     7$           ;BR IF NOT
221 003026 005037 050024          CLR     RUNDIF      ;INDICATE NO DIFLIN
222 003032 000137 003360          JMP     LOGIC
223 003036 122737 000110 052412  7$:    CMPB   #'H',RUNIT ;DETERMINE IF AN H
224 003044 001002          BNE     10$          ;BR IF NOT
225 003046 000137 002034          JMP     BEGIN        ;START OVER
226 003052 122737 000037 052412  10$:   CMPB   #37,RUNIT    ;DETERMINE IF AN '?'
227 003060 001002          BNE     11$          ;BR IF NOT
228 003062 000137 002034          JMP     BEGIN        ;START OVER
229 003066 005737 050026      11$:   TST      WFMODE    ;TEST IF OPTION CHECKOUT MODE
230 003072 001406          BEQ     12$          ;BR IF NOT
231 003074 122737 000124 052412  CMPB   #'T',RUNIT    ;DETERMINE IF AN T
232 003102 001002          BNE     12$          ;BR IF NOT
233 003104 000137 042022          JMP     BTALK        ;KEYBOARD LOOP WITH BLASTER
234 003110 005737 050026      12$:   TST      WFMODE    ;TEST IF OPTION CHECKOUT MODE

```

235	003114	001406				BEQ	13\$:BR IF NOT
236	003116	122737	000103	052412		CMPB	#'C,RUNIT	:DETERMINE IF AN C
237	003124	001002				BNE	13\$:BR IF NOT
238	003126	000137	042756			JMP	PBLAST	:RUN M8036 PROM BLASTING CODE
239	003132	122737	000117	052412	13\$:	CMPB	#'O,RUNIT	:DETERMINE IF AN O
240	003140	001002				BNE	14\$:BR IF NOT
241	003142	000137	003242			JMP	XTTY	:CHANGE THE TTY ADDRESS
242	003146	122737	000123	052412	14\$:	CMPB	#'S,RUNIT	:DETERMINE IF AN S
243	003154	001002				BNE	15\$:BR IF NOT
244	003156	104406				GTSWR		:GET SWITCH REGISTER VALUE
245	003160	000632				BR	RBEGO	:RESTART
246	003162	122737	000115	052412	15\$:	CMPB	#'M,RUNIT	:DETERMINE IF AN M
247	003170	001005				BNE	FATFNG	:BR IF NOT
248	003172	012737	000377	050024		MOV	#377,RUNDIF	:INDICATE DIFLIN WITH EXPAND REPORT
249	003200	000137	043550			JMP	DIFLIN	:RUN DIF LIN

```

251 ;THE OPERATOR SELECTED THE WRONG THING - THEY SOMETIME HAVE "FAT FINGERS"
252 003204 104401 FATFNG: TYPE
253 003206 001170 $QUE C ;TYPE QUESTION MARK
254 003210 000616 BR RBEGO
255
256
257 ;RETURN TO HERE UPON UNEXPECTED BUS TRAP
258 003212 104401 052342 BUSTRP: TYPE, FATALO ;REPORT FATAL TRAP TO THE OPERATOR
259 003216 011646 MOV (SP),-(SP) ;GET TRAP ADDRESS
260 003220 104402 TYPOC
261 003222 005777 175712 1$: TST @SWR ;TEST IF HALT ON ERROR
262 003226 100001 BPL 2$ ;BR IF SET
263 003230 000000 HALT ;FATAL BUS TRAP DETECTED
264 003232 104401 052432 2$: TYPE, RUNITA
265 003236 000137 002014 JMP RTRT ;RESTART
266
267 ;ROUTINE TO INPUT NEW TTY ADDRESS FROM THE OPERATOR
268 003242 104401 052522 XTTY: TYPE, WARNO ;TELL THE OPERATOR THE RULES
269 003246 104412 RDOCT ;LISTEN TO THE OPERATOR
270 003250 012600 MOV (SP)+,RO ;READ THEIR INPUT
271 003252 001754 BEQ FATFNG ;BR IF JUST "CR"
272 003254 005710 TST (RO) ;ADDRESS THE DEVICE AND CHECK BUS TRAP
273 ;IF NO BUS TRAP - THEY MUST KNOW WHAT THEY ARE DOING !
274 003256 012701 001144 MOV #TKS,R1 ;GET POINTER
275 003262 010021 1$: MOV RO,(R1)+ ;LOAD THE VALUE
276 003264 005720 TST (RO)+ ;BUMP VALUE
277 003266 020127 001154 CMP R1,#TKS+10 ;TEST IF DONE ALL ADDRESSES
278 003272 001373 BNE 1$ ;BR IF NOT
279 003274 000137 002044 JMP RTRTA ;RETYPE THE HEADER AGAIN ON THE NEW TERMINAL
280
281 ;OPERATOR CHOSE THE FIELD ADJUSTMENT LOOP
282 003300 104401 051166 TFSITE: TYPE, PRIM6 ;TELL OPER. ABOUT A017
283 003304 104401 041344 TYPE, FIELDI ;ASK THE OPERATOR FOR INPUT Z MODE
284 003310 005037 040360 CLR INOUTZ ;DEFAULT TO USER SUPPLIED Z PULSES
285 003314 104411 RDLIN ;READ OPER. INPUT
286 003316 013637 003352 MOV @(SP)+,10$ ;GET CHAR
287 003322 142737 000040 003352 BICB #40,10$ ;ENSURE UPPER CASE
288 003330 122737 000131 003352 CMPB #'Y,10$ ;TEST FOR 'Y' INPUT
289 003336 001003 BNE 1$ ;BR IF NOT 'Y'
290 003340 012737 000002 040360 MOV #2,INOUTZ ;SET MAINT Z CONSTANT
291 003346 000137 037360 1$: JMP FSITE ;NOW DO THE LOOP
292 003352 000000 10$: 0
293
294 003354 000000 ADNOKT: 0 ;LAST ADDRESS WITHOUT M.M. OR XXDP
295 003356 000000 NLSI11: 0 ;NON-ZERO INDICATES LSI-11 PROCESSOR

```

```

297
298 003360 000005          LOGIC:  RESET
299 003362 005737 001202      TST      $PASS          ;TEST IF FIRST PASS
300 003366 001124          BNE      TST1           ;:BR IF NOT
301 003370 005737 050026      TST      WFMODE        ;CHECK IF ON TESTER
302 003374 001402          BEQ      5$             ;BR IF NOT
303 003376 104401 052250      TYPE,   PRIM4         ;TELL OPERATOR CABLE MUST BE CONNECTED
304 003402 005037 036124      5$:    CLR      $KT11    ;INDICATE NO KT11
305 003406 004737 036066      JSR     PC,$SIZE      ;SIZE BASIC MEMORY SIZE
306 003412 162737 001000 036404 SUB     #1000,$LSTAD   ;PROTECT ABSLDR
307 003420 005737 000042      TST     @#42         ;TEST IF XXDP CHAIN MODE ?
308 003424 001403          BEQ      1$             ;BR IF NOT CHAIN MODE
309 003426 162737 006200 036404 SUB     #3200.,$LSTAD  ;LEAVE ROOM FOR XXDP MONITOR
310 003434 013737 036404 003354 1$:    MOV     $LSTAD,ADNOKT ;SAVE BASIC MEMORY STAGE
311 003442 012737 000200 036124 MOV     #BIT7,$KT11   ;WITH M.M.
312 003450 004737 036066      JSR     PC,$SIZE      ;DETERMINE THE AMOUNT OF MEMORY
313 003454 005737 036124      TST     $KT11        ;TEST IF KT11 IS HERE
314 003460 100025          BPL     2$             ;BR IF NOT
315 003462 012737 000200 172342 MOV     #200,@#KIPAR1 ;
316 003470 012737 077406 172302 MOV     #77406,@#KIPDR1
317 003476 012737 000400 172344 MOV     #400,@#KIPAR2
318 003504 012737 077406 172304 MOV     #77406,@#KIPDR2
319 003512 012737 077406 172306 MOV     #77406,@#KIPDR3 ;4K, UP R/W FOR 60000
320 003520 012737 007600 172356 MOV     #7600,@#KIPAR7 ;I/O PAGE MAP
321 003526 012737 077406 172316 MOV     #77406,@#KIPDR7 ;4K, UP, R/W FOR I/O PAGE ACCESS
322 003534 105737 001134      2$:    TSTB    $AUTOB    ;TEST IF AUTO MODE
323 003540 001016          BNE     4$             ;:BR IF AUTO MODE
324 003542 104401 052463      TYPE,   MSGMEM        ;TELL OPERATOR THE AMOUNT OF MEMORY
325 003546 013700 036406      MOV     $LSTBK,RO    ;GET LAST GOOD BANK
326 003552 006200          ASR     RO
327 003554 006200          ASR     RO
328 003556 006200          ASR     RO
329 003560 006200          ASR     RO
330 003562 006200          ASR     RO
331 003564 005200          INC     RO
332 003566 010046          MOV     RO,-(SP)      ;CONVERT
333 003570 104405          TYPDS   ;TYPE OUT DEC. VALUE
334 003572 104401 052503      TYPE,   MSGK          ;AND THEN TYPE OUT THE REMAINDER OF MEMORY MESSAGE
335 003576 013737 000004 002004 4$:    MOV     @#ERRVEC,$TEMP ;SAVE LOC 4 VALUE
336 003604 012737 003632 000004 MOV     #3$,@#ERRVEC  ;LOAD LOC 4
337 003612 005037 003356      CLR     NLSI11       ;INDICATE NOT A LSI-11 CPU
338 003616 005037 177776      CLR     PSW          ;SHOULD FAIL IF LSI-11
339 003622 013737 002004 000004 MOV     $TEMP,@#ERRVEC ;RESTORE LOC 4 VALUE
340 003630 000403          BR      TST1         ;:BR IF TEST
341 003632 005237 003356      3$:    INC     NLSI11    ;SET AN LSI-11 FLAG
342 003636 000002          RTI                    ;RETURN

```

```

350      ;*****
(3)      ;*TEST 1          VERIFY THE 8 NCV11 AND 2 NCV11 CLOCK BUS ADDRESSES RESPOND
(3)      ;*****
(2) 003640 000004      TST1:  SCOPE
(1) 003642 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
351 003650 013737 000004 004016      MOV      @#ERRVEC,10$      ;SAVE BUS TRAP VECTOR
352 003656 012737 003754 000004      MOV      #1$,@#ERRVEC      ;LOAD BUS TRAP VECTOR
353 003664 005777 176056      TST      @CSR      ;ADDRESS
354 003670 005777 176054      TST      @OFF      ;
355 003674 005777 176052      TST      @WCR      ;THE
356 003700 005777 176050      TST      @BAR      ;
357 003704 005777 176046      TST      @SFR      ;NCV11
358 003710 005777 176044      TST      @ADM      ;
359 003714 005777 176044      TST      @BAR1      ;ADDRESSES
360 003720 012737 003762 000004      MOV      #2$,@#ERRVEC      ;LOAD NEW RETURN
361 003726 005777 176006      TST      @KWCSR      ;TEST CLOCK
362 003732 005777 176006      TST      @KWPSR      ;ADDRESSES
363      ;TEMP FIX THE DIAG TO NOT USE CLOCK JUMPERS
364 003736 012737 000001 004020      MOV      #1,DEADKW      ;INDICATE NCV11 CLOCK IS NOT THERE
365 003744 013737 004016 000004      MOV      10$,@#ERRVEC      ;RESTORE THE BUS TRAP VECTOR
366 003752 000423      BR      TST2      ;;BR AND TEST THE NCV11
367 003754 022626      1$:  CMP      (SP)+,(SP)+      ;CLEAN THE STACK
368 003756 104001      ERROR  1      ;BUS TRAP WHEN REFERENCING THE NCV11
369 003760 000411      BR      3$
370 003762 022626      2$:  CMP      (SP)+,(SP)+      ;CLEAN THE STACK
371 003764 104041      ERROR  41      ;BUS TRAP WHEN REFERENCING THE NCV11 CLOCK
372 003766 012737 000001 004020      MOV      #1,DEADKW      ;INDICATE NCV11 CLOCK IS NOT THERE
373 003774 013737 004016 000004      MOV      10$,@#ERRVEC      ;RESTORE LOC 4
374 004002 000407      BR      TST2      ;;
375 004004 013737 004016 000004      3$:  MOV      10$,@#ERRVEC      ;RESTORE BUS TRAP VECTOR
376 004012 000137 030102      JMP      $EOP      ;EXIT
377 004016 000000      10$:  0
378 004020 000000      DEADKW: 0      ;NON-ZERO SAYS NO NCV11 CLOCK

```



```

381
(3)
(3)
(2) 004022 000004
(1) 004024 012737 000100 001160
382 004032 012737 004046 001110
383 004040 012737 004000 002004
384
385 004046 013777 002004 175672
386 004054 017737 175666 001126
387 004062 013737 002004 001124
388 004070 052737 000200 001124
389 004076 023737 001124 001126
390 004104 001401
391 004106 104002
392
393 004110 006237 002004
394 004114 022737 000001 002004
395 004122 001351
396
397
(3)
(3)
(2) 004124 000004
(1) 004126 012737 000010 001160
398 004134 012777 007776 175604
399 004142 000005
400 004144 012737 000200 001124
401 004152 017737 175570 001126
402 004160 023737 001124 001126
403 004166 001401
404 004170 104002
405
406
(3)
(3)
(2) 004172 000004
(1) 004174 012737 000010 001160
407 004202 012777 007776 175536
408 004210 012777 004000 175540
409 004216 012737 000200 001124
410 004224 017737 175516 001126
411 004232 023737 001124 001126
412 004240 001401
413 004242 104002
414

```

```

*****
:*TEST 2      FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.
*****
TST2:  SCOPE
      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
      MOV      #1$, $LPERR      ;LOAD LOOP ADDRESS
      MOV      #BIT11,$STEMP    ;LOAD INITIAL REG. VALUE
1$:    MOV      $STEMP,@CSR      ;LOAD CSR REG.
      MOV      @CSR,$BDDAT      ;READ CSR
      MOV      $STEMP,$GDDAT    ;LOAD EXPECTED
      BIS      #BIT7,$GDDAT     ;FUDGE THE 'READY' BIT
      CMP      $GDDAT,$BDDAT    ;COMPARE THE VALUES
      BEQ      2$               ;;BR IF SAME
      ERROR   2                 ;UNEXPECTED VALUE IN THE CSR REGISTER
2$:    ASR      $STEMP          ;TRY THE NEXT DATA BIT
      CMP      #1,$STEMP        ;TEST IF NOW BIT 0
      BNE     1$               ;BR IF NOT
*****
:*TEST 3      VERIFY THAT "INIT" CLEARS THE CSR REGISTER
*****
TST3:  SCOPE
      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
      MOV      #7776,@CSR      ;LOAD CSR REG.
      RESET                       ;INITILIZE THE REGISTER
      MOV      #BIT7,$GDDAT    ;LOAD EXPECTED VALUE
      MOV      @CSR,$BDDAT     ;READ CSR REG.
      CMP      $GDDAT,$BDDAT   ;COMPARE THE VALUES
      BEQ      TST4            ;;BR IF EQUAL
      ERROR   2                 ;"BUS INIT" FAILED TO CLEAR CSR REG.
*****
:*TEST 4      VERIFY THAT "CLEAR ALL" CLEARS THE CSR REGISTER
*****
TST4:  SCOPE
      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
      MOV      #7776,@CSR      ;LOAD CSR REG.
      MOV      #CLRALL,@SFR     ;GENERATE "CLR ALL L"
      MOV      #BIT7,$GDDAT    ;LOAD EXPECTED VALUE
      MOV      @CSR,$BDDAT     ;READ THE CSR REG.
      CMP      $GDDAT,$BDDAT   ;COMPARE VALUES
      BEQ      TST5            ;;BR IF SAME
      ERROR   2                 ;"CLR ALL L" FAILED TO CLEAR CSR REG.

```

```
416
(3)
(3)
(2) 004244 000004
(1) 004246 012737 000100 001160
417 004254 012777 003636 175464
418 004262 012737 003600 001124
419 004270 105077 175452
420 004274 017737 175446 001126
421 004302 023737 001124 001126
422 004310 001401
423 004312 104002
424 004314 012777 003636 175424
425 004322 012737 000236 001124
426 004330 105077 175432
427 004334 017737 175406 001126
428 004342 023737 001124 001126
429 004350 001401
430 004352 104002
431
(3)
(3)
(2) 004354 000004
432 004356 012737 004372 001110
433 004364 012737 000020 002004
434
435 004372 013777 002004 175356
436 004400 017737 175352 001126
437 004406 013737 002004 001124
438 004414 023737 001124 001126
439 004422 001401
440 004424 104003
441 004426 006237 002004
442 004432 022737 000001 002004
443 004440 001354
444
445
(3)
(3)
(2) 004442 000004
(1) 004444 012737 000010 001160
446 004452 012777 000014 175276
447 004460 012737 000014 001124
448 004466 105077 175276
449 004472 017737 175260 001126
450 004500 023737 001124 001126
451 004506 001401
452 004510 104003
453
```

```
*****
*TEST 5 VERIFY LOW BYTE OPERATION OF THE "CSR" REGISTER
*****
TST5: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #3636,@CSR ;LOAD CSR REGISTER
MOV #3600,$GDDAT ;LOAD EXPECTED VALUE
CLRB @CSR ;CLEAR LOW BYTE
MOV @CSR,$BDDAT ;READ STATUS REG.
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 1$ ;;BR IF SAME
ERROR 2 ;CLEARING LOW BYTE OF THE CSR CHANGED THE HIGH B
1$: MOV #3636,@CSR ;LOAD CSR REGISTER
MOV #236,$GDDAT ;LOAD EXPECTED VALUE
CLRB @CSRHB ;CLEAR HIGH BYTE OF THE CSR
MOV @CSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ TST6 ;;BR IS SAME
ERROR 2 ;CLEARING HIGH BYTE OF CSR CHANGED THE LOW BYTE
*****
*TEST 6 FLOAT A 1 ACROSS 4 BITS OF THE SPECIAL FUNCTION REGISTER
*****
TST6: SCOPE
MOV #1$,$LPERR ;LOAD LOOP ADDRESS
MOV #BIT4,$TEMP ;LOAD INITIAL REG. VALUE
1$: MOV $TEMP,@SFR ;LOAD SFR REG.
MOV @SFR,$BDDAT ;READ SFR
MOV $TEMP,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE THE VALUES
BEQ 2$ ;;BR IF SAME
ERROR 3 ;UNEXPECTED VALUE IN THE SFR REGISTER
2$: ASR $TEMP ;TRY THE NEXT DATA BIT
CMP #1,$TEMP ;TEST IF NOW BIT 0
BNE 1$ ;BR IF NOT
*****
*TEST 7 VERIFY THAT CLEARING HIGH BYTE OF SFR DOES NOT CLEAR LOW BYTE
*****
TST7: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #14,@SFR ;LOAD THE S.F. REGISTER
MOV #14,$GDDAT ;LOAD THE EXPECTED VALUE
CLRB @SFRHB ;CLEAR HIGH BYTE OF S.F. REG.
MOV @SFR,$BDDAT ;READ THE REGISTER
CMP $GDDAT,$BDDAT ;COMPARE THE VALUES
BEQ TST10 ;;BR IF SAME
ERROR 3 ;CLEARING HIGH BYTE OF CSR REG. CHANGED THE LOW
```

455
(3)
(3)
(2) 004512 000004
(1) 004514 012737 000010 001160
456 004522 012777 000016 175226
457 004530 000005
458 004532 005037 001124
459 004536 017737 175214 001126
460 004544 001401
461 004546 104003
462
463
(3)
(3)
(2) 004550 000004
(1) 004552 012737 000010 001160
464 004560 012777 000016 175170
465 004566 052777 004000 175162
466 004574 005037 001124
467 004600 017737 175152 001126
468 004606 001401
469 004610 104003
470

```
::*****  
:*TEST 10 VERIFY THAT "INIT" CLEARS THE SFR REGISTER  
:*****  
TST10: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #16,@SFR ;LOAD SFR REG.  
RESET ;INITILIZE THE REGISTER  
CLR $GDDAT ;LOAD EXPECTED  
MOV @SFR,$BDDAT ;READ SFR REG.  
BEQ TST11 ;;BR IF EQUAL  
ERROR 3 ;"BUS INIT" FAILED TO CLEAR SFR REG.  
  
:*****  
:*TEST 11 VERIFY THAT "CLEAR ALL" CLEARS THE SFR REGISTER  
:*****  
TST11: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #16,@SFR ;LOAD SFR REG.  
BIS #CLRALL,@SFR ;GENERATE "CLR ALL L"  
CLR $GDDAT ;LOAD EXPECTED VALUE  
MOV @SFR,$BDDAT ;READ THE SFR REG.  
BEQ TST12 ;;BR IF SAME  
ERROR 3 ;"CLR ALL L" FAILED TO CLEAR SFR REG.
```

```

472
(3)
(3)
(2) 004612 000004
(1) 004614 012737 000100 001160
473 004622 012737 000001 001124
474 004630 012737 004636 001110
475
476 004636 012777 004000 175112 1$: MOV #CLRALL,@SFR ;RESET THE DEVICE
477 004644 013777 001124 175100 MOV $GDDAT,@WCR ;LOAD WORD COUNT "A"
478 004652 017737 175074 001126 MOV @WCR,$BDDAT ;READ WORD COUNT
479 004660 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
480 004666 001401 BEQ 2$ ;;BR IF SAME
481 004670 104004 ERROR 4 ;;WORD COUNT REG. IN ERROR
482
483 004672 006337 001124 2$: ASL $GDDAT ;CHANGE THE DATA
484 004676 001357 BNE 1$ ;BR IF MORE DATA TO LOAD
485
(3)
(3)
(2) 004700 000004
(1) 004702 012737 000100 001160
486 004710 012737 000001 001124
487 004716 012737 004724 001110
488
489 004724 012777 004000 175024 1$: MOV #CLRALL,@SFR ;RESET THE DEVICE
490 004732 013777 001124 175014 MOV $GDDAT,@BAR ;LOAD BUS ADDRESS "A"
491 004740 017737 175010 001126 MOV @BAR,$BDDAT ;READ BUS ADDRESS
492 004746 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
493 004754 001401 BEQ 2$ ;;BR IF SAME
494 004756 104005 ERROR 5 ;BUS ADDRESS REG. IN ERROR
495
496 004760 006337 001124 2$: ASL $GDDAT ;CHANGE THE DATA
497 004764 001357 BNE 1$ ;BR IF MORE DATA TO LOAD
498
(3)
(3)
(2) 004766 000004
(1) 004770 012737 000100 001160
499 004776 012737 000001 001124
500 005004 012737 005012 001110
501
502 005012 012777 004000 174736 1$: MOV #CLRALL,@SFR ;RESET THE DEVICE
503 005020 013777 001124 174722 MOV $GDDAT,@OFF ;LOAD OFFSET "A"
504 005026 017737 174716 001126 MOV @OFF,$BDDAT ;READ OFFSET
505 005034 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
506 005042 001401 BEQ 2$ ;;BR IF SAME
507 005044 104006 ERROR 6 ;OFFSET REG. IN ERROR
508
509 005046 006337 001124 2$: ASL $GDDAT ;CHANGE THE DATA
510 005052 001357 BNE 1$ ;BR IF MORE DATA TO LOAD

```

```
512 ::*****  
(3) :*TEST 15 VERIFY NO DUAL REGISTER SELECTION  
(3) :*****  
(2) 005054 000004 TST15: SCOPE  
(1) 005056 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
513 ;LOAD DIFFERENT NUMBERS INTO THE WCR, BAR AND OFF REGISTERS  
514 005064 012777 004000 174664 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
515 005072 012777 011111 174650 MOV #11111,@OFF ;LOAD OFFSET REGISTER  
516 005100 012777 022222 174644 MOV #22222,@WCR ;LOAD W.C. REGISTER  
517 005106 012777 033333 174640 MOV #33333,@BAR ;LOAD B.A. REGISTER  
518 005114 012777 000036 174634 MOV #36,@SFR ;LOAD SPECIAL FUNCTION REGISTER  
519 005122 012777 007636 174616 MOV #7636,@CSR ;LOAD COMMAND/STATUS REGISTER  
520 ;NOW READ EACH REGISTER AND CHECK THE VALUE  
521 005130 012737 011111 001124 MOV #11111,$GDDAT ;LOAD EXPECTED VALUE  
522 005136 017737 174606 001126 MOV @OFF,$BDDAT ;READ A OFFSET REG.  
523 005144 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
524 005152 001401 BEQ 1$ ;;BR IF SAME  
525 005154 104007 ERROR 7 ;DUAL ADDRESS ERROR  
526 005156 012737 022222 001124 1$: MOV #22222,$GDDAT ;LOAD EXPECTED VALUE  
527 005164 017737 174562 001126 MOV @WCR,$BDDAT ;READ W.C. REG.  
528 005172 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
529 005200 001401 BEQ 2$ ;;BR IF SAME  
530 005202 104007 ERROR 7 ;DUAL ADDRESS ERROR  
531 005204 012737 033333 001124 2$: MOV #33333,$GDDAT ;LOAD EXPECTED VLAUE  
532 005212 017737 174536 001126 MOV @BAR,$BDDAT ;READ B.A. REG.  
533 005220 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
534 005226 001401 BEQ 3$ ;;BR IF SAME  
535 005230 104007 ERROR 7 ;DUAL ADDRESS ERROR  
536 005232 012737 000036 001124 3$: MOV #36,$GDDAT ;LOAD EXPECTED VALUE  
537 005240 017737 174512 001126 MOV @SFR,$BDDAT ;READ SPECIAL FUNCTION REG  
538 005246 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
539 005254 001401 BEQ 4$ ;;BR IF SAME  
540 005256 104007 ERROR 7 ;DUAL ADDRESS ERROR  
541 005260 012737 007636 001124 4$: MOV #7636,$GDDAT ;LOAD EXPECTED VALUE  
542 005266 017737 174454 001126 MOV @CSR,$BDDAT ;READ COMMAND/STATUS REGISTER  
543 005274 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
544 005302 001401 BEQ 5$ ;;BR IF SAME  
545 005304 104007 ERROR 7 ;DUAL ADDRESS ERROR  
546 005306 012777 004000 174442 5$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
```

```
548  
(3) :*****  
(3) :*TEST 16 VERIFY "CLR ALL" CLEARS THE EXTENDED OFFSET BITS  
(2) :*****  
(1) 005314 000004 TST16: SCOPE  
549 005316 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
550 005324 005037 001124 CLR $GDDAT ;CLEAR EXPECTED VALUE  
551 005330 012777 004000 174420 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
552 005336 012777 000003 174404 MOV #3,@OFF ;LOAD EXTENDED OFFSET REGISTER  
553 005344 012777 004000 174404 MOV #CLRALL,@SFR ;CLEAR THE EXTENDED OFFSET REG.'S  
554 005352 017737 174372 001126 MOV @OFF,$BDDAT ;READ EXTENDED OFFSET REG.  
555 005360 001401 BEQ TST17 ;;BR IF CLEARED  
556 005362 104006 ERROR 6 ;CLEAR ALL FAILED TO CLEAR EXTENDED OFFSET REGIS  
557 :*****  
(3) :*TEST 17 TEST THE "ACTIVE" FLOP CAN SET AND CLEAR  
(3) :*****  
(2) 005364 000004 TST17: SCOPE  
558 005366 012777 004000 174362 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
559 005374 052777 000010 174354 BIS #TSTDMA,@SFR ;SET THE "TEST DMA" TO PREVENT DATA TRANSFERS  
560 005402 000240 NOP  
561 005404 000240 NOP  
562 005406 000240 NOP  
563 005410 052777 000022 174330 BIS #BIT4!BIT1,@CSR ;SET "MATRIX MODE" AND "BYTE" MODE  
564 :NOW SET THE "ACTIVE" FLOP AND VERIFY "INTERFACE IDLE" GOES LOW  
565 005416 012737 000022 001124 MOV #BIT4!BIT1,$GDDAT ;LOAD EXPECTED VALUE  
566 005424 052777 000001 174314 BIS #BIT0,@CSR ;SET "ACTIVE" TO A 1  
567 005432 017737 174310 001126 MOV @CSR,$BDDAT ;READ STATUS  
568 005440 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES  
569 005446 001404 BEQ 1$ ;;BR IF EXPECTED  
570 005450 052777 000400 174300 BIS #ENDDMA,@SFR ;STOP DMA IF POSSIBLE  
571 005456 104002 ERROR 2 ;"ACTIVE" FLOP FAILED TO SET  
572 :POKE THE "END DMA" SIGNAL AND VERIFY "ACTIVE" CLEARS  
573 005460 052777 000400 174270 1$: BIS #ENDDMA,@SFR ;SEND "END DMA" SIGNAL  
574 005466 012737 000222 001124 MOV #BIT7!BIT4!BIT1,$GDDAT ;SET "INTERFACE IDLE" INTO EXPECTED  
575 005474 017737 174246 001126 MOV @CSR,$BDDAT ;READ STATUS  
576 005502 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
577 005510 001401 BEQ 2$ ;;BR IF SAME  
578 005512 104002 ERROR 2 ;"END DMA" AGAIN FAILED TO CLEAR "ACTIVE" FLOP  
579 :POKE "ENABLE DMA" AND THEN ISSUE "CLR ALL" SIGNAL TO  
580 : ENSURE THE "ACTIVE" FLOP CLEARS  
581 005514 012737 000200 001124 2$: MOV #BIT7,$GDDAT ;LOAD EXPECTED  
582 005522 052777 000001 174216 BIS #BIT0,@CSR ;POKE "ENABLE DMA"  
583 005530 052777 004000 174220 BIS #CLRALL,@SFR ;GENERATE "CLR ALL"  
584 005536 017737 174204 001126 MOV @CSR,$BDDAT ;READ STATUS  
585 005544 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
586 005552 001401 BEQ TST20 ;;BR IF SAME  
587 005554 104001 ERROR 1 ;"CLR ALL" FAILED TO CLEAR "ACTIVE" FLOPS
```

```
592 ::*****  
(3) :*TEST 20 VERIFY Z INPUTS CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN MATRI  
(3) :*****  
(2) 005556 000004 TST20: SCOPE  
(1) 005560 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS  
593 005566 012777 004000 174162 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
594 005574 012777 000010 174154 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP  
595 005602 012777 000000 174142 MOV #0,@WCR ;LOAD W.C. REG  
596 005610 012777 000000 174136 MOV #0,@BAR ;LOAD B.A. REG  
597 005616 012777 000022 174122 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE  
598 005624 052777 000001 174114 BIS #BIT0,@CSR ;GO "ACTIVE"  
599 005632 052777 000002 174116 BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
(1) 005640 042777 000002 174110 BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
600 005646 012737 000001 001124 MOV #1,$GDDAT ;LOAD EXPECTED  
601 005654 017737 174074 001126 MOV @BAR,$BDDAT ;READ LOW 16 BITS  
602 005662 001002 BNE 1$ ;;BR IF NON-ZERO  
603 005664 104010 ERROR 10 ;Z INPUT FAILED TO CAUSE THE LOW BYTE OF 32 BIT  
604 005666 000420 BR TST21 ;;  
605 005670 017737 174056 001126 1$: MOV @WCR,$BDDAT ;READ HIGH 16 BITS  
606 005676 001401 BEQ 2$ ;;BR IF CLEARED  
607 005700 104011 ERROR 11 ;HIGH 16 BIS CHANGED IN ERROR  
608 005702 012737 000020 001124 2$: MOV #BIT4,$GDDAT ;LOAD EXPECTED  
609 005710 017737 174032 001126 MOV @CSR,$BDDAT ;READ STATUS  
610 005716 032737 040000 001126 BIT #BIT14,$BDDAT ;TEST IF "CELL OVERFLOW"  
611 005724 001401 BEQ TST21 ;;BR IF CORRECT  
612 005726 104012 ERROR 12 ;"CELL OVERFLOW" FLOP SET IN ERROR  
613 :*****  
(3) :*TEST 21 VERIFY Z INPUTS CAUSE THE LOW 16 BITS OF 32 BIT COUNTER TO CHANGE  
(3) :*****  
(2) 005730 000004 TST21: SCOPE  
(1) 005732 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS  
614 005740 012777 004000 174010 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
615 005746 012777 000010 174002 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP  
616 005754 012777 000000 173770 MOV #0,@WCR ;LOAD W.C. REG  
617 005762 012777 000376 173764 MOV #376,@BAR ;LOAD B.A. REG  
618 005770 012777 000022 173750 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE  
619 005776 052777 000001 173742 BIS #BIT0,@CSR ;GO "ACTIVE"  
620 006004 052777 000002 173744 BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
(1) 006012 042777 000002 173736 BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
621 006020 012737 000400 001124 MOV #BIT8,$GDDAT ;LOAD EXPECTED  
622 006026 017737 173722 001126 MOV @BAR,$BDDAT ;READ LOW 16 BITS  
623 006034 105737 001127 TSTB $BDDAT+1 ;TEST HIGH BYTE  
624 006040 001002 BNE 1$ ;BR IF NON-ZERO  
625 006042 104010 ERROR 10 ;Z INPUT FAILED TO CAUSE THE HIGH BYTE OF THE LO  
626 006044 000423 BR TST22 ;;  
627 006046 012737 000000 001124 1$: MOV #0,$GDDAT ;LOAD EXPECTED  
628 006054 017737 173672 001126 MOV @WCR,$BDDAT ;READ HIGH 16 BITS  
629 006062 001401 BEQ 2$ ;;BR IF CLEARED  
630 006064 104011 ERROR 11 ;HIGH 16 BITS CHANGED IN ERROR  
631 006066 012737 000020 001124 2$: MOV #BIT4,$GDDAT ;LOAD EXPECTED  
632 006074 017737 173646 001126 MOV @CSR,$BDDAT ;READ STATUS  
633 006102 032737 040000 001126 BIT #BIT14,$BDDAT ;TEST IF "CELL OVERFLOW"  
634 006110 001401 BEQ TST22 ;;BR IF CORRECT  
635 006112 104012 ERROR 12 ;"CELL OVERFLOW" FLOP SET IN ERROR
```

```

637
(3)
(3)
(2) 006114 000004
(1) 006116 012737 000100 001160
638 006124 012777 004000 173624
639 006132 012777 000010 173616
640 006140 012777 000000 173604
641 006146 012777 177776 173600
642 006154 012777 000022 173564
643 006162 052777 000001 173556
644 006170 052777 000002 173560
(1) 006176 042777 000002 173552
645 006204 012737 000001 001124
646 006212 017737 173534 001126
647 006220 001002
648 006222 104011
649 006224 000413
650 006226 012737 000020 001124
651 006234 017737 173506 001126
652 006242 032737 040000 001126
653 006250 001401
654 006252 104012
655
(3)
(3)
(2) 006254 000004
(1) 006256 012737 000100 001160
656 006264 012777 004000 173464
657 006272 012777 000010 173456
658 006300 012777 000377 173444
659 006306 012777 177776 173440
660 006314 012777 000022 173424
661 006322 052777 000001 173416
662 006330 052777 000002 173420
(1) 006336 042777 000002 173412
663 006344 012737 000400 001124
664 006352 017737 173374 001126
665 006360 105737 001127
666 006364 001001
667 006366 104011

*****
*TEST 22 VERIFY Z INPUTS CAUSE THE LOW 24 BITS OF 32 BIT COUNTER TO CHANGE
*****
TST22: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #0,@WCR ;LOAD W.C. REG
MOV #-2,@BAR ;LOAD B.A. REG
MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;GO "ACTIVE"
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV #1,$GDDAT ;LOAD EXPECTED
1$: MOV @WCR,$BDDAT ;READ HIGH 16 BITS
BNE 2$ ;BR IF SET
ERROR 11 ;LOW BYTE OF THE HIGH 16 BITS FAILED TO CHANGE
BR TST23 ;;
2$: MOV #BIT4,$GDDAT ;LOAD EXPECTED
MOV @CSR,$BDDAT ;READ STATUS
BIT #BIT14,$BDDAT ;TEST IF "CELL OVERFLOW"
BEQ TST23 ;;BR IF CORRECT
ERROR 12 ;"CELL OVERFLOW" FLOP SET IN ERROR
*****
*TEST 23 VERIFY Z INPUTS CAUSE THE HIGH 8 BITS OF THE 32 BIT COUNTER TO CHANGE
*****
TST23: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #377,@WCR ;LOAD W.C. REG
MOV #-2,@BAR ;LOAD B.A. REG
MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;GO "ACTIVE"
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV #18,$GDDAT ;LOAD EXPECTED
1$: MOV @WCR,$BDDAT ;READ HIGH 16 BITS
TSTB $BDDAT+1 ;TEST HIGH BYTE
BNE TST24 ;;BR IF SET
ERROR 11 ;HIGH 8 BITS OF THE 32 BIT COUNTER FAILED TO CHANGE

```



```

669
(3)
(3)
(2) 006370 000004
(1) 006372 012737 000100 001160
670 006400 012777 004000 173350
671 006406 012777 000000 173336
672 006414 012777 000000 173332
673 006422 005077 173320
674 006426 052777 000002 173322
(1) 006434 042777 000002 173314
675 006442 012737 000000 001124
676 006450 017737 173300 001126
677 006456 001401
678 006460 104010
679
680
(3)
(3)
(2) 006462 000004
(1) 006464 012737 000010 001160
681 006472 012777 004000 173256
682 006500 012777 000010 173250
683 006506 012777 177777 173236
684 006514 012777 177776 173232
685 006522 012777 000022 173216
686 006530 052777 000001 173210
687
688 006536 052777 000002 173212
(1) 006544 042777 000002 173204
689 006552 013700 002006
690 006556 005001
691 006560 012737 040020 001124
692
693 006566 032777 040000 173152 1$:
694 006574 001011
695 006576 005301
696 006600 001372
697 006602 005300
698 006604 001370
699 006606 012777 000400 173142
700 006614 104012
701 006616 000416
702
703
704 006620 052777 004000 173130 2$:
705 006626 012737 000200 001124
706 006634 017737 173106 001126
707 006642 023737 001124 001126
708 006650 001401
709 006652 104012

```

```

*****
*TEST 24 VERIFY Z INPUTS DO NOT CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE
*****
TST24: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #0,@WCR ;LOAD W.C. REG
MOV #0,@BAR ;LOAD B.A. REG
CLR @CSR ;ENSURE LIST MODE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV #0,$GDDAT ;LOAD EXPECTED
MOV @BAR,$BDDAT ;READ LOW 16 BITS
BEQ TST25 ;;BR IF ZERO
ERROR 10 ;Z INPUT CAUSE THE 32 BIT COUNTER TO CHANGE IN LIST MODE

*****
*TEST 25 TEST THAT "WCA OVFL" SETS Z/WC OVERFLOW FLOP AND "CLR ALL" CLEARS IT
*****
TST25: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #-1,@WCR ;LOAD Z COUNTER
MOV #-2,@BAR ;LOAD Z COUNTER
MOV #BIT4!BIT1,@CSR ;ENTER "MATRIX" MODE
BIS #BIT0,@CSR ;ENABLE NCV11
;NOW ENABLE "TEST Z" INPUTS
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV CPUDLO,R0 ;LOAD GROSS TIMER
CLR R1
MOV #BIT14!BIT4,$GDDAT ;LOAD EXPECTED STATUS
1$: BIT #BIT14,@CSR ;TEST FOR Z/WC OVERFLOW
BNE 2$ ;BR IF SET
DEC R1 ;DELAY
BNE 1$
DEC R0 ;DELAY
BNE 1$
MOV #ENDDMA,@SFR ;STOP DMA TRANSFERS
ERROR 12 ;AFTER A GROSS TIME, THE Z/WC OVERFLOW FLOP FAILED TO SE
BR TST26 ;;

;NOW GENERATE "CLR ALL" TO CLEAR Z/WC OVERFLOW BIT
2$: BIS #CLRALL,@SFR ;CLEAR Z/WC FLOP
MOV #BIT7,$GDDAT ;LOAD EXPECTED
MOV @CSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST26 ;;BR IF SAME
ERROR 12 ;"CLR ALL" FAILED TO CLEAR "Z/WC" FLOP

```

711
(3)
(3)
(2) 006654 000004
(1) 006656 012737 000010 001160
712 006664 012777 004000 173064
713 006672 012777 000010 173056
714 006700 012777 177777 173044
715 006706 012777 177776 173040
716 006714 012777 000022 173024
717 006722 052777 000001 173016
718
719 006730 052777 000002 173020
(1) 006736 042777 000002 173012
720
721 006744 013700 002006
722 006750 005001
723 006752 012737 040020 001124
724
725 006760 032777 040000 172760 1\$:
726 006766 001014
727 006770 005301
728 006772 001372
729 006774 005300
730 006776 001370
731 007000 012777 000400 172750
732 007006 042777 000002 172742
733 007014 104012
734 007016 000416
735
736
737 007020 052777 040000 172730 2\$:
738 007026 012737 000222 001124
739 007034 017737 172706 001126
740 007042 023737 001124 001126
741 007050 001401
742 007052 104012
743

```
*****  
;*TEST 26 TEST THAT 'WCA OVFL' SETS Z/WC OVERFLOW FLOP AND 'CLR WC OVFL' CLEARS IT  
*****  
TST26: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP  
MOV #-1,@WCR ;LOAD Z COUNTER  
MOV #-2,@BAR ;LOAD Z COUNTER  
MOV #BIT4!BIT1,@CSR ;ENABLE 'MATRIX' MODE  
BIS #BIT0,@CSR ;SET NCV11 ACTIVE  
;NOW ENABLE 'TEST Z' INPUTS  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV CPUDLO,R0 ;LOAD GROSS TIMER  
CLR R1  
MOV #BIT14!BIT4,$GDDAT ;LOAD EXPECTED STATUS  
1$: BIT #BIT14,@CSR ;TEST FOR Z/WC OVERFLOW  
BNE 2$ ;BR IF SET  
DEC R1 ;DELAY  
BNE 1$  
DEC R0 ;DELAY  
BNE 1$  
MOV #ENDDMA,@SFR ;STOP DAM TRANSFERS  
BIC #TESTZ,@SFR ;STOP 'Z' INPUTS  
ERROR 12 ;AFTER A GROSS TIME, THE Z/WC OVERFLOW FLOP FAILED TO SET  
BR TST27 ;;  
;NOW GENERATE 'CLR WC OVFL' TO CLEAR Z/WC OVERFLOW BIT  
2$: BIS #CLRWC0,@SFR ;CLEAR Z/WC FLOP  
MOV #BIT7!BIT4!BIT1,$GDDAT ;LOAD EXPECTED  
MOV @CSR,$BDDAT ;READ STATUS  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST27 ;;BR IF SAME  
ERROR 12 ;'CLR WC OVFL' FAILED TO  
; CLEAR 'Z/WC' FLOP (BIT 14)
```

```
745      ;*****  
(3)      ;*TEST 27      TEST THAT 'WCA OVFL' GENERATES AN INTERRUPT  
(3)      ;*****  
(2) 007054 000004  
(1) 007056 012737 000010 001160 TST27: SCOPE  
746 007064 012777 004000 172664      MOV #10,$TIMES      ;;DO 10 ITERATIONS  
747 007072 012777 000010 172656      MOV #CLRALL,@SFR      ;CLEAR DEVICE  
748 007100 012777 177777 172644      MOV #TSTDMA,@SFR      ;SET TEST DMA FLOP  
749 007106 012777 177776 172640      MOV #-1,@WCR      ;LOAD Z COUNTER  
750 007114 012777 000022 172624      MOV #-2,@BAR      ;LOAD Z COUNTER  
751 007122 052777 000001 172616      MOV #BIT4!BIT1,@CSR      ;SET MATRIX MODE  
752      BIS #BIT0,@CSR      ;SET NCV11 ACTIVE  
753 007130 052777 000002 172620 ;NOW ENABLE 'TEST Z' INPUTS  
(1) 007136 042777 000002 172612      BIS #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES  
754 007144 013700 002006      BIC #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES  
755 007150 005001      MOV CPUDLO,R0      ;LOAD GROSS TIMER  
756 007152 012737 040020 001124      CLR R1  
757      MOV #BIT14!BIT4,$GDDAT      ;LOAD EXPECTED STATUS  
758 007160 032777 040000 172560 1$: BIT #BIT14,@CSR      ;TEST FOR Z/WC OVERFLOW  
759 007166 001014      BNE 2$      ;BR IF SET  
760 007170 C05301      DEC R1      ;DELAY  
761 007172 001372      BNE 1$  
762 007174 005300      DEC R0      ;DELAY  
763 007176 001370      BNE 1$  
764 007200 012777 000400 172550      MOV #ENDDMA,@SFR      ;STOP DMA TRANSFERS  
765 007206 042777 000002 172542      BIC #TESTZ,@SFR      ;STOP 'Z' INPUTS  
766 007214 104012      ERROR 12      ;AFTER A GROSS TIME, THE Z/WC OVERFLOW FLOP FAILED TO SET  
767 007216 000430      BR 5$      ;;BR TO CLEAN UP  
768  
769      ;NOW ENABLE THE WC/Z OVERFLOW INTERRUPT BIT AND WAIT FOR AN INTERRUPT  
770 007220 012746 000000 2$: MOV #0,-(SP)  
771 007224 012746 007232      MOV #3$,-(SP)      ;LSI-11 HACK  
772 007230 000002      RTI  
773 007232 012777 007276 172532 3$: MOV #4$,@VECTA0      ;LOAD INTR. VECTOR  
774 007240 052777 000100 172500      BIS #BIT6,@CSR      ;ENABLE INTERRUPT  
775 007246 000240      NOP  
776 007250 000240      NOP  
777 007252 000240      NOP  
778 007254 000240      NOP  
779 007256 017737 172464 001126      MOV @CSR,$BDDAT      ;READ STATUS  
780 007264 012777 004000 172464      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE  
781 007272 104013      ERROR 13      ;WC/Z OVERFLOW FAILED TO GENERATE INTERRUPT  
782 007274 000401      BR 5$      ;;BR TO CLEAN UP  
783  
784 007276 022626 4$: CMP (SP)+,(SP)+  
785 007300 005077 172442 5$: CLR @CSR  
786 007304 012777 004000 172444      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE  
787 007312 013777 001774 172452      MOV VECTA1,@VECTA0  
788 007320 005077 172450      CLR @VECTA1  
789
```

```

791
(3)
(3)
(2) 007324 000004
(1) 007326 012737 000010 001160
792 007334 012777 004000 172414
793 007342 012777 000014 172406
794 007350 012777 177777 172374
795 007356 012777 177776 172370
796 007364 005077 172360
797 007370 012777 000022 172350
798 007376 052777 000001 172342
799 007404 052777 000002 172344
(1) 007412 042777 000002 172336
800 007420 017737 172322 001126
801 007426 012737 040222 001124
802 007434 023737 001124 001126
803 007442 001402
804 007444 104012
805
806 007446 000407
807
808 007450 005037 001124 1$: CLR $GDDAT
809 007454 017737 172270 001126 MOV @OFF,$BDDAT
810 007462 001401 BEQ TST31
811 007464 104012 ERROR 12
812

```

```

*****
:*TEST 30 VERIFY 'WCA OVFL' CLEARS 'ACTIVE'
*****
TST30: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA
MOV #-1,@WCR ;LOAD W.C. REG.
MOV #-2,@BAR ;LOAD BAR REG.
CLR @OFF ;CLEAR OFFSET REG.
MOV #BIT4!BIT1,@CSR ;ENABLE MATRIX MODE
BIS #BIT0,@CSR ;SET 'ACTIVE'
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @CSR,$BDDAT ;READ STATUS
MOV #BIT14!BIT7!BIT4!BIT1,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF SAME
ERROR 12 ;'WCA OVFL' FAILED TO
;CLEAR 'ACTIVE' (BIT 7 SET)
BR TST31 ;;
1$: CLR $GDDAT ;CLEAR THE EXPECTED
MOV @OFF,$BDDAT ;READ THE ACTUAL
BEQ TST31 ;;BR IF CLEARED
ERROR 12 ;'WCA INC X OFF' IN MATRIX MODE CHANGED
;THE OFFSET REGISTER

```

```

817      ;*****
(3)      ;*TEST 31      VERIFY JOYSTICK DONE FLOP SETS
(3)      ;*****
(2) 007466 000004      TST31: SCOPE
(1) 007470 012737 000040 001160      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
818 007476 012777 004000 172252      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
819 007504 052777 000001 172244      BIS      #REDJOY,@SFR      ;REQUEST JOYSTICK DATA
820 007512 013700 002010      MOV      CPUDL1,R0      ;LOAD DELAY COUNTER
821 007516 105777 172234      1$: TSTB  @SFR      ;WAIT FOR JOYSTICK READY
822 007522 100411      BMI      2$      ;BR IF SET
823 007524 005300      DEC      R0      ;DELAY
824 007526 001373      BNE      1$      ;BR IF NOT EXHAUSTED
825 007530 017737 172222 001126      MOV      @SFR,$BDDAT      ;READ STATUS
826 007536 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
827 007544 104014      ERROR 14      ;"JOYSTICK READY" FAILED TO SET
828      ;NOW PERFORM A "TST" INSTRUCTION TO THE JOYSTICK REGISTER
829      ;THE ACCESS OF THIS BUS ADDRESS SHOULD CLEAR JOY READY FLOP"
830 007546 005777 172210      2$: TST  @JOY      ;ADDRESS THE REGISTER
831 007552 017737 172200 001126      MOV      @SFR,$BDDAT      ;READ STATUS
832 007560 105737 001126      TSTB  $BDDAT      ;TEST IF THE BIT CLEARED
833 007564 100003      BPL      TST32      ;;BR IF YES
834 007566 005037 001124      CLR      $GDDAT      ;LOAD EXPECTED
835 007572 104014      ERROR 14      ;ADDRESSING THE JOYSTICK ADDRESS FAILED TO CLEAR
836      ;*****
(3)      ;*TEST 32      VERIFY THAT "RESET" INSTRUCTION CLEARS THE JOY READY FLOP
(3)      ;*****
(2) 007574 000004      TST32: SCOPE
(1) 007576 012737 000040 001160      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
837 007604 012777 004000 172144      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
838 007612 052777 000001 172136      BIS      #REDJOY,@SFR      ;REQUEST JOYSTICK DATA
839 007620 013700 002010      MOV      CPUDL1,R0      ;LOAD DELAY
840 007624 105777 172126      1$: TSTB  @SFR      ;WAIT FOR JOYSTICK READY
841 007630 100411      BMI      2$      ;BR IF SET
842 007632 005300      DEC      R0      ;DELAY
843 007634 001373      BNE      1$      ;BR IF NOT EXHAUSTED
844 007636 017737 172114 001126      MOV      @SFR,$BDDAT      ;READ STATUS
845 007644 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED VALUE
846 007652 104014      ERROR 14      ;JOY READY FAILED TO SET
847      ;NOW ISSUE A "CLR ALL" AND VERIFY THE "JOY READY" DOES NOT CLEAR
848 007654 052777 004000 172074      2$: BIS  #CLRALL,@SFR      ;GENERATE "CLR ALL"
849 007662 017737 172070 001126      MOV      @SFR,$BDDAT      ;READ STATUS
850 007670 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED VALUE
851 007676 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
852 007704 001401      BEQ      3$      ;;BR IF SAME
853 007706 104014      ERROR 14      ;"CLR ALL" CLEARED THE "JOY READY" FLOP IN ERROR
854      ;NOW ISSUE A BUS "RESET" AND VERIFY THE "JOY READY" CLEARS
855 007710 000005      3$: RESET      ;BUS "INIT"
856 007712 017737 172040 001126      MOV      @SFR,$BDDAT      ;READ STATUS
857 007720 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED
858 007726 017737 172024 001126      MOV      @SFR,$BDDAT      ;READ STATUS
859 007734 001401      BEQ      TST33      ;;BR IF CLEARED
860 007736 104014      ERROR 14      ;BUS INIT FAILED TO CLEAR JOY READY

```

873

874

(3)

(3)

(2)

(1)

875

(1)

(1)

(1)

(1)

(1)

(1)

(1)

876

877

878

879

(3)

(3)

(2)

(1)

880

(1)

(1)

(1)

(1)

(1)

(1)

(1)

881

882

883

884

(3)

(3)

(2)

(1)

885

(1)

(1)

(1)

(1)

(1)

(1)

(1)

886

887

*TEST 33 JOYSTICK DATA PATH = GAIN = 0 ZB ENABLE = 0 RES. = 000

TST33: SCOPE
MOV #10,\$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #0,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1\$
MOV @JOY,\$BDDAT ;READ THE REGISTER
MOV #0,\$GDDAT ;LOAD EXPECTED
CMP \$GDDAT,\$BDDAT ;COMPARE
BEQ TST34 ;;BR IF SAME
ERROR 15 ;JOYSTICK DATA PATH BIT SET

*TEST 34 JOYSTICK DATA PATH = GAIN = 1

TST34: SCOPE
MOV #10,\$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #2000,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1\$
MOV @JOY,\$BDDAT ;READ THE REGISTER
MOV #124250,\$GDDAT ;LOAD EXPECTED
CMP \$GDDAT,\$BDDAT ;COMPARE
BEQ TST35 ;;BR IF SAME
ERROR 15 ;JOYSTICK DATA PATH ERROR

*TEST 35 JOYSTICK DATA PATH = ZB ENABLE = 1

TST35: SCOPE
MOV #10,\$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #4000,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1\$
MOV @JOY,\$BDDAT ;READ THE REGISTER
MOV #050120,\$GDDAT ;LOAD EXPECTED
CMP \$GDDAT,\$BDDAT ;COMPARE
BEQ TST36 ;;BR IF SAME
ERROR 15 ;JOYSTICK DATA PATH ERROR

```

889      ;:*****
(3)      ;*TEST 36      JOYSTICK DATA PATH RES. = 001
(3)      ;:*****
(2) 010224 000004      TST36: SCOPE
(1) 010226 012737 000010 001160      MOV #10,$TIMES      ;;DO 10 ITERATIONS
890 010234 012777 004000 171514      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE
(1) 010242 012777 000014 171506      MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 010250 012777 000023 171470      MOV #23,@CSR      ;LOAD CSR REGISTER
(1) 010256 052777 000001 171472      BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 010264 105777 171466      1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 010270 100375      BPL 1$
(1) 010272 017737 171464 001126      MOV @JOY,$BDDAT ;READ THE REGISTER
(1) 010300 012737 000401 001124      MOV #401,$GDDAT ;LOAD EXPECTED
(1) 010306 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
891 010314 001401      BEQ TST37 ;:BR IF SAME
892 010316 104015      ERROR 15 ;JOYSTICK DATA PATH ERROR RES. = 001
893
894      ;:*****
(3)      ;*TEST 37      JOYSTICK DATA PATH RES. = 010
(3)      ;:*****
(2) 010320 000004      TST37: SCOPE
(1) 010322 012737 000010 001160      MOV #10,$TIMES      ;;DO 10 ITERATIONS
895 010330 012777 004000 171420      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE
(1) 010336 012777 000014 171412      MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 010344 012777 000025 171374      MOV #25,@CSR      ;LOAD CSR REGISTER
(1) 010352 052777 000001 171376      BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 010360 105777 171372      1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 010364 100375      BPL 1$
(1) 010366 017737 171370 001126      MOV @JOY,$BDDAT ;READ THE REGISTER
(1) 010374 012737 001002 001124      MOV #1002,$GDDAT ;LOAD EXPECTED
(1) 010402 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
896 010410 001401      BEQ TST40 ;:BR IF SAME
897 010412 104015      ERROR 15 ;JOYSTICK DATA PATH ERROR RES. = 010
898
899      ;:*****
(3)      ;*TEST 40      JOYSTICK DATA PATH RES. = 100
(3)      ;:*****
(2) 010414 000004      TST40: SCOPE
(1) 010416 012737 000010 001160      MOV #10,$TIMES      ;;DO 10 ITERATIONS
900 010424 012777 004000 171324      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE
(1) 010432 012777 000014 171316      MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 010440 012777 000031 171300      MOV #31,@CSR      ;LOAD CSR REGISTER
(1) 010446 052777 000001 171302      BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 010454 105777 171276      1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 010460 100375      BPL 1$
(1) 010462 017737 171274 001126      MOV @JOY,$BDDAT ;READ THE REGISTER
(1) 010470 012737 002004 001124      MOV #2004,$GDDAT ;LOAD EXPECTED
(1) 010476 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
901 010504 001401      BEQ TST41 ;:BR IF SAME
902 010506 104015      ERROR 15 ;JOY STICK DATA PATH ERROR RES. = 100

```

```
907
(3)
(3)
(2) 010510 000004
(1) 010512 012737 000010 001160
908 010520 012777 004000 171230
(1) 010526 012777 000014 171222
(1) 010534 012777 000421 171204
(1) 010542 052777 000001 171206
(1) 010550 105777 171202
(1) 010554 100375
(1) 010556 017737 171200 001126
(1) 010564 012737 000401 001124
(1) 010572 023737 001124 001126
909 010600 001401
910 010602 104016
911
912
(3)
(3)
(2) 010604 000004
(1) 010606 012737 000010 001160
913 010614 012777 004000 171134
(1) 010622 012777 000014 171126
(1) 010630 012777 002437 171110
(1) 010636 052777 000001 171112
(1) 010644 105777 171106
(1) 010650 100375
(1) 010652 017737 171104 001126
(1) 010660 012737 130260 001124
(1) 010666 023737 001124 001126
914 010674 001401
915 010676 104016
916
917
918
(3)
(3)
(2) 010700 000004
(1) 010702 012737 000010 001160
919 010710 012777 004000 171040
(1) 010716 012777 000014 171032
(1) 010724 012777 006437 171014
(1) 010732 052777 000001 171016
(1) 010740 105777 171012
(1) 010744 100375
(1) 010746 017737 171010 001126
(1) 010754 012737 177777 001124
(1) 010762 023737 001124 001126
920 010770 001401
921 010772 104016
922
```

```
*****
*TEST 41 VERIFY THE DATA INCREMENT FUNCTION
*****
TST41: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #421,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1$
MOV @JOY,$BDDAT ;READ THE REGISTER
MOV #401,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST42 ;;BR IF SAME
ERROR 16 ;MAINT. CAM01 FAILED TO INCREMENT DATA REGISTER

*****
*TEST 42 VERIFY THE DATA INCREMENT CARRY BIT
*****
TST42: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #2437,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1$
MOV @JOY,$BDDAT ;READ THE REGISTER
MOV #130260,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST43 ;;BR IF SAME
ERROR 16 ;MAINT. CAM01 WITH RES.=7, G=1, ZB =0
;FAILED TO CAUSE DATA INCREMENT CARRY PROPERLY

*****
*TEST 43 VERIFY THE DATA INCREMENT FUNCTION IS INHIBITED
*****
TST43: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
MOV #6437,@CSR ;LOAD CSR REGISTER
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY
BPL 1$
MOV @JOY,$BDDAT ;READ THE REGISTER
MOV #177777,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST44 ;;BR IS SAME
ERROR 16 ;FAILED TO INHIBIT DATA INCREMENT FUNCTION
;IF THE BAD DATA WAS 0
```


924
(3) :*****
(3) :*TEST 44 VERIFY THE DATA DECREMENT FUNCTION
(2) :*****
(1) 010774 000004 TST44: SCOPE
(1) 010776 012737 000010 001160 MOV #10,\$TIMES ;;DO 10 ITERATIONS
925 011004 012777 004000 170744 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
(1) 011012 012777 000014 170736 MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 011020 012777 001023 170720 MOV #1023,@CSR ;LOAD CSR REGISTER
(1) 011026 052777 000001 170722 BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 011034 105777 170716 1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 011040 100375 BPL 1\$
(1) 011042 017737 170714 001126 MOV @JOY,\$BDDAT ;READ THE REGISTER
(1) 011050 012737 000000 001124 MOV #0,\$GDDAT ;LOAD EXPECTED
(1) 011056 023737 001124 001126 CMP \$GDDAT,\$BDDAT ;COMPARE
926 011064 001401 BEQ TST45 ;;BR IF SAME
927 011066 104017 ERROR 17 ;IF DATA WAS 401, THE DATA DECREMENT FUNCTION FAILED
928 ;OTHERWISE DECREMENTED DATA ERROR
929

930 :*****
(3) :*TEST 45 VERIFY THE DATA DECREMENT BORROW
(3) :*****
(2) 011070 000004 TST45: SCOPE
(1) 011072 012737 000010 001160 MOV #10,\$TIMES ;;DO 10 ITERATIONS
931 011100 012777 004000 170650 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
(1) 011106 012777 000014 170642 MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 011114 012777 007037 170624 MOV #7037,@CSR ;LOAD CSR REGISTER
(1) 011122 052777 000001 170626 BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 011130 105777 170622 1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 011134 100375 BPL 1\$
(1) 011136 017737 170620 001126 MOV @JOY,\$BDDAT ;READ THE REGISTER
(1) 011144 012737 177376 001124 MOV #177376,\$GDDAT ;LOAD EXPECTED
(1) 011152 023737 001124 001126 CMP \$GDDAT,\$BDDAT ;COMPARE
932 011160 001401 BEQ TST46 ;;BR IF SAME
933 011162 104017 ERROR 17 ;DECREMENTED DATA ERROR
934

935 :*****
(3) :*TEST 46 VERIFY THE DATA DECREMENT FUNCTION IS INHIBITED
(3) :*****
(2) 011164 000004 TST46: SCOPE
(1) 011166 012737 000010 001160 MOV #10,\$TIMES ;;DO 10 ITERATIONS
936 011174 012777 004000 170554 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
(1) 011202 012777 000014 170546 MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP
(1) 011210 012777 001021 170530 MOV #1021,@CSR ;LOAD CSR REGISTER
(1) 011216 052777 000001 170532 BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA
(1) 011224 105777 170526 1\$: TSTB @SFR ;WAIT FOR JOYSTICE READY
(1) 011230 100375 BPL 1\$
(1) 011232 017737 170524 001126 MOV @JOY,\$BDDAT ;READ THE REGISTER
(1) 011240 012737 000000 001124 MOV #0,\$GDDAT ;LOAD EXPECTED
(1) 011246 023737 001124 001126 CMP \$GDDAT,\$BDDAT ;COMPARE
937 011254 001401 BEQ TST47 ;;BR IF SAME
938 011256 104017 ERROR 17 ;INHIBIT DECREMENT FUNCTION ERROR
939 ;IF DATA WAS 177777

956
(4)
(4)
(3) 011260 000004
(2) 011262 012737 000010 001160
(1) 011270 012777 004000 170460
(1) 011276 012777 000014 170452
(1) 011304 012777 000036 170434
(1) 011312 052777 000002 170436
(1) 011320 052777 000001 170420
(1) 011326 042777 000002 170422
(1) 011334 017737 170420 001126
(1) 011342 012737 003407 001124
(1) 011350 023737 001124 001126
(3) 011356 001401
(1) 011360 104020
(1)
957
(4)
(4)
(3) 011362 000004
(2) 011364 012737 000010 001160
(1) 011372 012777 004000 170356
(1) 011400 012777 000014 170350
(1) 011406 012777 002036 170332
(1) 011414 052777 000002 170334
(1) 011422 052777 000001 170316
(1) 011430 042777 000002 170320
(1) 011436 017737 170316 001126
(1) 011444 012737 127657 001124
(1) 011452 023737 001124 001126
(3) 011460 001401
(1) 011462 104020
(1)
958
(4)
(4)
(3) 011464 000004
(2) 011466 012737 000010 001160
(1) 011474 012777 004000 170254
(1) 011502 012777 000014 170246
(1) 011510 012777 004036 170230
(1) 011516 052777 000002 170232
(1) 011524 052777 000001 170214
(1) 011532 042777 000002 170216
(1) 011540 017737 170214 001126
(1) 011546 012737 053527 001124
(1) 011554 023737 001124 001126
(3) 011562 001401
(1) 011564 104020
(1)

```
*****
*TEST 47 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 0
*****
TST47: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #36,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #3407,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST50 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 7 GAIN = 0 ZB ENABLE = 0
*****
*TEST 50 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 1 ZB ENABLE = 0
*****
TST50: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #2036,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #127657,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST51 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 7 GAIN = 1 ZB ENABLE = 0
*****
*TEST 51 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 1
*****
TST51: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #4036,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #053527,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST52 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 7 GAIN = 0 ZB ENABLE = 1
*****
```

960
(4)
(4)
(3) 011566 000004
(2) 011570 012737 000010 001160
(1) 011576 012777 004000 170152
(1) 011604 012777 000014 170144
(1) 011612 012777 000034 170126
(1) 011620 052777 000002 170130
(1) 011626 052777 000001 170112
(1) 011634 042777 000002 170114
(1) 011642 017737 170112 001126
(1) 011650 012737 001406 001124
(1) 011656 023737 001124 001126
(3) 011664 001401
(1) 011666 104020
(1)
961
(4)
(4)
(3) 011670 000004
(2) 011672 012737 000010 001160
(1) 011700 012777 004000 170050
(1) 011706 012777 000014 170042
(1) 011714 012777 002034 170024
(1) 011722 052777 000002 170026
(1) 011730 052777 000001 170010
(1) 011736 042777 000002 170012
(1) 011744 017737 170010 001126
(1) 011752 012737 053656 001124
(1) 011760 023737 001124 001126
(3) 011766 001401
(1) 011770 104020
(1)
962
(4)
(4)
(3) 011772 000004
(2) 011774 012737 000010 001160
(1) 012002 012777 004000 167746
(1) 012010 012777 000014 167740
(1) 012016 012777 004034 167722
(1) 012024 052777 000002 167724
(1) 012032 052777 000001 167706
(1) 012040 042777 000002 167710
(1) 012046 017737 167706 001126
(1) 012054 012737 125526 001124
(1) 012062 023737 001124 001126
(3) 012070 001401
(1) 012072 104020
(1)

```
*****
*TEST 52 TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 0
*****
TST52: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #34,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #1406,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST53 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 6 GAIN = 0 ZB ENABLE = 0
*****
*TEST 53 TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 1 ZB ENABLE = 0
*****
TST53: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #2034,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #53656,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST54 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 6 GAIN = 1 ZB ENABLE = 0
*****
*TEST 54 TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 1
*****
TST54: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #4034,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #125526,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST55 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 6 GAIN = 0 ZB ENABLE = 1
*****
```

```

964
(4)
(4)
(3) 012074 000004
(2) 012076 012737 000010 001160
(1) 012104 012777 004000 167644
(1) 012112 012777 000014 167636
(1) 012120 012777 000032 167620
(1) 012126 052777 000002 167622
(1) 012134 052777 000001 167604
(1) 012142 042777 000002 167606
(1) 012150 017737 167604 001126
(1) 012156 012737 000402 001124
(1) 012164 023737 001124 001126
(3) 012172 001401
(1) 012174 104020
(1)
965
(4)
(4)
(3) 012176 000004
(2) 012200 012737 000010 001160
(1) 012206 012777 004000 167542
(1) 012214 012777 000014 167534
(1) 012222 012777 002032 167516
(1) 012230 052777 000002 167520
(1) 012236 052777 000001 167502
(1) 012244 042777 000002 167504
(1) 012252 017737 167502 001126
(1) 012260 012737 025526 001124
(1) 012266 023737 001124 001126
(3) 012274 001401
(1) 012276 104020
(1)
966
(4)
(4)
(3) 012300 000004
(2) 012302 012737 000010 001160
(1) 012310 012777 004000 167440
(1) 012316 012777 000014 167432
(1) 012324 012777 004032 167414
(1) 012332 052777 000002 167416
(1) 012340 052777 000001 167400
(1) 012346 042777 000002 167402
(1) 012354 017737 167400 001126
(1) 012362 012737 052452 001124
(1) 012370 023737 001124 001126
(3) 012376 001401
(1) 012400 104020
(1)
    
```

```

*****
*TEST 55 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 0
*****
TST55: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #32,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #402,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST56 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 5 GAIN = 0 ZB ENABLE = 0
*****
*TEST 56 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 1 ZB ENABLE = 0
*****
TST56: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #2032,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #25526,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST57 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 5 GAIN = 1 ZB ENABLE = 0
*****
*TEST 57 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 1
*****
TST57: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #4032,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #52452,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST60 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 5 GAIN = 0 ZB ENABLE = 1
    
```

968
 (4)
 (4)
 (3) 012402 000004
 (2) 012404 012737 000010 001160
 (1) 012412 012777 004000 167336
 (1) 012420 012777 000014 167330
 (1) 012426 012777 000030 167312
 (1) 012434 052777 000002 167314
 (1) 012442 052777 000001 167276
 (1) 012450 042777 000002 167300
 (1) 012456 017737 167276 001126
 (1) 012464 012737 000202 001124
 (1) 012472 023737 001124 001126
 (3) 012500 001401
 (1) 012502 104020

```

*****
*TEST 60 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 0
*****
TST60: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #30,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #202,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST61 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 4 GAIN = 0 ZB ENABLE = 0

```

969
 (4)
 (4)
 (3) 012504 000004
 (2) 012506 012737 000010 001160
 (1) 012514 012777 004000 167234
 (1) 012522 012777 000014 167226
 (1) 012530 012777 002030 167210
 (1) 012536 052777 000002 167212
 (1) 012544 052777 000001 167174
 (1) 012552 042777 000002 167176
 (1) 012560 017737 167174 001126
 (1) 012566 012737 012726 001124
 (1) 012574 023737 001124 001126
 (3) 012602 001401
 (1) 012604 104020

```

*****
*TEST 61 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 1 ZB ENABLE = 0
*****
TST61: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #2030,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #012726,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST62 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 4 GAIN = 1 ZB ENABLE = 0

```

970
 (4)
 (4)
 (3) 012606 000004
 (2) 012610 012737 000010 001160
 (1) 012616 012777 004000 167132
 (1) 012624 012777 000014 167124
 (1) 012632 012777 004030 167106
 (1) 012640 052777 000002 167110
 (1) 012646 052777 000001 167072
 (1) 012654 042777 000002 167074
 (1) 012662 017737 167072 001126
 (1) 012670 012737 025252 001124
 (1) 012676 023737 001124 001126
 (3) 012704 001401
 (1) 012706 104020

```

*****
*TEST 62 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 1
*****
TST62: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #4030,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #025252,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST63 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 4 GAIN = 0 ZB ENABLE = 1

```

972
(4)
(4)
(3) 012710 000004
(2) 012712 012737 000010 001160
(1) 012720 012777 004000 167030
(1) 012726 012777 000014 167022
(1) 012734 012777 000026 167004
(1) 012742 052777 000002 167006
(1) 012750 052777 000001 166770
(1) 012756 042777 000002 166772
(1) 012764 017737 166770 001126
(1) 012772 012737 000000 001124
(1) 013000 023737 001124 001126
(3) 013006 001401
(1) 013010 104020
(1)

```
*****  
*TEST 63 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 0  
*****  
TST63: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #26,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #0,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST64 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 3 GAIN = 0 ZB ENABLE = 0
```

973
(4)
(4)
(3) 013012 000004
(2) 013014 012737 000010 001160
(1) 013022 012777 004000 166726
(1) 013030 012777 000014 166720
(1) 013036 012777 002026 166702
(1) 013044 052777 000002 166704
(1) 013052 052777 000001 166666
(1) 013060 042777 000002 166670
(1) 013066 017737 166666 001126
(1) 013074 012737 005252 001124
(1) 013102 023737 001124 001126
(3) 013110 001401
(1) 013112 104020
(1)

```
*****  
*TEST 64 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 1 ZB ENABLE = 0  
*****  
TST64: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #2026,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #5252,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST65 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 3 GAIN = 1 ZB ENABLE = 0
```

974
(4)
(4)
(3) 013114 000004
(2) 013116 012737 000010 001160
(1) 013124 012777 004000 166624
(1) 013132 012777 000014 166616
(1) 013140 012777 004026 166600
(1) 013146 052777 000002 166602
(1) 013154 052777 000001 166564
(1) 013162 042777 000002 166566
(1) 013170 017737 166564 001126
(1) 013176 012737 012424 001124
(1) 013204 023737 001124 001126
(3) 013212 001401
(1) 013214 104020
(1)

```
*****  
*TEST 65 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 1  
*****  
TST65: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #4026,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #12424,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST66 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 3 GAIN = 0 ZB ENABLE = 1
```

976
(4)
(4)
(3) 013216 000004
(2) 013220 012737 000010 001160
(1) 013226 012777 004000 166522
(1) 013234 012777 000014 166514
(1) 013242 012777 000024 166476
(1) 013250 052777 000002 166500
(1) 013256 052777 000001 166462
(1) 013264 042777 000002 166464
(1) 013272 017737 166462 001126
(1) 013300 012737 000000 001124
(1) 013306 023737 001124 001126
(3) 013314 001401
(1) 013316 104020
(1)
977
(4)
(4)
(3) 013320 000004
(2) 013322 012737 000010 001160
(1) 013330 012777 004000 166420
(1) 013336 012777 000014 166412
(1) 013344 012777 002024 166374
(1) 013352 052777 000002 166376
(1) 013360 052777 000001 166360
(1) 013366 042777 000002 166362
(1) 013374 017737 166360 001126
(1) 013402 012737 002552 001124
(1) 013410 023737 001124 001126
(3) 013416 001401
(1) 013420 104020
(1)
978
(4)
(4)
(3) 013422 000004
(2) 013424 012737 000010 001160
(1) 013432 012777 004000 166316
(1) 013440 012777 000014 166310
(1) 013446 012777 004024 166272
(1) 013454 052777 000002 166274
(1) 013462 052777 000001 166256
(1) 013470 042777 000002 166260
(1) 013476 017737 166256 001126
(1) 013504 012737 005224 001124
(1) 013512 023737 001124 001126
(3) 013520 001401
(1) 013522 104020
(1)

```
*****  
*TEST 66 TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 0  
*****  
TST66: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #24,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #0,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST67 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 2 GAIN = 0 ZB ENABLE = 0  
*****  
*TEST 67 TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 1 ZB ENABLE = 0  
*****  
TST67: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #2024,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #2552,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST70 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 2 GAIN = 1 ZB ENABLE = 0  
*****  
*TEST 70 TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 1  
*****  
TST70: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"  
MOV #4024,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER  
MOV #5224,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ  
BEQ TST71 ;;BR IF SAME  
ERROR 20 ;INCORRECT ADDRESS MAKER DATA  
RESOLUTION = 2 GAIN = 0 ZB ENABLE = 1
```

980
(4)
(4)
(3) 013524 000004
(2) 013526 012737 000010 001160
(1) 013534 012777 004000 166214
(1) 013542 012777 000014 166206
(1) 013550 012777 000022 166170
(1) 013556 052777 000002 166172
(1) 013564 052777 000001 166154
(1) 013572 042777 000002 166156
(1) 013600 017737 166154 001126
(1) 013606 012737 000000 001124
(1) 013614 023737 001124 001126
(3) 013622 001401
(1) 013624 104020
(1)
981
(4)
(4)
(3) 013626 000004
(2) 013630 012737 000010 001160
(1) 013636 012777 004000 166112
(1) 013644 012777 000014 166104
(1) 013652 012777 002022 166066
(1) 013660 052777 000002 166070
(1) 013666 052777 000001 166052
(1) 013674 042777 000002 166054
(1) 013702 017737 166052 001126
(1) 013710 012737 001265 001124
(1) 013716 023737 001124 001126
(3) 013724 001401
(1) 013726 104020
(1)
982
(4)
(4)
(3) 013730 000004
(2) 013732 012737 000010 001160
(1) 013740 012777 004000 166010
(1) 013746 012777 000014 166002
(1) 013754 012777 004022 165764
(1) 013762 052777 000002 165766
(1) 013770 052777 000001 165750
(1) 013776 042777 000002 165752
(1) 014004 017737 165750 001126
(1) 014012 012737 002512 001124
(1) 014020 023737 001124 001126
(3) 014026 001401
(1) 014030 104020
(1)

```
*****
*TEST 71 TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 0
*****
TST71: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #22,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #0,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST72 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 1 GAIN = 0 ZB ENABLE = 0
*****
*TEST 72 TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 1 ZB ENABLE = 0
*****
TST72: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #2022,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #1265,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST73 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 1 GAIN = 1 ZB ENABLE = 0
*****
*TEST 73 TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 1
*****
TST73: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET "TEST DMA AND CONTROL"
MOV #4022,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #2512,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST74 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 1 GAIN = 0 ZB ENABLE = 1
*****
```


984
(3)
(3)
(2) 014032 000004
(1) 014034 012737 000010 001160
985 014042 012777 004000 165706
986 014050 012777 000014 165700
987 014056 012777 000000 165662
988 014064 052777 000002 165664
989 014072 052777 000001 165646
990 014100 042777 000002 165650
991 014106 017737 165646 001126
992 014114 012737 003407 001124
993 014122 023737 001124 001126
994 014130 001401
995 014132 104021
996
997
998
(3)
(3)
(2) 014134 000004
(1) 014136 012737 000010 001160
999 014144 012777 004000 165604
1000 014152 012777 000014 165576
1001 014160 012777 004000 165560
1002 014166 052777 000002 165562
1003 014174 052777 000001 165544
1004 014202 042777 000002 165546
1005 014210 017737 165544 001126
1006 014216 012737 052452 001124
1007 014224 023737 001124 001126
1008 014232 001401
1009 014234 104021
1010
1011
1012
(3)
(3)
(2) 014236 000004
(1) 014240 012737 000010 001160
1013 014246 012777 004000 165502
1014 014254 012777 000014 165474
1015 014262 012777 002000 165456
1016 014270 052777 000002 165460
1017 014276 052777 000001 165442
1018 014304 042777 000002 165444
1019 014312 017737 165442 001126
1020 014320 012737 127657 001124
1021 014326 023737 001124 001126
1022 014334 001401
1023 014336 104021
1024

```
*****
:*TEST 74 TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 0
*****
TST74: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET TEST "DMA AND CONTROL"
MOV #0,@CSR ;ENSURE LIST MODE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS MAKER VALUE
MOV #3407,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST75 ;;BR IF SAME
ERROR 21 ;INCORRECT ADDRESS MAKER DATA
;RESOLUTION 7 <DEFAULT WHEN ZB IS NOT ENABLED>
```

```
*****
:*TEST 75 TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 1 GAIN = 0
*****
TST75: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET TEST "DMA AND CONTROL"
MOV #4000,@CSR ;ENSURE LIST MODE AND ZB ENABLED
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS MAKER VALUE
MOV #52452,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST76 ;;BR IF SAME
ERROR 21 ;INCORRECT ADDRESS MAKER DATA
;RESOLUTION 5 <DEFAULT WHEN ZB IS ENABLED>
```

```
*****
:*TEST 76 TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 1
*****
TST76: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET TEST "DMA AND CONTROL"
MOV #2000,@CSR ;SET GAIN FLOP
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BITO,@CSR ;ENABLE THE NCV11
BIC #TESTZ,@SFR ;DISABLE THE "TEST Z" PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS MAKER VALUE
MOV #127657,$GDDAT ;LOAD THE EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST77 ;;BR IF SAME
ERROR 21 ;INCORRECT ADDRESS MAKER DATA
;RESOLUTION 7 - GAIN FLOP SET
```

```

1029 (3)
(3)
(2) 014340 000004
(1) 014342 012737 000040 001160
1030 014350 012777 004000 165400
1031 014356 012737 125252 060000
1032 014364 012777 011110 165356
1033 014372 012777 177777 165352
1034 014400 012777 060000 165346
1035 014406 012777 000014 165342
1036 014414 052777 000001 165324
1037 014422 052777 000002 165326
(1) 014430 042777 000002 165320
1038 014436 000240
1039 014440 000240
1040 014442 000240
1041 014444 052777 010000 165304
1042 014452 000240
1043 014454 000240
1044 014456 000240
1045 014460 017737 165262 001126
1046 014466 012777 004000 165262
1047 014474 012737 040200 001124
1048 014502 023737 001124 001126
1049 014510 001402
1050 014512 104036
1051 014514 000453
1052 014516 017737 165232 001126 4$:
1053 014524 012737 060002 001124
1054 014532 023737 001124 001126
1055 014540 001401
1056 014542 104022
1057
1058
1059 014544 017737 165202 001126 1$:
1060 014552 012737 000000 001124
1061 014560 023737 001124 001126
1062 014566 001401
1063 014570 104023
1064
1065 014572 005037 001124 2$:
1066 014576 017737 165146 001126
1067 014604 001401
1068 014606 104024
1069
1070 014610 013737 060000 001126 3$:
1071 014616 012737 003407 001124
1072 014624 012737 060000 001122
1073 014632 023737 001124 001126
1074 014640 001401
1075 014642 104037

*****
*TEST 77 ENABLE A ONE WORD TRANSFER SECTION LIST MODE
*****
TST77: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #125252,BUFO ;PRIME TARGET BUFFER
MOV #11110,@OFF ;LOAD THE OFFSET VALUE WITH A NUMBER
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
BIS #BIT0,@CSR ;ENABLE DEVICE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
NOP
NOP
NOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
NCP
MOV @CSR,$BDDAT ;READ STATUS
MOV #CLRALL,@SFR ;RESET THE DEVICE
MOV #40200,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;TEST STATUS
BEQ 4$ ;;BR IF EXPECTED
ERROR 36 ;UNEXPECTED STATUS AFTER A 1 WORD TRANSFER
BR TST100 ;;
MOV @BAR,$BDDAT ;READ BUS ADDRESS
MOV #BUFO+2,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 1$ ;;BR IF SAME
ERROR 22 ;INCORRECT BUS ADDRESS VALUE
;AFTER A 1 WORD TRANSFER
MOV @WCR,$BDDAT ;READ W.C. REGISTER
MOV #0,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 2$ ;;BR IF SAME
ERROR 23 ;INCORRECT WORD COUNT REGISTER VALUE
;AFTER A 1 WORD TRANSFER
CLR $GDDAT ;CLEAR THE EXPECTED VALUE
MOV @OFF,$BDDAT ;READ THE OFFSET REGISTER
BEQ 3$ ;;BR IF CLEARED
ERROR 24 ;OFFSET REG. FAILED TO CLEAR AFTER
;1 LIST MODE XFR.
MOV BUFO,$BDDAT ;GET BUFFER DATA
MOV #3407,$GDDAT ;LOAD EXPECTED
MOV #BUFO,$BDDADR ;LOAD BAD ADDRESS
CMP $GDDAT,$BDDAT ;COMPARE DATA
BEQ TST100 ;;BR IF SAME
ERROR 37 ;STATUS WAS OK BUT DATA WAS INCORRECT

```

```

1077      ::*****
(3)      ::*TEST 100      ENABLE A 512 WORD TRANSFER      SECTION LIST MODE
(3)      ::*****
(2) 014644 000004      TST100: SCOPE
(1) 014646 012737 000040 001160      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
1078 014654 012777 004000 165074      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
1079 014662 012700 060000      MOV      #BUFO,R0      ;LOAD BUFFER POINTER
1080 014666 012720 125252      1$: MOV      #125252,(R0)+      ;PRESET THE BUFFER WITH DATA
1081 014672 020027 062000      CMP      R0,#BUF1      ;TEST IF DONE
1082 014676 001373      BNE      1$      ;BR IF NOT
1083 014700 012777 177000 165044      MOV      #-512.,@WCR      ;SET UP 512. WORD TRANSFER
1084 014706 012777 060000 165040      MOV      #BUFO,@BAR      ;LOAD BUS ADDRESS FOR RESULT
1085 014714 012777 000014 165034      MOV      #TSTDMA!TSTCON,@SFR      ;ENABLE TEST CONTROL AND DMA FLOPS
1086 014722 052777 000001 165016      BIS      #BIT0,@CSR      ;ENABLE DEVICE
1087 014730 012737 001000 002004      MOV      #512.,$TEMP      ;LOAD THE COUNTER
1088 014736      2$:
(1) 014736 052777 000002 165012      BIS      #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES
(1) 014744 042777 000002 165004      BIC      #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES
1089 014752 052777 010000 164776      BIS      #BIT12,@SFR      ;ALLOW 1 DMA TRANSFER
1090 014760 005337 002004      DEC      $TEMP      ;FINISHED ALL WORDS?
1091 014764 001364      BNE      2$      ;BR UNTILL DONE
1092      ;THE TRANSFER IS NOW COMPLETE
1093 014766 017737 164754 001126      MOV      @CSR,$BDDAT      ;READ STATUS
1094 014774 012737 040200 001124      MOV      #40200,$GDDAT      ;LOAD EXPECTED STATUS
1095 015002 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA
1096 015010 001402      BEQ      3$      ;;BR IF EXPECTED STATUS
1097 015012 104036      ERROR    36      ;UNEXPECTED STATUS AFTER 512 WORD TRANSFER
1098 015014 000465      BR      TST101      ;;
1099 015016 005037 001124      3$: CLR      $GDDAT      ;CLEAR EXPECTED
1100 015022 017737 164722 001126      MOV      @OFF,$BDDAT      ;READ OFFSET REG.
1101 015030 001401      BEQ      4$      ;;BR IF CLEARED
1102 015032 104006      ERROR    6      ;UNEXPECTED OFFSET REGISTER BIT SET
1103 015034 012777 004000 164714      4$: MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
1104 015042 017737 164706 001126      MOV      @BAR,$BDDAT      ;READ BUS ADDRESS
1105 015050 012737 062000 001124      MOV      #BUF1,$GDDAT      ;LOAD EXPECTED BAR VALUE
1106 015056 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
1107 015064 001401      BEQ      5$      ;;BR IF SAME
1108 015066 104022      ERROR    22      ;INCORRECT BUS ADDRESS VALUE AFTER 1 WORD TRANSFER
1109
1110 015070 017737 164656 001126      5$: MOV      @WCR,$BDDAT      ;READ W.C. REGI.
1111 015076 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED W.C. VALUE
1112 015104 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
1113 015112 001401      BEQ      6$      ;;BR IF SAME
1114 015114 104023      ERROR    23      ;INCORRECT WORD COUNT REGISTER VALUE AFTER A 1 WORD TRANSFER
1115 015116 012737 003407 001124      6$: MOV      #3407,$GDDAT      ;LOAD EXPECTED DATA
1116 015124 012737 060000 001122      MOV      #BUFO,$BDADR      ;LOAD STARTING ADDRESS
1117 015132 017737 163764 001126      7$: MOV      @BDADR,$BDDAT      ;READ DATA WORD
1118 015140 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA
1119 015146 001401      BEQ      10$      ;;BR IF EXPECTED
1120 015150 104037      ERROR    37      ;INCORRECT DATA IN LIST MODE XFER.
1121 015152 062737 000002 001122      10$: ADD      #2,$BDADR      ;UPDATE POINTER
1122 015160 022737 062000 001122      CMP      #BUFO+1024.,$BDADR      ;TEST IF END OF BUFFER
1123 015166 001361      BNE      7$      ;;BR IF NOT DONE

```

1125
(3)
(3)
(2) 015170 000004
(1) 015172 012737 000040 001160
1126 015200 012777 004000 164550
1127 015206 012777 177777 164536
1128 015214 012777 000003 164526
1129 015222 012777 160000 164524
1130 015230 012777 000014 164520
1131 015236 012777 000016 164502
1132 015244 052777 000001 164474
1133 015252 052777 000002 164476
(1) 015260 042777 000002 164470
1134 015266 000240
1135 015270 000240
1136 015272 000240
1137 015274 052777 010000 164454
1138 015302 000240
1139 015304 000240
1140 015306 000240
1141 015310 012737 140200 001124
1142 015316 017737 164424 001126
1143 015324 100402
1144 015326 104025
1145 015330 000423
1146 015332 023737 001124 001126 1\$:
1147 015340 001401
1148 015342 104025
1149 015344 052777 004000 164404 2\$:
1150 015352 012737 000200 001124
1151 015360 017737 164362 001126
1152 015366 023737 001124 001126
1153 015374 001401
1154 015376 104025
1155
(3)
(3)
(2) 015400 000004
(1) 015402 012737 000040 001160
1156 015410 012777 004000 164340
1157 015416 012777 177777 164326
1158 015424 012777 000003 164316
1159 015432 012777 160000 164314
1160 015440 012777 000014 164310
1161 015446 052777 000001 164272
1162 015454 052777 000002 164274
(1) 015462 042777 000002 164266
1163 015470 000240
1164 015472 000240
1165 015474 000240
1166 015476 052777 010000 164252
1167 015504 000240
1168 015506 000240
1169 015510 000240
1170 015512 012737 140200 001124

```
*****  
*TEST 101 VERIFY "TIMEOUT" FLOP SETS AND "CLR ALL" CLEARS IT  
*****  
TST101: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #-1,@WCR ;LOAD W.C. REGISTER  
MOV #3,@OFF ;LOAD EXTENDED ADDRESS BITS  
MOV #160000,@BAR ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDRE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA  
MOV #16,@CSR ;LOAD RESOLUTION TO VERIFY "SFR INIT" CLEARS  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV #BIT15!BIT14!BIT7,$GDDAT ;LOAD EXPECTED  
MOV @CSR,$BDDAT ;READ STATUS REG.  
BMI 1$ ;BR IF "TIMEOUT" FLOP IS SET  
ERROR 2$ ;"TIMEOUT" FLOP FAILED TO SET  
BR TST102 ;;  
1$: CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 2$ ;;BR IF SAME  
ERROR 2$ ;"TIMEOUT" FLOP DID SET BUT FAILED TO GENERATE ""  
2$: BIS #CLRALL,@SFR ;GENERATE AN "CLR ALL" TO CLEAR TIMEOUT FLOP  
MOV #BIT7,$GDDAT ;LOAD EXPECTED  
MOV @CSR,$BDDAT ;READ STATUS  
CMP $GDDAT,$BDDAT ;COMPARE VALUE  
BEQ TST102 ;;BR IF SAME  
ERROR 2$ ;"CLR ALL" FAILED TO CLEAR TIMEOUT FLOP  
*****  
*TEST 102 VERIFY "TIMEOUT" FLOP SETS AND "CLR TIMEOUT" CLEARS IT  
*****  
TST102: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #-1,@WCR ;LOAD W.C. REGISTER  
MOV #3,@OFF ;LOAD EXTENDED ADDRESS BITS  
MOV #160000,@BAR ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDRE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV #BIT15!BIT14!BIT7,$GDDAT ;LOAD EXPECTED
```

```

1171 015520 017737 164222 001126      MOV    @CSR,$BDDAT      ;READ STATUS REG.
1172 015526 100402          BMI    1$              ;BR IF "TIMEOUT" FLOP IS SET
1173 015530 104025          ERROR  25              ;"TIMEOUT" FLOP FAILED TO SET
1174 015532 000423          BR     TST103          ;;
1175 015534 023737 001124 001126 1$:  CMP    $GDDAT,$BDDAT   ;COMPARE VALUES
1176 015542 001401          BEQ    2$              ;;BR IF SAME
1177 015544 104025          ERROR  25              ;"TIMEOUT" FLOP DID SET BUT FAILED TO GENERATE ""
1178 015546 052777 100000 164202 2$:  BIS    #BIT15,@SFR     ;GENERATE AN "CLR TIMEOUT" TO CLEAR TIMEOUT FLOP
1179 015554 012737 040200 001124      MOV    #BIT14!BIT7,$GDDAT ;LOAD EXPECTED
1180 015562 017737 164160 001126      MOV    @CSR,$BDDAT     ;READ STATUS
1181 015570 023737 001124 001126      CMP    $GDDAT,$BDDAT   ;COMPARE VALUE
1182 015576 001401          BEQ    TST103          ;;BR IF SAME
1183 015600 104025          ERROR  25              ;"CLR TIMEOUT" FAILED TO CLEAR TIMEOUT FLOP
1184          ;;*****
(3)          ;*TEST 103  VERIFY "TIMEOUT" INTERRUPT
(3)          ;;*****
(2) 015602 000004          TST103: SCOPE
(1) 015604 012737 000040 001160      MOV    #40,$TIMES      ;;DO 40 ITERATIONS
1185 015612 012777 004000 164136      MOV    #CLRALL,@SFR    ;CLEAR DEVICE
1186 015620 012777 177777 164124      MOV    #-1,@WCR        ;LOAD W.C. REGISTER
1187 015626 012777 000003 164114      MOV    #3,@OFF         ;LOAD EXTENDED ADDRESS BITS
1188 015634 012777 160000 164112      MOV    #160000,@BAR    ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDRE
1189 015642 012777 000014 164106      MOV    #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA
1190 015650 012746 000000          MOV    #0,-(SP)
1191 015654 012746 015662          MOV    #1$,-(SP)
1192 015660 000002          RTI
1193 015662 012777 015754 164102 1$:  MOV    #2$,@VECTA0
1194 015670 000240          NOP
1195 015672 000240          NOP
1196 015674 000240          NOP
1197 015676 000240          NOP
1198 015700 052777 000101 164040      BIS    #BIT6!BIT0,@CSR ;ENABLE THE DEVICE
1199 015706 052777 000002 164042      BIS    #TESTZ,@SFR     ;ENABLE "TEST Z" PULSES
(1) 015714 042777 000002 164034      BIC    #TESTZ,@SFR     ;DISABLE "TEST Z" PULSES
1200 015722 000240          NOP
1201 015724 000240          NOP
1202 015726 000240          NOP
1203 015730 052777 010000 164020      BIS    #BIT12,@SFR     ;ALLOW 1 DMA TRANSFER
1204 015736 000240          NOP
1205 015740 000240          NOP
1206 015742 000240          NOP
1207 015744 005077 163776          CLR    @CSR            ;CLEAR ENABLE
1208 015750 104026          ERROR  26              ;"TIMEOUT" FAILED TO INTERRUPT
1209 015752 000401          BR     3$              ;;BR TO CLEAN UP
1210
1211 015754 022626          2$:  CMP    (SP)+,(SP)+     ;CLEAN THE STACK
1212 015756 005077 163764          3$:  CLR    @CSR
1213 015762 012777 004000 163766      MOV    #CLRALL,@SFR    ;CLEAR THE DEVICE
1214 015770 013777 001774 163774      MOV    VECTA1,@VECTA0 ;RESET VECTOR
1215 015776 005077 163772          CLR    @VECTA1
  
```

1217
(3)
(3)
(2) 016002 000004
(1) 016004 012737 000040 001160
1218 016012 012777 004000 163736
1219 016020 023727 036406 002140
1220 016026 103450
1221 016030 012777 177777 163714
1222 016036 012777 177776 163710
1223 016044 005077 163700
1224 016050 012777 000014 163700
1225 016056 012777 000016 163662
1226 016064 052777 000001 163654
1227 016072 052777 000002 163656
(1) 016100 042777 000002 163650
1228 016106 052777 010000 163642
1229 016114 000240
1230 016116 000240
1231 016120 000240
1232 016122 012737 000001 001124
1233 016130 017737 163614 001126
1234 016136 023737 001124 001126
1235 016144 001401
1236 016146 104006
1237
1238
(3)
(3)
(2) 016150 000004
(1) 016152 012737 000040 001160
1239 016160 012777 004000 163570
1240 016166 023727 036406 006140
1241 016174 103454
1242 016176 012777 177777 163546
1243 016204 012777 177776 163542
1244 016212 012777 000001 163530
1245 016220 012777 000014 163530
1246 016226 012777 000016 163512
1247 016234 052777 000001 163504
1248 016242 052777 000002 163506
(1) 016250 042777 000002 163500
1249 016256 000240
1250 016260 000240
1251 016262 000240
1252 016264 052777 010000 163464
1253 016272 000240
1254 016274 000240
1255 016276 000240
1256 016300 012737 000002 001124
1257 016306 017737 163436 001126
1258 016314 023737 001124 001126
1259 016322 001401
1260 016324 104006

```
::*****  
:*TEST 104 VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 16  
:*****  
TST104: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CMP $LSTBK,#2140 ;TEST IF ENOUGH MEMORY  
BLO TST105 ;;BR IF NOT ENOUGH MEMORY  
MOV #-1,@WCR ;LOAD 1 WORD XFR  
MOV #177776,@BAR ;LOAD LAST ADDRESS  
CLR @OFF ;ENSURE CLEARED EXTENDED ADDRESS BITS  
MOV #TSTCON!TSTDMA,@SFR ;ENABLE CONTROLLER  
MOV #16,@CSR ;ENABLE THE MODE  
BIS #BIT0,@CSR ;ENABLE THE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
NOP  
MOV #1,$GDDAT ;LOAD EXPECTED VALUE  
MOV @OFF,$BDDAT ;READ ACUTAL VALUE  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST105 ;;BR IF SAME  
ERROR 6 ;EXTENDED ADDRESS BIT 16 FAILED TO SET
```

```
::*****  
:*TEST 105 VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 17  
:*****  
TST105: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CMP $LSTBK,#6140 ;TEST MEMORY SPACE >100K  
BLO TST106 ;;BR IF NOT ENOUGH MEMORY  
MOV #-1,@WCR ;LOAD 1 WORD XFR  
MOV #177776,@BAR ;LOAD LAST ADDRESS  
MOV #1,@OFF ;LOAD EXTENDED ADDRESS BIT  
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL  
MOV #16,@CSR ;ENABLE THE MODE  
BIS #BIT0,@CSR ;ENAVLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV #2,$GDDAT ;LOAD EXPECTED  
MOV @OFF,$BDDAT ;READ ACTUAL  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST106 ;;BR IF SAME  
ERROR 6 ;EXTENDED ADDRESS BIT 17 FAILED TO SET
```

```

1262
(3)
(3)
(2) 016326 000004
(1) 016330 012737 000040 001160
1263 016336 012777 004000 163412
1264 016344 012777 177777 163400
1265 016352 012777 060000 163374
1266 016360 012777 000014 163370
1267 016366 012737 007070 060000
1268 016374 052777 000001 163344
1269 016402 000240
1270 016404 000240
1271 016406 000240
1272 016410 052777 002000 163340
1273 016416 052777 010000 163332
1274 016424 000240
1275 016426 000240
1276 016430 000240
1277 016432 013737 060000 001126
1278 016440 012737 177777 001124
1279 016446 023737 001124 001126
1280 016454 001401
1281 016456 104027
1282
1283
(3)
(3)
(2) 016460 000004
(1) 016462 012737 000040 001160
1284 016470 012777 004000 163260
1285 016476 012777 177777 163246
1286 016504 012777 060000 163242
1287 016512 012777 000014 163236
1288 016520 012737 007070 060000
1289 016526 052777 000001 163212
1290 016534 000240
1291 016536 000240
1292 016540 000240
1293 016542 052777 001000 163206
1294 016550 052777 010000 163200
1295 016556 000240
1296 016560 000240
1297 016562 000240
1298 016564 013737 060000 001126
1299 016572 012737 000000 001124
1300 016600 023737 001124 001126
1301 016606 001401
1302 016610 104027
1303
  
```

```

*****
*TEST 106 VERIFY "SET EVENT" DATA GENERATES A 17777 DATA WORD
*****
TST106: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUFO ;PRESET DATA
BIS #BIT0,@CSR ;ENABLE DEVICE
NOP
NOP
BIS #BIT10,@SFR ;SET "EVENT" FLOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUFO,$BDDAT ;READ DATA
MOV #-1,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ TST107 ;;BR IF SAME
ERROR 27 ;INCORRECT DATA VALUE FOR "EVENT" MARK
;AFTER A 1 WORD TRANSFER

*****
*TEST 107 VERIFY "SET TIME" DATA GENERATES A 00000 DATA WORD
*****
TST107: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUFO ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE DEVICE
NOP
NOP
BIS #BIT9,@SFR ;SET "TIME" FLOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUFO,$BDDAT ;READ DATA
MOV #0,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ TST110 ;;BR IF SAME
ERROR 27 ;INCORRECT DATA VALUE FOR "TIME" MARK
;AFTER A 1 WORD TRANSFER
  
```

```
1305
(3)
(3)
(2) 016612 000004
(1) 016614 012737 000040 001160
1306 016622 012777 004000 163126
1307 016630 005737 004020
1308 016634 001044
1309 016636 012777 177777 163106
1310 016644 012777 060000 163102
1311 016652 012777 000014 163076
1312 016660 012737 007070 060000
1313 016666 052777 000001 163052
1314 016674 000240
1315 016676 000240
1316 016700 052777 000400 163032
1317 016706 052777 010000 163042
1318 016714 000240
1319 016716 000240
1320 016720 013737 060000 001126
1321 016726 012737 177777 001124
1322 016734 023737 001124 001126
1323 016742 001401
1324 016744 104040
1325
1326
(3)
(3)
(2) 016746 000004
(1) 016750 012737 000040 001160
1327 016756 012777 004000 162772
1328 016764 005737 004020
1329 016770 001052
1330 016772 012777 177777 162752
1331 017000 012777 060000 162746
1332 017006 012777 000014 162742
1333 017014 012737 007070 060000
1334 017022 052777 000001 162716
1335 017030 005077 162704
1336 017034 012777 177776 162702
1337 017042 012777 000011 162670
1338 017050 105777 162664 1$: TSTB
1339 017054 100375 BPL 1$
1340 017056 052777 010000 162672 BIS #BIT12,@SFR
1341 017064 000240 NOP
1342 017066 000240 NOP
1343 017070 013737 060000 001126 MOV BUFO,$BDDAT
1344 017076 012737 000000 001124 MOV #0,$GDDAT
1345 017104 023737 001124 001126 CMP $GDDAT,$BDDAT
1346 017112 001401 BEQ TST112
1347 017114 104040 ERROR 40
1348
```

```
*****
*TEST 110 VERIFY 'CLOCK ST1' GENERATES A EVENT (177777) DATA WORD
*****
TST110: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
TST DEADKW ;TEST IF NCV11 CLOCK IS PRESENT
BNE TST111 ;;BR IF NOT
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUFO ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE THE DEVICE
NOP
NOP
BIS #BIT8,@KWCSR ;GENERATE CLOCK ST1 TO SET 'EVENT' FLOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUFO,$BDDAT ;READ BUFFER DATA
MOV #-1,$GDDAT ;LOAD EXPECTED DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST111 ;;BR IF EXPECTED
ERROR 40 ;CLOCK ST1 FAILED TO GENERATE EVENT FLAG
;CHECK THE M8026 TO M7952 JUMPERS
*****
*TEST 111 VERIFY 'CLOCK OVERFLOW' GENERATES A TIME (000000) DATA WORD
*****
TST111: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
TST DEADKW ;TEST IF NCV11 CLOCK IS PRESENT
BNE TST112 ;;BR IF NOT
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUFO ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE THE DEVICE
CLR @KWCSR ;CLEAR STATUS
MOV #-2,@KWPSR ;LOAD COUNTER PRESET
MOV #11,@KWCSR ;ENABLE 1 MHZ. RATE AND CLOCK GO
TSTB @KWCSR ;WAIT FOR CLOCK
BPL 1$
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUFO,$BDDAT ;READ DATA
MOV #0,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ TST112 ;;BR IF SAME
ERROR 40 ;CLOCK OVERFLOW FAILED TO GENERATE 'TIME MARKS'
;CHECK THE M8026 TO M7952 JUMPERS
```


1350
(3)
(3)
(2) 017116 000004
(1) 017120 012737 000040 001160
1351 017126 012777 004000 162622
1352 017134 012777 177777 162610
1353 017142 012777 060000 162604
1354 017150 012777 000014 162600
1355 017156 012737 007070 060000
1356 017164 052777 000001 162554
1357 017172 000240
1358 017174 000240
1359 017176 000240
1360 017200 052777 003000 162550
1361 017206 052777 010000 162542
1362 017214 000240
1363 017216 000240
1364 017220 000240
1365 017222 013737 060000 001126
1366 017230 012737 000000 001124
1367 017236 023737 001124 001126
1368 017244 001401
1369 017246 104027
1370
(3)
(3)
(2) 017250 000004
(1) 017252 012737 000040 001160
1371 017260 012777 004000 162470
1372 017266 005037 060000
1373 017272 012777 060000 162450
1374 017300 012777 000022 162440
1375 017306 012777 000014 162442
1376 017314 052777 000001 162424
1377 017322 052777 000002 162426
(1) 017330 042777 000002 162420
1378 017336 000240
1379 017340 000240
1380 017342 000240
1381 017344 052777 010000 162404
1382 017352 000240
1383 017354 000240
1384 017356 000240
1385 017360 013737 060000 001126
1386 017366 012737 000001 001124
1387 017374 023737 001124 001126
1388 017402 001401
1389 017404 104030
1390
1391

```
*****  
*TEST 112 VERIFY "SET TIME" OVERRIDES "SET EVENT" DATA  
*****  
TST112: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER  
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT  
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS  
MOV #7070,BUFO ;PRESET THE DATA  
BIS #BIT0,@CSR ;ENABLE DEVICE  
NOP  
NOP  
NOP  
BIS #BIT10!BIT9,@SFR ;SET "TIME AND EVENT" FLOPS  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV BUFO,$BDDAT ;READ DATA  
MOV #0,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ TST113 ;;BR IF SAME  
ERROR 27 ;"TIME" MARK FAILED TO OVERRIDE "EVENT" OR DATA M  
*****  
*TEST 113 DO A ONE WORD MATRIX MODE TRANSFER -CHECK FOR INCREMENT FUNCTION  
*****  
TST113: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CLR BUFO ;CLEAR INITIAL INCREMENT LOCATION  
MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
MOV #BIT4!BIT1,@CSR ;ENABLE MATRIX MODE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
MOV BUFO,$BDDAT ;READ THE BUFO LOCATION  
MOV #1,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ TST114 ;;BR IF SAME  
ERROR 30 ;INCORRECT DATA IN MATRIX MODE  
;IF DATA WAS 0, THE ADDRESS ACCESSED WAS PROBABLY WRONG  
;IF DATA WAS NON ZERO, THE "INCR" REGISTER IS STUCK TO A 1
```

```

1393 .....
(3) *TEST 114 VERIFY EACH BIT OF THE INCREMENT REG. DATA PATH
(3) .....
(2) 017406 000004 TST114: SCOPE
(1) 017410 012737 000040 001160 MOV #40,$TIMES ;;DO 40 ITERATIONS
1394 ;CHECK FOR A "CARRY" INTO EACH BIT
1395 IE: 000002-000001
1396 000004-000003
1397 000010-000007
1398 000020-000017
1399 000040-000037
1400 000100-000077
1401 000200-000177
1402 ETC
1403 017416 012737 017432 001110 MOV #1$,$LPERR ;LOAD RETURN ADDRESS IF ERROR
1404 017424 012737 000002 002004 MOV #2,$TEMP ;LOAD INITIAL VALUE
1405
1406 017432 012777 004000 162316 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1407 017440 012777 060000 162302 MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER
1408 017446 012777 000024 162272 MOV #BIT4!BIT2,@CSR ;ENABLE WORD MATRIX MODE
1409 017454 012777 000014 162274 MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
1410 017462 052777 000001 162256 BIS #BIT0,@CSR ;ENABLE THE DEVICE
1411 017470 013737 002004 060000 MOV $TEMP,BUFO ;LOAD PRESET VALUE
1412 017476 005337 060000 DEC BUFO ;TO EXPECTED -1
1413 017502 052777 000002 162246 BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
(1) 017510 042777 000002 162240 BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
1414 017516 052777 010000 162232 BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
1415 017524 013737 002004 001124 MOV $TEMP,$GDDAT ;LOAD TYPEOUT EXPECTED
1416 017532 013737 060000 001126 MOV BUFO,$BDDAT ;READ THE BUFO LOCATION
1417 017540 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
1418 017546 001401 BEQ 2$ ;;BR IF SAME
1419 017550 104030 ERROR 30 ;INCORRECT DATA IN THE INCREMENT REGISTER
1420 017552 017737 162170 001126 2$: MOV @CSR,$BDDAT ;READ STATUS
1421 017560 032737 020000 001126 BIT #BIT13,$BDDAT ;TEST FOR UNEXPECTED CELL OVERFLOW
1422 017566 001407 BEQ 3$ ;BR IF NOT
1423 017570 012737 000024 001124 MOV #BIT4!BIT2,$GDDAT ;LOAD THE EXPECTED TYPEOUT
1424 017576 104031 ERROR 31 ;UNEXPECTED "CELL OVERFLOW" STATUS
1425 017600 052777 020000 162150 BIS #BIT13,@SFR ;TRY TO CLEAR THE FLAG
1426 017606 006337 002004 3$: ASL $TEMP ;CHANGE THE EXPCTED
1427 017612 001307 BNE 1$ ;BR IF MORE DATA BITS
  
```

1429
(3)
(3)
(2) 017614 000004
(1) 017616 012737 000040 001160
1430 017624 012777 004000 162124
1431 017632 005077 162114
1432 017636 005077 162112
1433 017642 012777 060000 162100
1434 017650 012777 000022 162070
1435 017656 012777 000014 162072
1436 017664 052777 000001 162054
1437 017672 052777 000002 162056
(1) 017700 042777 000002 162050
1438 017706 000240
1439 017710 000240
1440 017712 000240
1441 017714 112737 000377 060000
1442 017722 112737 000200 060001
1443 017730 052777 010000 162020
1444 017736 000240
1445 017740 000240
1446 017742 000240
1447 017744 017737 161776 001126
1448 017752 012737 020022 001124
1449 017760 023737 001124 001126
1450 017766 001401
1451 017770 104031
1452
1453
1454
1455
1456 017772 052777 020000 161756
1457 020000 017737 161742 001126
1458 020006 012737 000022 001124
1459 020014 023737 001124 001126
1460 020022 001401
1461 020024 104031
1462
1463
1464 020026 013737 060000 001126
1465 020034 012737 100377 001124
1466 020042 023737 001124 001126
1467 020050 001401
1468 020052 104031
1469
1470

```
*****  
*TEST 115 CHECK FOR LOW BYTE "INC OVFL" TO SET CELL OVERFLOW AND "CLR CELL" TO CLE  
*****  
TST115: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CLR @WCR ;CLEAR WC  
CLR @BAR ;CLEAR BAR  
MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
MOV #BIT4!BIT1,@CSR ;ENABLE BYTE MATRIX MODE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
MOVB #377,BUFO ;SET LOW BYTE OF BUFO LOC. TO BYTE -1  
MOVB #200,BUFO+1 ;SET HIGH BYTE OF BUFO TO KNOWN VALUE  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
MOV @CSR,$BDDAT ;READ STATUS  
MOV #BIT13!BIT4!BIT1,$GDDAT ;LOAD EXPECTED STATUS  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 1$ ;;BR IF SAME  
ERROR 31 ;"CELL OVERFLOW" FLOP FAILED TO SET  
;IN BYTE MODE FROM A LOW BYTE OVERFLOW  
  
;NOW GENERATE "CLR CELL" TO CLEAR CELL OVERFLOW FLOP  
1$: BIS #BIT13,@SFR ;GENERATE "CLR CELL"  
MOV @CSR,$BDDAT ;READ STATUS  
MOV #BIT4!BIT1,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 2$ ;;BR IF SAME  
ERROR 31 ;"CLR CELL" FAILED TO CLEAR "CELL OVFL" FLOP  
  
;NOW VERIFY THE BYTE DATA CELL  
2$: MOV BUFO,$BDDAT ;READ DATA  
MOV #100377,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST116 ;;BR IF SAME  
ERROR 31 ;OVERFLOW FROM INC. REG. FAILED TO INHIBIT  
;"INC CNT" FROM CHANGING THE DATA OR  
;"CSR BYTE CELLS" FAILED TO INHIBIT "INC ENA HI"
```

1472
(3)
(3)
(2) 020054 000004
(1) 020056 012737 000040 001160
1473 020064 012777 004000 161664
1474 020072 005077 161656
1475 020076 005077 161650
1476 020102 012777 060000 161640
1477 020110 012777 000024 161630
1478 020116 012777 000014 161632
1479 020124 052777 000001 161614
1480 020132 052777 000002 161616
(1) 020140 042777 000002 161610
1481 020146 000240
1482 020150 000240
1483 020152 000240
1484 020154 012737 177777 060000
1485 020162 052777 010000 161566
1486 020170 000240
1487 020172 000240
1488 020174 000240
1489 020176 017737 161544 001126
1490 020204 012737 020024 001124
1491 020212 023737 001124 001126
1492 020220 001401
1493 020222 104031
1494
1495 020224 052777 004000 161524
1496 020232 017737 161510 001126
1497 020240 012737 000200 001124
1498 020246 023737 001124 001126
1499 020254 001401
1500 020256 104031
1501
1502 020260 013737 060000 001126
1503 020266 012737 177777 001124
1504 020274 023737 001124 001126
1505 020302 001401
1506 020304 104031

```
::*****  
:*TEST 116 CHECK FOR WORD "INC OVFL" TO SET CELL OVERFLOW AND "CLR ALL" TO CLEAR IT  
:*****  
TST116: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CLR @BAR  
CLR @WCR  
MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
MOV #BIT4!BIT2,@CSR ;ENABLE MATRIX MODE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
NOP  
NOP  
NOP  
MOV #-1,BUFO ;SET BUFO LOC. TO WORD -1  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
MOV @CSR,$BDDAT ;READ STATUS  
MOV #BIT13!BIT4!BIT2,$GDDAT ;LOAD EXPECTED STATUS  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 1$ ;;BR IF SAME  
ERROR 31 ;"CELL OVERFLOW" FLOP FAILED TO SET  
:NOW GENERATE "CLR ALL" TO CLEAR CELL OVERFLOW FLOP  
1$: BIS #CLRALL,@SFR ;GENERATE "CLR ALL"  
MOV @CSR,$BDDAT ;READ STATUS  
MOV #BIT7,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 2$ ;;BR IF SAME  
ERROR 31 ;"CLR ALL" FAILED TO CLEAR "CELL OVFL" FLOP  
:NOW VERIFY THE WORD CELL DATA  
2$: MOV BUFO,$BDDAT ;READ DATA  
MOV #-1,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST117 ;;BR IF SAME  
ERROR 31
```

```
1508 (3) *****  
(3) *TEST 117 CHECK FOR "CELL OVERFLOW" INTERRUPT  
(2) 020306 000004 TST117: SCOPE  
(1) 020310 012737 000040 001160 MOV #40,$TIMES ;;DO 40 ITERATIONS  
1509 020316 012777 004000 161432 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
1510 020324 005077 161424 CLR @BAR  
1511 020330 005077 161416 CLR @WCR  
1512 020334 012777 060000 161406 MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
1513 020342 012777 000064 161376 MOV #BIT5!BIT4!BIT2,@CSR ;ENABLE INTR. AND MATRIX MODE  
1514 020350 012777 000014 161400 MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
1515 020356 052777 000001 161362 BIS #BIT0,@CSR ;ENABLE THE DEVICE  
1516 020364 052777 000002 161364 BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
(1) 020372 042777 000002 161356 BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
1517 020400 000240 NOP  
1518 020402 000240 NOP  
1519 020404 000240 NOP  
1520 020406 012737 177777 060000 MOV #-1,BUFO ;SET BUFO LOC. TO WORD -1  
1521 020414 012746 000000 MOV #0,-(SP)  
1522 020420 012746 020426 MOV #1$,-(SP)  
1523 020424 000002 RTI  
1524 020426 012777 020454 161342 1$: MOV #2$,@VECTBO ;LOAD RETURN VECTOR  
1525 020434 052777 010000 161314 BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
1526 020442 000240 NOP  
1527 020444 000240 NOP  
1528 020446 000240 NOP  
1529 020450 104032 ERROR 32 ;"CELL OVERFLOW" FAILED TO CAUES AN INTERRUPT  
1530 020452 000414 BR 3$ ;;BR TO CLEAN UP  
1531 020454 022626 2$: CMP (SP)+,(SP)+  
1532 020456 017737 161264 001126 MOV @CSR,$BDDAT ;READ STATUS  
1533 020464 012737 020264 001124 MOV #BIT13!BIT7!BIT5!BIT4!BIT2,$GDDAT ;LOAD EXPECTED  
1534 020472 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUE  
1535 020500 001401 BEQ 3$ ;;BR IF "ACTIVE" CLEARED  
1536 020502 104031 ERROR 31 ;"ACTIVE" FAILED TO CLEAR  
1537 020504 0177 004000 161244 3$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
1538 020512 013777 002000 161256 MOV VECTB1,@VECTBO ;RESET THE VECTORS  
1539 020520 005077 161254 CLR @VECTB1
```

```

1541      ;:*****
(3)      ;*TEST 120  VERIFY CORRECT BR LEVEL THE NCV-11
(3)      ;:*****
(2) 020524 000004  TST120: SCOPE
(1) 020526 012737 000040 001160  MOV #40,$TIMES ;:DO 40 ITERATIONS
1542 020534 013746 002002  MOV BRLEV,-(SP) ;:PUSH THE EXPECTED BR LEVEL ON STACK
1543 020540 012746 020546  MOV #1$,-(SP) ;:PUSH THE RETURN ADDRESS ON STACK
1544 020544 000002  RTI ;FAKE AN INTERRUPT TO BE LSI-11 AND PDP-11 COMPATABLE
1545 020546 012777 004000 161202 1$: MOV #CLRALL,@SFR ;:CLEAR THE DEVICE
1546 020554 012777 020750 161214  MOV #10$,@VECTB0 ;:LOAD UNEXPECTED INTERRUPT RETURN
1547 020562 012777 000340 161210  MOV #340,@VECTB1
1548 020570 005077 161160  CLR @BAR ;:INIT THE LOW Z COUNT
-1549 020574 005077 161152  CLR @WCR ;:INIT THE HIGH Z COUNT
1550 020600 012777 060000 161142  MOV #BUFO,@OFF ;:PRIME THE OFFSET REGISTER
1551 020606 012777 000064 161132  MOV #64,@CSR ;:SELECT INTR. AND MATRIX MODE
1552 020614 012777 000014 161134  MOV #TSTDMA!TSTCON,@SFR ;:MAINT. MODE
1553 020622 012737 177777 060000  MOV #-1,BUFO ;:PRIME THE TARGET LOCATION
1554 020630 052777 000001 161110  BIS #BIT0,@CSR ;:ENABLE THE NCV-11
1555 020636 052777 000002 161112  BIS #TESTZ,@SFR ;:ENABLE "TEST Z" PULSES
(1) 020644 042777 000002 161104  BIC #TESTZ,@SFR ;:DISABLE "TEST Z" PULSES
1556 020652 000240  NOP
1557 020654 000240  NOP
1558 020656 000240  NOP
1559 020660 052777 010000 161070  BIS #BIT12,@SFR ;:ENABLE 1 TRANSFER
1560 020666 000240  NOP
1561 020670 000240  NOP
1562 020672 000240  NOP
1563 020674 000240  NOP
1564 020676 000240  NOP
1565 020700 000240  NOP
1566 020702 013700 002002  MOV BRLEV,R0 ;:GET CURRENT BR LEVEL
1567 020706 162700 000040  SUB #40,R0 ;:AND MAKE IT 1 LEVEL LOWER
1568 020712 005737 003356  TST NLSI11 ;:TEST IF LSI-11 CPU
1569 020716 001401  BEQ 2$ ;:BR IF NOT
1570 020720 005000  CLR R0 ;:LOAD BR LEVEL 0
1571 020722 012777 020756 161046 2$: MOV #20$,@VECTB0 ;:RELOAD TO EXPECTED VECTOR
1572 020730 010046  MOV R0,-(SP) ;:PUSH ADJUSTED BR LEVEL
1573 020732 012746 020740  MOV #3$,-(SP) ;:PUSH RETURN ADDRESS
1574 020736 000002  RTI ;:LOWER CPU BR LEVEL
1575 020740 000240 3$: NOP
1576 020742 000240  NOP
1577 020744 104042  ERROR 42 ;:NCV11 FAILED TO INTERRUPT - INCORRECT NCV11 BR LEVEL?
1578 020746 000404  BR 21$ ;:BR TO CLEAN-UP
1579
1580 ;:UNEXPECTED INTERRUPT WITH BR LEVEL INDICATED
1581 020750 022626 10$: CMP (SP)+,(SP)+ ;:CLEAN THE STACK
1582 020752 104042  ERROR 42 ;:NCV11 INTERRUPTED ON AN INCORRECT BR LEVEL
1583 020754 000401  BR 21$ ;:BR TO CLEANUP
1584
1585 ;:EXPECTED INTERRUPT DID OCCUR
1586 020756 022626 20$: CMP (SP)+,(SP)+ ;:CLEAN THE STACK
1587 020760 012777 004000 160770 21$: MOV #CLRALL,@SFR ;:CLEAN THE DEVICE
1588 020766 013777 002000 161002  MOV VECTB1,@VECTB0 ;:RESET THE VECTOR
1589 020774 005077 161000  CLR @VECTB1
  
```

1612

(4)
(4)
(3) 021000 000004
(2) 021002 012737 000100 001160
(1) 021010 012777 004000 160740
(1) 021016 012777 060000 160724
(1) 021024 012777 000014 160724
(1) 021032 012777 000024 160706
(1) 021040 052777 000001 160700
(2) 021046 052777 000002 160702
(2) 021054 042777 000002 160674
(1) 021062 005037 060000
(1) 021066 052777 010000 160662
(1) 021074 012737 060000 001122
(1) 021102 012737 000001 001124
(1) 021110 017737 160006 001126
(1) 021116 023737 001124 001126
(3) 021124 001401
(1) 021126 104033
(1)

```
::*****  
:*TEST 121 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 13  
:*****  
TST121: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60000,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60000 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60000,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST122 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
;THE WRONG ADDRESS - EXPECTED ADR. WAS 60000
```

1613

(4)
(4)
(3) 021130 000004
(2) 021132 012737 000100 001160
(1) 021140 012777 004000 160610
(1) 021146 012777 070000 160574
(1) 021154 012777 000014 160574
(1) 021162 012777 000024 160556
(1) 021170 052777 000001 160550
(2) 021176 052777 000002 160552
(2) 021204 042777 000002 160544
(1) 021212 005037 070000
(1) 021216 052777 010000 160532
(1) 021224 012737 070000 001122
(1) 021232 012737 000001 001124
(1) 021240 017737 157656 001126
(1) 021246 023737 001124 001126
(3) 021254 001401
(1) 021256 104033
(1)

```
::*****  
:*TEST 122 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 12  
:*****  
TST122: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #70000,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#70000 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #70000,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST123 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
;THE WRONG ADDRESS - EXPECTED ADR. WAS 70000
```

1615

(4)
(4)
(3) 021260 000004
(2) 021262 012737 000100 001160
(1) 021270 012777 004000 160460
(1) 021276 012777 064000 160444
(1) 021304 012777 000014 160444
(1) 021312 012777 000024 160426
(1) 021320 052777 000001 160420
(2) 021326 052777 000002 160422
(2) 021334 042777 000002 160414
(1) 021342 005037 064000
(1) 021346 052777 010000 160402
(1) 021354 012737 064000 001122
(1) 021362 012737 000001 001124
(1) 021370 017737 157526 001126
(1) 021376 023737 001124 001126
(3) 021404 001401
(1) 021406 104033

```
::*****  
:*TEST 123 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 11  
:*****  
TST123: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #64000,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#64000 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #64000,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST124 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
;THE WRONG ADDRESS - EXPECTED ADR. WAS 64000
```

1616

(4)
(4)
(3) 021410 000004
(2) 021412 012737 000100 001160
(1) 021420 012777 004000 160330
(1) 021426 012777 062000 160314
(1) 021434 012777 000014 160314
(1) 021442 012777 000024 160276
(1) 021450 052777 000001 160270
(2) 021456 052777 000002 160272
(2) 021464 042777 000002 160264
(1) 021472 005037 062000
(1) 021476 052777 010000 160252
(1) 021504 012737 062000 001122
(1) 021512 012737 000001 001124
(1) 021520 017737 157376 001126
(1) 021526 023737 001124 001126
(3) 021534 001401
(1) 021536 104033

```
::*****  
:*TEST 124 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 10  
:*****  
TST124: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #62000,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#62000 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #62000,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST125 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
;THE WRONG ADDRESS - EXPECTED ADR. WAS 62000
```


1618

(4)
(4)
(3) 021540 000004
(2) 021542 012737 000100 001160
(1) 021550 012777 004000 160200
(1) 021556 012777 061000 160164
(1) 021564 012777 000014 160164
(1) 021572 012777 000024 160146
(1) 021600 052777 000001 160140
(2) 021606 052777 000002 160142
(2) 021614 042777 000002 160134
(1) 021622 005037 061000
(1) 021626 052777 010000 160122
(1) 021634 012737 061000 001122
(1) 021642 012737 000001 001124
(1) 021650 017737 157246 001126
(1) 021656 023737 001124 001126
(3) 021664 001401
(1) 021666 104033

```
*****  
*TEST 125 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 09  
*****  
TST125: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #61000,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#61000 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #61000,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST126 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 61000
```

1619

(4)
(4)
(3) 021670 000004
(2) 021672 012737 000100 001160
(1) 021700 012777 004000 160050
(1) 021706 012777 060400 160034
(1) 021714 012777 000014 160034
(1) 021722 012777 000024 160016
(1) 021730 052777 000001 160010
(2) 021736 052777 000002 160012
(2) 021744 042777 000002 160004
(1) 021752 005037 060400
(1) 021756 052777 010000 157772
(1) 021764 012737 060400 001122
(1) 021772 012737 000001 001124
(1) 022000 017737 157116 001126
(1) 022006 023737 001124 001126
(3) 022014 001401
(1) 022016 104033

```
*****  
*TEST 126 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 08  
*****  
TST126: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60400,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60400 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60400,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST127 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60400
```

1621

(4)
(4)
(3) 022020 000004
(2) 022022 012737 000100 001160
(1) 022030 012777 004000 157720
(1) 022036 012777 060200 157704
(1) 022044 012777 000014 157704
(1) 022052 012777 000024 157666
(1) 022060 052777 000001 157660
(2) 022066 052777 000002 157662
(2) 022074 042777 000002 157654
(1) 022102 005037 060200
(1) 022106 052777 010000 157642
(1) 022114 012737 060200 001122
(1) 022122 012737 000001 001124
(1) 022130 017737 156766 001126
(1) 022136 023737 001124 001126
(3) 022144 001401
(1) 022146 104033

```
::*****  
:*TEST 127 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 07  
:*****  
TST127: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60200,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60200 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60200,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST130 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60200
```

1622

(4)
(4)
(3) 022150 000004
(2) 022152 012737 000100 001160
(1) 022160 012777 004000 157570
(1) 022166 012777 060100 157554
(1) 022174 012777 000014 157554
(1) 022202 012777 000024 157536
(1) 022210 052777 000001 157530
(2) 022216 052777 000002 157532
(2) 022224 042777 000002 157524
(1) 022232 005037 060100
(1) 022236 052777 010000 157512
(1) 022244 012737 060100 001122
(1) 022252 012737 000001 001124
(1) 022260 017737 156636 001126
(1) 022266 023737 001124 001126
(3) 022274 001401
(1) 022276 104033
(1)

```
::*****  
:*TEST 130 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 06  
:*****  
TST130: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60100,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60100 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60100,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST131 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60100
```

1624

(4)
(4)
(3) 022300 000004
(2) 022302 012737 000100 001160
(1) 022310 012777 004000 157440
(1) 022316 012777 060040 157424
(1) 022324 012777 000014 157424
(1) 022332 012777 000024 157406
(1) 022340 052777 000001 157400
(2) 022346 052777 000002 157402
(2) 022354 042777 000002 157374
(1) 022362 005037 060040
(1) 022366 052777 010000 157362
(1) 022374 012737 060040 001122
(1) 022402 012737 000001 001124
(1) 022410 017737 156506 001126
(1) 022416 023737 001124 001126
(3) 022424 001401
(1) 022426 104033

```
*****  
: *TEST 131 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 05  
: *****  
TST131: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60040,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60040 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60040,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST132 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60040
```

1625

(4)
(4)
(3) 022430 000004
(2) 022432 012737 000100 001160
(1) 022440 012777 004000 157310
(1) 022446 012777 060020 157274
(1) 022454 012777 000014 157274
(1) 022462 012777 000024 157256
(1) 022470 052777 000001 157250
(2) 022476 052777 000002 157252
(2) 022504 042777 000002 157244
(1) 022512 005037 060020
(1) 022516 052777 010000 157232
(1) 022524 012737 060020 001122
(1) 022532 012737 000001 001124
(1) 022540 017737 156356 001126
(1) 022546 023737 001124 001126
(3) 022554 001401
(1) 022556 104033

```
*****  
: *TEST 132 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 04  
: *****  
TST132: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60020,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60020 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60020,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST133 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60020
```

1627

(4)

(4)

(3) 022560 000004
(2) 022562 012737 000100 001160
(1) 022570 012777 004000 157160
(1) 022576 012777 060010 157144
(1) 022604 012777 000014 157144
(1) 022612 012777 000024 157126
(1) 022620 052777 000001 157120
(2) 022626 052777 000002 157122
(2) 022634 042777 000002 157114
(1) 022642 005037 060010
(1) 022646 052777 010000 157102
(1) 022654 012737 060010 001122
(1) 022662 012737 000001 001124
(1) 022670 017737 156226 001126
(1) 022676 023737 001124 001126
(3) 022704 001401
(1) 022706 104033

```
*****  
*TEST 133 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 03  
*****  
TST133: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60010,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60010 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60010,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST134 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60010
```

1628

(4)

(4)

(3) 022710 000004
(2) 022712 012737 000100 001160
(1) 022720 012777 004000 157030
(1) 022726 012777 060004 157014
(1) 022734 012777 000014 157014
(1) 022742 012777 000024 156776
(1) 022750 052777 000001 156770
(2) 022756 052777 000002 156772
(2) 022764 042777 000002 156764
(1) 022772 005037 060004
(1) 022776 052777 010000 156752
(1) 023004 012737 060004 001122
(1) 023012 012737 000001 001124
(1) 023020 017737 156076 001126
(1) 023026 023737 001124 001126
(3) 023034 001401
(1) 023036 104033
(1)

```
*****  
*TEST 134 VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 02  
*****  
TST134: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #60004,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;ENABLE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#60004 ;CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
MOV #60004,$BDADR ;LOAD EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD EXPECTED DATA  
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST135 ;;BR IF SAME  
ERROR 33 ;OFFSET INPUT TO MATRIX MUX. SELECTED  
 ;THE WRONG ADDRESS - EXPECTED ADR. WAS 60004
```

1633
(3)
(3)
(2) 023040 000004
(1) 023042 012737 000010 001160
1634
1635
1636
1637 023050 012777 004000 156700
1638 023056 012777 062550 156664
1639 023064 012777 000014 156664
1640 023072 012777 002024 156646
1641 023100 052777 000001 156640
1642 023106 052777 000002 156642
(1) 023114 042777 000002 156634
1643 023122 005037 065322
1644 023126 005037 062722
1645 023132 052777 010000 156616
1646 023140 000240
1647 023142 000240
1648 023144 000240
1649 023146 012737 065322 001122
1650 023154 012737 000001 001124
1651 023162 017737 155734 001126
1652 023170 023737 001124 001126
1653 023176 001413
1654 023200 005737 062722
1655 023204 001002
1656 023206 104033
1657
1658 023210 000406
1659 023212 013737 062722 001126 1\$:
1660 023220 005037 001124
1661 023224 104033
1662

```
::*****  
:*TEST 135 VERIFY THE ADM INPUT TO THE MATRIX MUX. USING GAIN ENABLE  
:*****  
TST135: SCGPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
:WITH GAIN SET IN TEST CONTROL MODE, THE SELF TEST DATA SHOULD BE BYTE 250  
:WHEN PROCESSED THRU THE ADDRESS MAKER LOGIC THE NEW VALUE IS "2552"  
:VERIFY THAT THE MATRIX MUX CAN ADD 62550 AND 2552 CORRECTLY  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #BUFO+2550,@OFF ;LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
MOV #2024,@CSR ;ENABLE GAIN, WORD MATRIX MODE  
BIS #BIT0,@CSR ;ENABLE THE NCV11  
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES  
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES  
CLR @#BUFO+5322 ;CLEAR THE TARGET ADDRESS  
CLR @#BUFO+2722 ;CLEAR THE TARGET ADDRESS IF "TESTX" FAILS  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
NOP  
MOV #BUFO+5322,$BDADR ;LOAD THE EXPECTED ADDRESS  
MOV #1,$GDDAT ;LOAD THE EXPECTED DATA  
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST136 ;;BR IF SAME  
TST @#BUFO+2722 ;TEST OTHER ADDRESS  
BNE 1$ ;BR IF NON-ZERO  
ERROR 33 ;MATRIX MODE ADDER ERROR  
;USING GAIN AND WORD MATRIX MODE, THE ADDRESSES SELECTED WAS INCORRECT  
BR TST136 ;;  
MOV @#BUFO+2722,$BDDAT ;LOAD INCORRECT DATA  
CLR $GDDAT ;CLEAR THE EXPECTED DATA  
ERROR 33 ;MATRIX MODE ADDER ERROR DUE TO  
;"TESTX" LOGIC SET IN ERROR
```

```

1664
(3)
(3)
(2) 023226 000004
(1) 023230 012737 000010 001160
1665
1666
1667
1668 023236 012777 004000 156512
1669 023244 012777 065224 156476
1670 023252 012777 000014 156476
1671 023260 012777 004024 156460
1672 023266 052777 000002 156462
1673 023274 052777 000001 156444
1674 023302 042777 000001 156446
1675 023310 005037 072450
1676 023314 052777 010000 156434
1677 023322 012737 072450 001122
1678 023330 012737 000001 001124
1679 023336 017737 155560 001126
1680 023344 023737 001124 001126
1681 023352 001401
1682 023354 104033
1683
1684
(3)
(3)
(2) 023356 000004
(1) 023360 012737 000010 001160
1685 023366 012777 004000 156362
1686 023374 012777 060000 156346
1687 023402 012777 000014 156346
1688 023410 012777 002024 156330
1689 023416 052777 000001 156322
1690 023424 052777 000002 156324
(1) 023432 042777 000002 156316
1691 023440 052777 000020 156310
1692 023446 005037 062552
1693 023452 005037 060152
1694 023456 052777 010000 156272
1695 023464 000240
1696 023466 000240
1697 023470 000240
1698 023472 012737 000001 001124
1699 023500 013737 060152 001126
1700 023506 023737 001124 001126
1701 023514 001413
1702 023516 005737 062552
1703 023522 001002
1704 023524 104034
1705 023526 000406
1706 023530 013737 062552 001126 1$:
1707 023536 005037 001124
1708 023542 104034

*****
*TEST 136 VERIFY THE ADM INPUT TO THE MATRIX MUX. ADDER USING ZB ENABLE
*****
TST136: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
;WITH ZB ENABLE SET IN TEST CONTROL MODE, THE SELF TEST DATA SHOULD BE BYTE 224
;WHEN PROCESSED THRU THE ADDRESS MAKER LOGIC THE NEW VALUE IS 12450
;VERIFY THAT THE MATRIX MUX CAN ADD CORRECTLY
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUFO+5224,@OFF ;LOAD THE OFFSET REGISTER
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #4024,@CSR ;ENABLE ZB AND WORD MATRIX MODE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIS #BIT0,@CSR ;ENABLE NCV11
BIC #BIT0,@SFR ;DISABLE "TEST Z" PULSES
CLR @#BUFO+12450 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
MOV #BUFO+12450,$BDADR ;LOAD THE EXPECTED ADDRESS
MOV #1,$GDDAT ;LOAD THE EXPECTED DATA
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST137 ;;BR IF SAME
ERROR 33 ;ADM INPUT TO THE MATRIX MUX SELECTED
;WRONG ADDRESS - EXPECTED ADDRESS WAS BUFO + 2450

*****
*TEST 137 VERIFY LOW BYTE OPERATION OF THE "TESTX" FLOP
*****
TST137: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUFO,@OFF ;LOAD OFFSET REG.
MOV #TSTDMA!TSICON,@SFR ;LOAD TEST DMA AND CONTROL
MOV #2024,@CSR ;SET "GAIN" AND WORD MATRIX MODE
BIS #BIT0,@CSR ;SET NCV11 ACTIVE
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
BIS #BIT4,@SFR ;SET "TESTX" FLOP
CLR @#BUFO+2552 ;CLEAR THE TEST FAILED LOCATION
CLR @#BUFO+0152 ;CLEAR THE TESTX WORKED LOCATION
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
NOP
NOP
NOP
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @#BUFO+0152,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE DATA
BEQ TST140 ;;BR IF SAME
TST @#BUFO+2552 ;TEST TESTX FAILED LOCATION
BNE 1$ ;;BR IF YES
ERROR 34 ;TESTX FAILED TO INHIBIT BITS 8-15 OF THE "SUM ADDER"
BR TST140
MOV @#BUFO+2552,$BDDAT ;GET ACTUAL DATA
CLR $GDDAT ;CLEAR EXPECTED
ERROR 34 ;TESTX FAILED TO INHIBIT BITS 8-15 OF THE "SUM A

```

```

1710
(3)
(3)
(2) 023544 000004
(1) 023546 012737 000100 001160
1711 023554 012777 004000 156174
1712 023562 012777 050000 156160
1713 023570 012777 000014 156160
1714 023576 012777 002030 156142
1715 023604 052777 000001 156134
1716 023612 052777 000002 156136
(1) 023620 042777 090002 156130
1717 023626 005037 062726
1718 023632 052777 010000 156116
1719 023640 012737 062726 001122
1720 023646 012737 000001 001124
1721 023654 017737 155242 001126
1722 023662 023737 001124 001126
1723 023670 001401
1724 023672 104033
1725
1726
(3)
(3)
(2) 023674 000004
(1) 023676 012737 000100 001160
1727 023704 012777 004000 156044
1728 023712 012777 037774 156030
1729 023720 012777 000014 156030
1730 023726 012777 002032 156012
1731 023734 052777 000001 156004
1732 023742 052777 000002 156006
(1) 023750 042777 000002 156000
1733 023756 005037 065522
1734 023762 052777 010000 155766
1735 023770 012737 065522 001122
1736 023776 012737 000001 001124
1737 024004 017737 155112 001126
1738 024012 023737 001124 001126
1739 024020 001401
1740 024022 104033
1741
  
```

```

*****
;*TEST 140 VERIFY BIT 12 ADDER INPUT TO MATRIX MODE MUX
*****
TST140: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #50000,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2030,@CSR ;SELECT GAIN MATRIX BYTE MODE
BIS #BIT0,@CSR ;ENABLE NCV11
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
CLR @#BUFO+2726 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER
MOV #BUFO+2726,$BDADR ;LOAD EXPECTED ADDRESS
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST141 ;;BR IF SAME
ERROR 33 ;MATRIX ADDER FAILED TO SELECT CORRECT ADDRESS
;THE WRONG ADDRESS - EXPECTED ADR. WAS BUFO+2726
  
```

```

*****
;*TEST 141 VERIFY BIT 13 ADDER INPUT TO MATRIX MODE MUX
*****
TST141: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #37774,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2032,@CSR ;SELECT GAIN MATRIX BYTE MODE
BIS #BIT0,@CSR ;ENABLE NCV11
BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
CLR @#BUFO+5522 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER
MOV #BUFO+5522,$BDADR ;LOAD EXPECTED ADDRESS
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @#BDADR,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST142 ;;BR IF SAME
ERROR 33 ;MATRIX ADDER FAILED TO SELECT CORRECT ADDRESS
;THE WRONG ADDRESS - EXPECTED ADR. WAS BUFO+5522
  
```

```
1743      ;*****  
(3)      ;*TEST 142   VERIFY BIT 14 ADDER INPUT TO MATRIX MODE MUX  
(3)      ;*****  
(2) 024024 000004  
(1) 024026 012737 000100 001160 TST142: SCOPE  
1744      MOV #100,$TIMES      ;;DO 100 ITERATIONS  
1745      ;OFFSET = 50000  
1746      ;ADM OUTPUT = 53656  
1747      ;TARGET = 123656  
1747 024034 012777 004000 155714 MOV #CLRALL,@SFR      ;CLEAR THE DEVICE  
1748 024042 023727 036406 001340 CMP $LSTBK,#1340      ;TEST IF ENOUGH MEMORY >20K  
1749 024050 103445 BLO TST143      ;;BR IF NO ROOM  
1750 024052 012777 050000 155670 MOV #50000,@OFF      ;LOAD THE OFFSET REG.  
1751 024060 012777 000014 155670 MOV #TSTDMA!TSTCON,@SFR      ;SET TEST DMA AND CONTROL  
1752 024066 012777 002034 155652 MOV #2034,@CSR      ;SELECT GAIN MATRIX BYTE MODE  
1753 024074 052777 000001 155644 BIS #BIT0,@CSR      ;ENABLE NCV11  
1754 024102 052777 000002 155646 BIS #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES  
(1) 024110 042777 000002 155640 BIC #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES  
1755 024116 005037 123656 CLR @#BUFO+43656      ;CLEAR THE TARGET LOCATION  
1756 024122 052777 010000 155626 BIS #BIT12,@SFR      ;ALLOW 1 WORD TRANSFER  
1757 024130 012737 123656 001122 MOV #BUFO+43656,$BDADR      ;LOAD EXPECTED ADDRESS  
1758 024136 012737 000001 001124 MOV #1,$GDDAT      ;LOAD EXPECTED DATA  
1759 024144 017737 154752 001126 MOV @$BDADR,$BDDAT      ;READ THE ACTUAL DATA  
1760 024152 023737 001124 001126 CMP $GDDAT,$BDDAT      ;COMPARE  
1761 024160 001401 BEQ TST143      ;;BR IF SAME  
1762 024162 104033 ERROR 33      ;MATRIX ADDER FAILED TO SELECT CORRECT ADDRESS  
1763      ;THE WRONG ADDRESS - EXPECTED ADR. WAS BUFO+43656
```



```
1765      ;:*****  
(3)      ;*TEST 143      CHECK FOR HIGH BYTE "INC OVFL" TO SET CELL OVERFLOW  
(3)      ;:*****  
(2) 024164 000004      TST143: SCOPE  
(1) 024166 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS  
1766 024174 012777 004000 155554      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE  
1767 024202 005077 155544      CLR      @WCR      ;CLEAR W.C.  
1768 024206 005077 155542      CLR      @BAR      ;CLEAR B.A.  
1769 024212 012777 060000 155530      MOV      #BUFO,@OFF      ;LOAD INITIAL OFFSET REGISTER  
1770 024220 012777 002022 155520      MOV      #BIT10!BIT4!BIT1,@CSR      ;ENABLE MATRIX MODE BYTE AND SET GAIN  
1771 024226 012777 000014 155522      MOV      #TSTDMA!TSTCON,@SFR      ;SET TEST DMA AND CONTROL  
1772 024234 052777 000001 155504      BIS      #BIT0,@CSR      ;ENABLE THE DEVICE  
1773 024242 052777 000002 155506      BIS      #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES  
(1) 024250 042777 000002 155500      BIC      #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES  
1774 024256 000240      NOP  
1775 024260 000240      NOP  
1776 024262 000240      NOP  
1777 024264 112737 000100 061264      MOVB     #100,BUFO+1264      ;LOAD LOW BYTE TO A KNOWN VALUE  
1778 024272 112737 000377 061265      MOVB     #377,BUFO+1265      ;SET HIGH BYTE TO 377  
1779 024300 052777 010000 155450      BIS      #BIT12,@SFR      ;ALLOW 1 TRANSFER  
1780 024306 000240      NOP  
1781 024310 000240      NOP  
1782 024312 000240      NOP  
1783 024314 012737 022022 001124      MOV      #BIT13!BIT10!BIT4!BIT1,$GDDAT      ;LOAD EXPECTED STATUS  
1784 024322 017737 155420 001126      MOV      @CSR,$BDDAT      ;READ THE ACTUAL STATUS  
1785 024330 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE  
1786 024336 001401      BEQ      1$      ;;BR IF SAME  
1787 024340 104030      ERROR    30      ;CELL OVERFLOW FAILED TO SET  
1788      ;WHEN INCREMENTING THE HIGH BYTE  
1789 024342 013737 061264 001126 1$:      MOV      BUFO+1264,$BDDAT      ;READ THE BUFO LOCATION  
1790 024350 012737 177500 001124      MOV      #177500,$GDDAT      ;LOAD EXPECTED VALUE  
1791 024356 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES  
1792 024364 001401      BEQ      TST144      ;;BR IF SAME  
1793 024366 104030      ERROR    30      ;TARGET LOC. DATA WAS INCORRECT
```

```

1795
(3)
(3)
(2) 024370 000004
(1) 024372 012737 000400 001160
1796 024400 005737 050026
1797 024404 001454
1798 024406 012737 000001 001124
1799 024414 012737 024422 001110
1800
1801 024422 012777 004000 155326 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1802 024430 012777 000010 155320 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
1803 024436 012777 000000 155306 MOV #0,@WCR ;LOAD UPPER 16 BITS
1804 024444 013777 001124 155302 MOV $GDDAT,@BAR ;
1805 024452 005377 155276 DEC @BAR ;LOAD INITIAL COUNTER VALUE
1806 024456 012777 000022 155262 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
1807 024464 052777 000001 155254 BIS #BIT0,@CSR ;ENABLE THE DEVICE
1808 024472 052777 000001 155226 BIS #BIT0,@DACSR ;GENERATE '7' PULSE
1809 024500 105777 155222 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
1810 024504 100375 BPL 2$
1811 024506 017737 155242 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE
1812 024514 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1813 024522 001402 BEQ 3$ ;BR IF SAME
1814 024524 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER IN ERROR
1815 024526 000403 BR TST145 ;
1816 024530 006337 001124 3$: ASL $GDDAT ;CHANGE THE DATA BIT
1817 024534 001332 BNE 1$ ;BR AND TRY NEXT BIT
1818
(3)
(3)
(2) 024536 000004
(1) 024540 012737 000400 001160
1819 024546 005737 050026
1820 024552 001454
1821 024554 012737 000001 001124
1822 024562 012737 024570 001110
1823
1824 024570 012777 004000 155160 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1825 024576 012777 000010 155152 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
1826 024604 013777 001124 155140 MOV $GDDAT,@WCR ;
1827 024612 005377 155134 DEC @WCR ;LOAD INITIAL COUNTER VALUE
1828 024616 012777 177777 155130 MOV #-1,@BAR ;LOAD LOWER 16 BITS
1829 024624 012777 000022 155114 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
1830 024632 052777 000001 155106 BIS #BIT0,@CSR ;ENABLE THE DEVICE
1831 024640 052777 000001 155060 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
1832 024646 105777 155054 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
1833 024652 100375 BPL 2$
1834 024654 017737 155072 001126 MOV @WCR,$BDDAT ;READ EXPECTED REGISTER VALUE
1835 024662 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1836 024670 001402 BEQ 3$ ;BR IF SAME
1837 024672 104011 ERROR 11 ;HIGHER 16 BITS OF THE Z COUNTER IN ERROR
1838 024674 000403 BR TST146 ;
1839 024676 006337 001124 3$: ASL $GDDAT ;CHANGE THE DATA BIT
1840 024702 001332 BNE 1$ ;BR AND TRY NEXT BIT

```

```

1842      ::*****
(3)      : *TEST 146   VERIFY THAT CAMERA 01 CHANNEL IS OPERATIONAL   (TESTER ONLY)
(3)      : *****
(2) 024704 000004
(1) 024706 012737 000400 001160
1843 024714 005737 050026
1844 024720 001443
1845 024722 012737 000001 001124
1846
1847 024730 012777 004000 155020 1$:  MOV   #CLRALL,@SFR           ;CLEAR THE DEVICE
1848 024736 012777 000010 155012      MOV   #TSTDMA,@SFR         ;SET TEST DMA FLOP
1849 024744 012777 000000 155000      MOV   #0,@WCR            ;LOAD UPPER 16 BITS
1850 024752 012777 000000 154774      MOV   #0,@BAR            ;LOAD LOWER 16 BITS
1851 024760 012777 000422 154760      MOV   #BIT8!BIT4!BIT1,@CSR ;ENTER MATRIX MODE ON CAMERA 01
1852 024766 052777 000001 154752      BIS   #BIT0,@CSR         ;ENABLE THE DEVICE
1853 024774 052777 000001 154724      BIS   #BIT0,@DACSR       ;GENERATE 'Z' PULSE
1854 025002 105777 154720 2$:  TSTB  @DACSR             ;WAIT FOR Z PULSE COMPLETION
1855 025006 100375
1856 025010 017737 154740 001126      MOV   @BAR,$BDDAT        ;READ EXPECTED REGISTER VALUE
1857 025016 023737 001124 001126      CMP   $GDDAT,$BDDAT      ;COMPARE
1858 025024 001401
1859 025026 104010
1860
1861
1862      ::*****
(3)      : *TEST 146   VERIFY THAT CAMERA 01 CHANNEL IS OPERATIONAL   (TESTER ONLY)
(3)      : *****
(2) 024704 000004
(1) 024706 012737 000400 001160
1843 024714 005737 050026
1844 024720 001443
1845 024722 012737 000001 001124
1846
1847 024730 012777 004000 155020 1$:  MOV   #CLRALL,@SFR           ;CLEAR THE DEVICE
1848 024736 012777 000010 155012      MOV   #TSTDMA,@SFR         ;SET TEST DMA FLOP
1849 024744 012777 000000 155000      MOV   #0,@WCR            ;LOAD UPPER 16 BITS
1850 024752 012777 000000 154774      MOV   #0,@BAR            ;LOAD LOWER 16 BITS
1851 024760 012777 000422 154760      MOV   #BIT8!BIT4!BIT1,@CSR ;ENTER MATRIX MODE ON CAMERA 01
1852 024766 052777 000001 154752      BIS   #BIT0,@CSR         ;ENABLE THE DEVICE
1853 024774 052777 000001 154724      BIS   #BIT0,@DACSR       ;GENERATE 'Z' PULSE
1854 025002 105777 154720 2$:  TSTB  @DACSR             ;WAIT FOR Z PULSE COMPLETION
1855 025006 100375
1856 025010 017737 154740 001126      MOV   @BAR,$BDDAT        ;READ EXPECTED REGISTER VALUE
1857 025016 023737 001124 001126      CMP   $GDDAT,$BDDAT      ;COMPARE
1858 025024 001401
1859 025026 104010
1860
1861
1862      ::*****
(3)      : *TEST 147   VERIFY THAT CAMERA 10 CHANNEL IS OPERATIONAL  (TESTER ONLY)
(3)      : *****
(2) 025030 000004
(1) 025032 012737 000100 001160
1863 025040 005737 050026
1864 025044 001443
1865 025046 012737 000001 001124
1866
1867 025054 012777 004000 154674 1$:  MOV   #CLRALL,@SFR           ;CLEAR THE DEVICE
1868 025062 012777 000010 154666      MOV   #TSTDMA,@SFR         ;SET TEST DMA FLOP
1869 025070 012777 000000 154654      MOV   #0,@WCR            ;LOAD UPPER 16 BITS
1870 025076 012777 000000 154650      MOV   #0,@BAR            ;LOAD LOWER 16 BITS
1871 025104 012777 001022 154634      MOV   #BIT9!BIT4!BIT1,@CSR ;ENTER MATRIX MODE
1872 025112 052777 000001 154626      BIS   #BIT0,@CSR         ;ENABLE THE DEVICE
1873 025120 052777 000001 154600      BIS   #BIT0,@DACSR       ;GENERATE 'Z' PULSE
1874 025126 105777 154574 2$:  TSTB  @DACSR             ;WAIT FOR Z PULSE COMPLETION
1875 025132 100375
1876 025134 017737 154614 001126      MOV   @BAR,$BDDAT        ;READ EXPECTED REGISTER VALUE
1877 025142 023737 001124 001126      CMP   $GDDAT,$BDDAT      ;COMPARE
1878 025150 001401
1879 025152 104010
1880

```

```

1862      ::*****
(3)      : *TEST 147   VERIFY THAT CAMERA 10 CHANNEL IS OPERATIONAL  (TESTER ONLY)
(3)      : *****
(2) 025030 000004
(1) 025032 012737 000100 001160
1863 025040 005737 050026
1864 025044 001443
1865 025046 012737 000001 001124
1866
1867 025054 012777 004000 154674 1$:  MOV   #CLRALL,@SFR           ;CLEAR THE DEVICE
1868 025062 012777 000010 154666      MOV   #TSTDMA,@SFR         ;SET TEST DMA FLOP
1869 025070 012777 000000 154654      MOV   #0,@WCR            ;LOAD UPPER 16 BITS
1870 025076 012777 000000 154650      MOV   #0,@BAR            ;LOAD LOWER 16 BITS
1871 025104 012777 001022 154634      MOV   #BIT9!BIT4!BIT1,@CSR ;ENTER MATRIX MODE
1872 025112 052777 000001 154626      BIS   #BIT0,@CSR         ;ENABLE THE DEVICE
1873 025120 052777 000001 154600      BIS   #BIT0,@DACSR       ;GENERATE 'Z' PULSE
1874 025126 105777 154574 2$:  TSTB  @DACSR             ;WAIT FOR Z PULSE COMPLETION
1875 025132 100375
1876 025134 017737 154614 001126      MOV   @BAR,$BDDAT        ;READ EXPECTED REGISTER VALUE
1877 025142 023737 001124 001126      CMP   $GDDAT,$BDDAT      ;COMPARE
1878 025150 001401
1879 025152 104010
1880

```

```

1882      ;*****
(3)      ;*TEST 150      VERIFY THAT CAMERA 11 CHANNEL IS OPERATIONAL      (TESTER ONLY)
(3)      ;*****
(2) 025154 000004      TST150: SCOPE
(1) 025156 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
1883 025164 005737 050026      TST      WFMODE      ;TEST IF TESTER MODE
1884 025170 001443      BEQ      TST151      ;;BR IF NOT
1885 025172 012737 000001 001124      MOV      #1,$GDDAT      ;LOAD EXPECTED VALUE
1886
1887 025200 012777 004000 154550 1$:      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
1888 025206 012777 000010 154542      MOV      #TSTDMA,@SFR      ;SET TEST DMA FLOP
1889 025214 012777 000000 154530      MOV      #0,@WCR      ;LOAD UPPER 16 BITS
1890 025222 012777 000000 154524      MOV      #0,@BAR      ;LOAD LOWER 16 BITS
1891 025230 012777 001422 154510      MOV      #BIT9!BIT8!BIT4!BIT1,@CSR      ;ENTER MATRIX MODE
1892 025236 052777 000001 154502      BIS      #BIT0,@CSR      ;ENABLE THE DEVICE
1893 025244 052777 000001 154454      BIS      #BIT0,@DACSR      ;GENERATE 'Z' PULSE
1894 025252 105777 154450      2$:      TSTB      @DACSR      ;WAIT FOR Z PULSE COMPLETION
1895 025256 100375      BPL      2$
1896 025260 017737 154470 001126      MOV      @BAR,$BDDAT      ;READ EXPECTED REGISTER VALUE
1897 025266 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1898 025274 001401      BEQ      TST151      ;;BR IF SAME
1899 025276 104010      ERROR 10      ;LOWER 16 BITS OF THE Z COUNTER
1900      ;IN ERROR WHEN USING CAMERA 11

```

```

1902      ;*****
(3)      ;*TEST 151      DYNAMIC MATRIX MODE ADDRESS
(3)      ;*****
(2) 025300 000004      TST151: SCOPE
(1) 025302 012737 000002 001160      MOV      #2,$TIMES      ;;DO 2 ITERATIONS
1903      ;CELL INCREMENT OF A FLOATING LOCATION (60000,64000,62000,61000)
1904      ;FILL THE MEMORY BUFFER WITH A KNOWN PATTERN (125252)
1905      ;COLLECT DATA OF 0 AND INCREMENT A TARGET LOCATION
1906      ;VERIFY THAT NO OTHER ADDRESS IS CHANGED
1907      ;AFTER VERIFYING THE WHOLE BUFFER - UPDATE THE OFFSET REGISTER
1908      ;
1909 025310 012737 060000 050032      MOV      #BUFO,CURRENT      ;PRIME THE CURRENT TARGET LOC.
1910 025316 012737 004000 025666      MOV      #BIT11,100$      ;LOAD FORCE BIT
1911 025324 012737 025332 001110      MOV      #2$,$LPERR      ;LOAD RETURN ADDRESS
1912      ;PRIME THE BUFFER WITH A 125252 PATTERN
1913 025332 012700 060000      2$: MOV      #BUFO,R0      ;LOAD POINTER TO BUFFER
1914 025336 012701 125252      MOV      #125252,R1      ;LOAD VALUE
1915 025342 013702 003354      MOV      ADNOKT,R2      ;LOAD BUFFER END ADDRESS
1916 025346 010120      3$: MOV      R1,(R0)+      ;LOAD BUFFER WITH DATA
1917 025350 020002      CMP      R0,R2      ;TEST FOR END
1918 025352 001375      BNE      3$
1919 025354 012777 177770 022450      MOV      #-10,@CURRENT      ;PRIME THE TARGET LOCATION
1920
1921 025362 012777 004000 154366      MOV      #CLRALL,@SFR      ;INIT THE DEVICE
1922 025370 013777 050032 154352      MOV      CURRENT,@OFF      ;LOAD OFFSET TO TARGET
1923 025376 012777 177777 154346      MOV      #-1,@WCR      ;PRIME THE HIGH 16 BIT Z COUNTER
1924 025404 005077 154344      CLR      @BAR      ;CLEAR LOW 16 BIT COUNTER
1925 025410 012777 000004 154340      MOV      #TSTCON,@SFR      ;ENABLE TEST CONNECTOR
1926 025416 012777 000024 154322      MOV      #24,@CSR      ;LOAD MATRIX WORD MODE
1927 025424 052777 000002 154324      BIS      #TESTZ,@SFR      ;SET TEST Z FLOP
1928 025432 052777 000001 154306      BIS      #BIT0,@CSR      ;ENABLE THE NCV11
1929
1930 025440 013700 002012      MOV      CPUDL2,R0      ;PRIME THE DELAY
1931 025444 005001      CLR      R1
1932 025446 032777 060000 154272 4$: BIT      #BIT14!BIT13,@CSR      ;TEST FOR CELL OR Z OVERFLOW
1933 025454 001014      BNE      5$      ;;BR IF EITHER IS SET
1934 025456 005301      DEC      R1      ;DELAY
1935 025460 001372      BNE      4$
1936 025462 005300      DEC      R0      ;DELAY
1937 025464 001370      BNE      4$
1938 025466 017737 154254 001126      MOV      @CSR,$BDDAT      ;READ BAD STATUS
1939 025474 012737 060224 001124      MOV      #BIT14!BIT13+224,$GDDAT ;LOAD EXPECTED STATUS
1940 025502 104002      ERROR 2      ;DYNAMIC MATRIX MODE STATUS ERROR
1941 025504 000471      BR      TST152      ;;
1942 025506 012700 060000      5$: MOV      #BUFO,R0      ;GET BUFFER POINER
1943 025512 023700 050032      6$: CMP      CURRENT,R0      ;TEST IF TARGET ADDRESS
1944 025516 001415      BEQ      7$      ;;BR IF YES
1945 025520 010037 001122      MOV      R0,$BDADR      ;GET BAD ADDRESS FOR TYPE-OUT
1946 025524 011037 001126      MOV      (R0),$BDDAT      ;GET BAD DATA
1947 025530 012737 125252 001124      MOV      #125252,$GDDAT ;LOAD EXPECTED DATA
1948 025536 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE
1949 025544 001420      BEQ      10$      ;;BR IF SAME
1950 025546 104033      ERROR 33      ;CHANGED AN INCORRECT TARGET LOCATION
1951 025550 000447      BR      TST152      ;;
1952 025552 013737 050032 001122 7$: MOV      CURRENT,$BDADR ;LOAD EXPECTED ADDRESS
1953 025560 017737 022246 001126      MOV      @CURRENT,$BDDAT ;LOAD ACTUAL DATA

```

```

1954 025566 012737 177777 001124      MOV    #-1,$GDDAT      ;LOAD EXPECTED DATA
1955 025574 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;COMPARE
1956 025602 001401          BEQ    10$             ;:BR IF SAME
1957 025604 104030          ERROR  30             ;CELL INCREMENT DATA ERROR
1958
1959 025606 005720          10$:  TST    (R0)+        ;BUMP THE POINTER INTO THE BUFFER
1960 025610 023700 003354      CMP    ADNOKT,R0      ;TEST IF FINISHED THE BUFFER
1961 025614 001336          BNE    6$             ;BR AND RETEST THE REST OF THE BUFFER
1962
1963 025616 005737 001202          TST    $PASS          ;TEST IF FIRST PASS
1964 025622 001422          BEQ    TST152         ;:BR IF YES
1965 025624 032777 004000 153306      BIT    #SW11,@SWR     ;TEST INHIBIT INTER.
1966 025632 001016          BNE    TST152         ;:BR IF SET
1967 025634 006237 025666      ASR    100$           ;CHANGE THE FORCED ADDR. BIT
1968 025640 022737 000002 025666      CMP    #2,100$        ;TEST IF FINISHED
1969 025646 001410          BEQ    TST152         ;:BR IF FINISHED
1970 025650 012737 060000 050032      MOV    #BUFO,CURRENT ;MAKE UP NEW ADDRESS
1971 025656 053737 025666 050032      BIS    100$,CURRENT  ;
1972 025664 000622          BR     2$             ;
1973 025666 004000          100$: BIT11
1974
(3)
(3)
(2) 025670 000004          ;:*****
(1) 025672 012737 000002 001160      TST152: SCOPE        ;*TEST 152 DYNAMIC LIST MODE ADDRESS
1975          MOV    #2,$TIMES    ;:DO 2 ITERATIONS
1976          ;FILL THE MEMORY BUFFER WITH A KNOWN PATTERN (125252)
1977          ;DO A 1 WORD TRANSFER TO A TARGET LOCATION
1978          ;VERIFY THAT NO OTHER LOCATION IN THE BUFFER IS CHANGED
1979          ;AFTER VERIFYING BUMP THE BUS ADDRESS AND DO SUB-TEST AGAIN
1979 025700 012737 060000 050032      MOV    #BUFO,CURRENT ;PRIME THE CURRENT TARGET LOC.
1980 025706 012737 020000 026246      MOV    #BIT13,100$   ;LOAD FORCE BIT
1981 025714 012737 025722 001110      MOV    #2$,$LPERR    ;LOAD RETURN ADDRESS
1982          ;PRIME THE BUFFER WITH A 125252 PATTERN
1983 025722 012700 060000      2$:  MOV    #BUFO,R0    ;LOAD POINTER TO BUFFER
1984 025726 012701 125252      MOV    #125252,R1    ;LOAD VALUE
1985 025732 013702 003354      MOV    ADNOKT,R2     ;LOAD BUFFER END ADDRESS
1986 025736 010120          3$:  MOV    R1,(R0)+    ;LOAD BUFFER WITH DATA
1987 025740 020002          CMP    R0,R2         ;TEST FOR END
1988 025742 001375          BNE    3$
1989
1990 025744 012777 004000 154004      MOV    #CLRALL,@SFR  ;INIT THE DEVICE
1991 025752 012777 177777 153772      MOV    #-1,@WCR      ;PRIME THE WORD COUNT
1992 025760 013777 050032 153766      MOV    CURRENT,@BAR  ;LOAD TARGET LOCATION POINTER
1993 025766 012777 000004 153762      MOV    #TSTCON,@SFR ;ENABLE MAINT. MODE
1994 025774 052777 000001 153744      BIS    #BIT0,@CSR    ;ENABLE THE NCV11
1995 026002 052777 000002 153746      BIS    #TESTZ,@SFR  ;ENABLE "TEST Z" PULSES
(1) 026010 042777 000002 153740      BIC    #TESTZ,@SFR  ;DISABLE "TEST Z" PULSES
1996
1997 026016 013700 002012          MOV    CPUDL2,R0     ;PRIME THE DELAY
1998 026022 005001          CLR    R1
1999 026024 032777 060000 153714      4$:  BIT    #BIT14:BIT13,@CSR ;TEST FOR CELL OR Z OVERFLOW
2000 026032 001014          BNE    5$             ;:BR IF EITHER IS SET
2001 026034 005301          DEC    R1             ;DELAY
2002 026036 001372          BNE    4$
2003 026040 005300          DEC    R0             ;DELAY
2004 026042 001370          BNE    4$

```

2005	026044	017737	153676	001126		MOV	@CSR,\$BDDAT	;READ BAD STATUS
2006	026052	012737	060224	001124		MOV	#BIT14!BIT13+224,\$GDDAT	;LOAD EXPECTED STATUS
2007	026060	104036				ERROR	36	;DYNAMIC LIST MODE STATUS ERROR
2008	026062	000472				BR	TST153	::
2009	026064	012700	060000		5\$:	MOV	#BUFO,RO	;GET BUFFER POINER
2010	026070	023700	050032		6\$:	CMP	CURRENT,RO	;TEST IF TARGET ADDRESS
2011	026074	001415				BEQ	7\$::BR IF YES
2012	026076	010037	001122			MOV	RO,\$BDADR	;GET BAD ADDRESS FOR TYPE-OUT
2013	026102	011037	001126			MOV	(RO),\$BDDAT	;GET BAD DATA
2014	026106	012737	125252	001124		MOV	#125252,\$GDDAT	;LOAD EXPECTED DATA
2015	026114	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;COMPARE
2016	026122	001421				BEQ	10\$::BR IF SAME
2017	026124	104021				ERROR	21	;CHANGED AN INCORRECT TARGET LOCATION
2018	026126	000450				BR	TST153	::
2019	026130	013737	050032	001122	7\$:	MOV	CURRENT,\$BDADR	;LOAD EXPECTED ADDRESS
2020	026136	017737	021670	001126		MOV	@CURRENT,\$BDDAT	;GET ACTUAL DATA
2021	026144	012737	003407	001124		MOV	#3407,\$GDDAT	;LOAD EXPECTED DATA
2022	026152	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;COMPARE
2023	026160	001402				BEQ	10\$::BR IF SAME
2024	026162	104037				ERROR	37	;LIST MODE DATA ERROR
2025	026164	000431				BR	TST153	::
2026	026166	005720			10\$:	TST	(RO)+	;BUMP THE POINTER INTO THE BUFFER
2027	026170	023700	003354			CMP	ADNOKT,RO	;TEST IF FINISHED THE BUFFER
2028	026174	001335				BNE	6\$;BR AND RETEST THE REST OF THE BUFFER
2029								
2030	026176	005737	001202			TST	\$PASS	;TEST IF FIRST PASS
2031	026202	001422				BEQ	TST153	::BR IF FIRST PASS
2032	026204	032777	004000	152726		BIT	#SW11,@SWR	;TEST INHIBIT INTER.
2033	026212	001016				BNE	TST153	::BR IF SET
2034	026214	006237	026246			ASR	100\$;CHANGE THE FORCE ADDRESS
2035	026220	022737	000002	026246		CMP	#2,100\$;TEST IF END
2036	026226	001410				BEQ	TST153	::BR IF FINISHED
2037	026230	012737	060000	050032		MOV	#BUFO,CURRENT	;MAKE NEW ADDRESS
2038	026236	053737	026246	050032		BIS	100\$,CURRENT	:
2039	026244	000626				BR	2\$	
2040	026246	020000			100\$:	BIT13		

```
2042 (3) (3) (2) 026250 000004 (1) 026252 012737 000040 001160 2043 026260 013700 003354 2044 026264 162700 060000 2045 026270 005400 2046 026272 006200 2047 2048 026274 012703 060000 2049 026300 012701 125252 2050 026304 013702 003354 2051 026310 010123 2052 026312 020302 2053 026314 001375 2054 2055 026316 012777 004000 153432 2056 026324 010077 153422 2057 026330 012777 060000 153416 2058 026336 012777 000004 153412 2059 026344 052777 000001 153374 2060 026352 052777 000002 153376 2061 026360 013700 002012 2062 026364 032777 060000 153354 2063 026372 001014 2064 026374 005301 2065 026376 001372 2066 026400 005300 2067 026402 001370 2068 026404 017737 153336 001126 2069 026412 012737 060200 001124 2070 026420 104036 2071 026422 000423 2072 026424 012700 060000 2073 026430 010037 001122 2074 026434 011037 001126 2075 026440 012737 003407 001124 2076 026446 023737 001124 001126 2077 026454 001402 2078 026456 104037 2079 026460 000404 2080 026462 005720 2081 026464 023700 003354 2082 026470 001357
```

```
*****  
*TEST 153 DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K  
*****  
TST153: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV ADNOKT,R0 ;GET LAST ADDRESS  
SUB #BUFO,R0 ;DETERMINE THE WORD COUNT VALUE  
NEG R0  
ASR R0  
;PRIME THE BUFFER WITH A 125252 PATTERN  
2$: MOV #BUFO,R3 ;LOAD POINTER TO BUFFER  
MOV #125252,R1 ;LOAD VALUE  
MOV ADNOKT,R2 ;LOAD BUFFER END ADDRESS  
3$: MOV R1,(R3)+ ;LOAD BUFFER WITH DATA  
CMP R3,R2 ;TEST FOR END  
BNE 3$  
;NOW COLLET THE LIST MODE DATA  
MOV #CLRALL,@SFR ;INIT THE DEVICE  
MOV RO,@WCR ;PRIME THE WORD COUNT  
MOV #BUFO,@BAR ;LOAD TARGET LOCATION POINTER  
MOV #TSTCON,@SFR ;ENABLE MAINT. MODE  
BIS #BIT0,@CSR ;ENABLE THE NCV11  
BIS #TESTZ,@SFR ;ENABLE MAINT. Z  
MOV CPUDL2,R0 ;PRIME THE DELAY  
4$: BIT #BIT14!BIT13,@CSR ;TEST FOR CELL OR Z OVERFLOW  
BNE 5$ ;;BR IF EITHER IS SET  
DEC R1 ;DELAY  
BNE 4$  
DEC RO ;DELAY  
BNE 4$  
MOV @CSR,$BDDAT ;READ BAD STATUS  
MOV #BIT14!BIT13+200,$GDDAT ;LOAD EXPECTED STATUS  
ERROR 36 ;DYNAMIC LIST MODE STATUS ERROR - NO WORD COUNT OVERFLOW  
BR TST154  
5$: MOV #BUFO,R0 ;GET BUFFER POINER  
6$: MOV RO,$BDADR ;GET BAD ADDRESS FOR TYPE-OUT  
MOV (RO),$BDDAT ;GET BAD DATA  
MOV #3407,$GDDAT ;LOAD EXPECTED DATA  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ 10$ ;;BR IF SAME  
ERROR 37 ;LIST MODE DATA ERROR  
BR TST154  
10$: TST (RO)+ ;BUMP THE POINTER INTO THE BUFFER  
CMP ADNOKT,R0 ;TEST IF FINISHED THE BUFFER  
BNE 6$ ;BR AND RETEST THE REST OF THE BUFFER
```



```

2084      ;*****
(3)      ;*TEST 154      ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY
(3)      ;*****
(2) 026472 000004      TST154: SCOPE
(1) 026474 012737 000002 001160      MOV #2,$TIMES      ;DO 2 ITERATIONS
2085 026502 005737 036124      TST $KT11      ;TEST IF KT-11 INSTALLED
2086 026506 100127      BPL TST155      ;BR IF NONE
2087      ;DO A BYTE MATRIX MODE TRANSFER TO A 1 BYTE LOCATION IN EACH 4K EXTENDED
2088      ;MEMORY BANK
2089 026510 012737 000600 041744      MOV #600,OUT      ;LOAD INITIAL BANK VALUE
2090 026516 013737 041744 172346 1$: MOV OUT,@#KIPAR3      ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
2091 026524 012700 060000      MOV #BUF0,R0      ;LOAD BUFFER POINTER
2092 026530 012710 125252      MOV #125252,(R0)      ;PRIME BAD TARGET LOC. IF EXT ADD FAILS
2093 026534 052737 001400 177572      BIS #1400,@#SRO      ;ENABLE MAINT. MODE KT-11
2094 026542 012710 000370      MOV #370,(R0)      ;LOAD TARGET LOCATION DATA VALUE
2095 026546 042737 001400 177572      BIC #1400,@#SRO      ;DISABLE MAINT. MODE KT-11
2096 026554 013701 041744      MOV OUT,R1      ;GET BANK VALUE
2097 026560 005002      CLR R2      ;CLEAR EXT. ADD. TEMPORARY
2100 026562 006301      ASL R1      ;MOVE LEFT
(1) 026564 006301      ASL R1      ;MOVE LEFT
(1) 026566 006301      ASL R1      ;MOVE LEFT
(1) 026570 006301      ASL R1      ;MOVE LEFT
2101 026572 006301      ASL R1
2102 026574 006102      ROL R2      ;SAVE EA BITS
2103 026576 006301      ASL R1
2104 026600 006102      ROL R2      ;SAVE EA BITS
2105 026602 010237 041730      MOV R2,NARROW      ;SAVE EA BITS FOR TYPEOUT
2106 026606 010137 001122      MOV R1,$BDADR      ;SAVE ADDRESS BITS FOR TYPEOUT
2107 026612 060201      ADD R2,R1      ;MAKE COMPLETE ADDRESS
2108
2109      ;NOW GET READY TO DO THE TRANSFER
2110 026614 012777 004000 153134      MOV #CLRALL,@SFR      ;INIT THE NCV11
2111 026622 010177 153122      MCV R1,@OFF      ;LOAD COMBINED BUFFER ADDRESS
2112 026626 012777 000022 153112      MOV #BIT4!BIT1,@CSR      ;ENABLE THE NCV11
2113 026634 012777 000014 153114      MOV #TSTDMA!TSTCON,@SFR      ;ENABLE MAINT NCV11 MODE
2114 026642 052777 000001 153076      BIS #BIT0,@CSR      ;ENABLE THE NCV11
2115 026650 052777 000002 153100      BIS #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES
(1) 026656 042777 000002 153072      BIC #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES
2116 026664 012737 000371 001124      MOV #371,$GDDAT      ;LOAD EXPECTED DATA
2117 026672 052777 010000 153056      BIS #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER
2118 026700 052737 000001 177572      BIS #BIT0,@#SRO      ;ENABLE KT-11
2119 026706 011037 001126      MOV (R0),$BDDAT      ;GET ACTUAL DATA
2120 026712 042737 000001 177572      BIC #BIT0,@#SRO      ;DISABLE KT-11
2121 026720 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE DATA
2122 026726 001401      BEQ 2$      ;BR IF SAME
2123 026730 104035      ERROR 35      ;DATA TRANSFER ERROR TO EXTENDED MEMORY
2124 026732 005737 001202 2$: TST $PASS      ;TEST PASS COUNTER
2125 026736 001413      BEQ TST155      ;BR IF FIRST PASS
2126 026740 032777 004000 152172      BIT #SW11,@SWR      ;TEST IF INHIBIT INTER.
2127 026746 001007      BNE TST155      ;BR IF SET
2128 026750 062737 000200 041744      ADD #200,OUT      ;UPDATE BANK VALUE
2129 026756 023737 036406 041744      CMP $LSTBK,OUT      ;TEST IF DONE
2130 026764 101254      BHI 1$      ;BR IF NOT
  
```

```

2132      ;*****
(3)      ;*TEST 155      ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY
(3)      ;*****
(2) 026766 000004      TST155: SCOPE
(1) 026770 012737 000002 001160      MOV #2,$TIMES      ;;DO 2 ITERATIONS
2133 026776 005737 036124      TST $KT11      ;TEST IF KT-11 INSTALLED
2134 027002 100130      BPL TST156      ;;BR IF NONE
2135      ;DO A 1 WORD MODE TRANSFER TO A 1 WORD BUFFER IN EACH 4K EXTENDED MEMORY BANK
2136 027004 012737 000600 041744      MOV #600,OUT      ;LOAD INITIAL BANK VALUE <60000>
2137 027012 013737 041744 172346 1$:      MOV OUT,@#KIPAR3      ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
2138 027020 012700 060000      MOV #BUFO,R0      ;LOAD BUFFER POINTER
2139 027024 012710 052525      MOV #52525,(R0)      ;LOAD BAD TARGET LOC. IF EXT ADD FAILS
2140 027030 052737 001400 177572      BIS #1400,@#SRO      ;ENABLE MAINT. MODE KT-11
2141 027036 012710 125252      MOV #125252,(R0)      ;PRESET THE TARGET LOCATION
2142 027042 042737 001400 177572      BIC #1400,@#SRO      ;DISABLE MAINT. MODE KT-11
2143 027050 013701 041744      MOV OUT,R1      ;GET BANK VALUE
2144 027054 005002      CLR R2      ;CLEAR EXT. ADD TEMPORARY
2147 027056 006301      ASL R1
(1) 027060 006301      ASL R1
(1) 027062 006301      ASL R1
(1) 027064 006301      ASL R1
2148 027066 006301      ASL R1
2149 027070 006102      ROL R2      ;SAVE EXT. ADDRESS BITS
2150 027072 006301      ASL R1
2151 027074 006102      ROL R2      ;SAVE EXT. ADDRESS BITS
2152
2153      ;NOW GET READY TO DO THE TRANSFER
2154 027076 012777 004000 152652      MOV #CLRALL,@SFR      ;INIT THE NCV11
2155 027104 010277 152640      MOV R2,@OFF      ;LOAD THE EXTENDED ADDRESS BITS
2156 027110 012777 177777 152634      MOV #-1,@WCR      ;LOAD WORD COUNT
2157 027116 010177 152632      MOV R1,@BAR      ;LOAD BUS ADDRESS
2158 027122 012777 000014 152626      MOV #TSTDMA!TSICON,@SFR      ;ENABLE MAINT NCV11 MODE
2159 027130 052777 000001 152610      BIS #BIT0,@CSR      ;ENABLE THE NCV11
2160 027136 052777 000002 152612      BIS #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES
(1) 027144 042777 000002 152604      BIC #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES
2161 027152 010137 001122      MOV R1,$BDADR      ;SAVE TARGET ADDRESS
2162 027156 010237 041730      MOV R2,NARROW      ;SAVE EA BITS FOR ERROR TYPEOUT
2163 027162 052777 010000 152566      BIS #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER
2164 027170 012737 003407 001124      MOV #3407,$GDDAT      ;LOAD EXPECTED DATA
2165 027176 052737 000001 177572      BIS #BIT0,@#SRO      ;ENABLE KT-11
2166 027204 011037 001126      MOV (R0),$BDDAT      ;GET ACTUAL DATA
2167 027210 042737 000001 177572      BIC #BIT0,@#SRO      ;DISABLE KT-11
2168 027216 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE DATA
2169 027224 001401      BEQ 2$      ;;BR IF SAME
2170 027226 104035      ERROR 35      ;LIST MODE DATA TRANSFER ERROR TO EXTENDED MEMORY
2171 027230 005737 001202 2$:      TST $PASS      ;TEST PASS COUNTER
2172 027234 001413      BEQ TST156      ;;BR IF FIRST PASS
2173 027236 032777 004000 151674      BIT #SW11,@SWR      ;TEST INHIBIT INTER.
2174 027244 001007      BNE TST156      ;;BR IF SET
2175 027246 062737 000200 041744      ADD #200,OUT      ;UPDATE BANK VALUE
2176 027254 023737 036406 041744      CMP $LSTBK,OUT      ;TEST IF DONE
2177 027262 101253      BHI 1$      ;BR IF NOT
  
```

```

2179          ;*****
(3)          ;*TEST 156      VERIFY BIT 15 MATRIX ADDER INPUT
(3)          ;*****
(2) 027264 000004          TST156: SCOPE
(1) 027266 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
2180          ;ADM OUTPUT      =      127657
2181          ;OFFSET          =      140000
2182          ;TARGET          =      267657
2183 027274 023727 036406 003140      CMP      $LSTBK,#3140    ;TEST IF >52K IS AVAILABLE
2184 027302 103476          BLO      TST157          ;;BR IF NO MORE MEMORY
2185 027304 012737 002676 172346      MOV      #2676,@#KIPAR3 ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
2186 027312 012700 060056          MOV      #BUF0+56,R0    ;LOAD BUFFER POINTER
2187 027316 012710 125252          MOV      #125252,(R0)   ;LOAD BAD TARGET LOC. IF EXT ADD FAILS
2188 027322 052737 001400 177572      BIS      #1400,@#SRO    ;ENABLE MAINT. MODE KT-11
2189 027330 005010          CLR      (R0)           ;LOAD TARGET LOCATION DATA VALUE
2190 027332 042737 001400 177572      BIC      #1400,@#SRO    ;DISABLE MAINT. MODE KT-11
2191
2192          ;NOW GET READY TO DO THE TRANSFER
2193 027340 012777 004000 152410      MOV      #CLRALL,@SFR   ;INIT THE NCV11
2194 027346 012777 140000 152374      MOV      #140000,@OFF   ;LOAD COMBINED BUFFER ADDRESS
2195 027354 012777 002036 152364      MOV      #2036,@CSR     ;ENABLE THE NCV11
2196 027362 012777 000014 152366      MOV      #TSTDMA!TSTCON,@SFR ;ENABLE MAINT NCV11 MODE
2197 027370 052777 000001 152350      BIS      #BIT0,@CSR     ;ENABLE THE NCV11
2198 027376 052777 000002 152352      BIS      #TESTZ,@SFR    ;ENABLE "TEST Z" PULSES
(1) 027404 042777 000002 152344      BIC      #TESTZ,@SFR    ;DISABLE "TEST Z" PULSES
2199 027412 000240          NOP
2200 027414 000240          NOP
2201 027416 052777 010000 152332      BIS      #BIT12,@SFR    ;ENABLE 1 BYTE TRANSFER
2202 027424 000240          NOP
2203 027426 000240          NOP
2204 027430 000240          NOP
2205 027432 012737 000001 041730      MOV      #1,NARROW     ;SAVE EA BITS FOR ERROR TYPEOUT
2206 027440 012737 000400 001124      MOV      #400,$GDDAT    ;LOAD EXPECTED DATA
2207 027446 052737 000001 177572      BIS      #BIT0,@#SRO    ;ENABLE KT-11
2208 027454 011037 001126          MOV      (R0),$BDDAT    ;SAVE ACTUAL DATA
2209 027460 042737 000001 177572      BIC      #BIT0,@#SRO    ;DISABLE KT-11
2210 027466 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE DATA
2211 027474 001401          BEQ      TST157         ;;BR IF SAME
2212 027476 104020          ERROR 20              ;BIT 15 INPUT TO MATRIX MODE ADDER FAILED
  
```

```
2214      ::*****  
(3)      :*TEST 157   VERIFY HIGH BYTE OPERATION OF THE "TEST X"  
(3)      :*****  
(2) 027500 000004      TST157: SCOPE  
(1) 027502 012737 000040 001160      MOV #40,$TIMES      ;;DO 40 ITERATIONS  
2215      :TEST X      = 1  
2216      :OFFSET      =100000  
2217      :ADM OUTPUT  =127657  
2218      :TARGET      =100257  
2219 027510 012777 004000 152240      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE  
2220 027516 023727 036406 001140      CMP $LSTBK,#1140      ;TEST IF LEAST 20K  
2221 027524 103445      BLO TST160      ;;BR IF NOT  
2222 027526 012777 100000 152214      MOV #100000,@OFF      ;SET BIT 15 OF OFFSET  
2223 027534 012777 000034 152214      MOV #TSTDMA!TSTCON!BIT4,@SFR      ;SET TEST DMA, CONTROL AND TESTX  
2224 027542 012777 002036 152176      MOV #2036,@CSR      ;ENABLE THE NCV11  
2225 027550 052777 000001 152170      BIS #BIT0,@CSR      ;ENABLE THE NCV11  
2226 027556 052777 000002 152172      BIS #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES  
(1) 027564 042777 000002 152164      BIC #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES  
2227 027572 005037 100256      CLR @#BUFO+20256      ;CLEAR THE TARGET LOC.  
2228 027576 052777 010000 152152      BIS #BIT12,@SFR      ;ALLOW 1 DMA TRANSFER  
2229 027604 012737 100256 001122      MOV #BUFO+20256,$BDADR      ;LOAD EXPECTED ADDRESSES VALUE  
2230 027612 012737 000400 001124      MOV #400,$GDDAT      ;LOAD EXPECTED VALUE READ  
2231 027620 017737 151276 001126      MOV @$BDADR,$BDDAT      ;READ ACTUAL DATA  
2232 027626 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE  
2233 027634 001401      BEQ TST160      ;;BR IF SAME  
2234 027636 104034      ERROR 34      ;MATRIX ADDER INPUT OF ADM15 AND TESTX L  
2235      ;FAILED TO BE INHIBITED
```

```

2237      ;:*****
(3)      ;:*TEST 160      VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER
(3)      ;:*****
(2) 027640 000004
(1) 027642 012737 000100 001160
2238      ;ADM OUTPUT      =      253527      ;;DO 100 ITERATIONS
2239      ;OFFSET      =      210000
2240      ;TARGET      =      463527
2241 027650 023727 036406 005140      CMP      $LSTBK,#5140      ;TEST IF ENOUGH MEMORY >84K
2242 027656 103500      BLO      TST161      ;;BR IF NOT
2243 027660 012737 004635 172346      MOV      #4635,@#KIPAR3      ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
2244 027666 012700 060026      MOV      #BUFO+26,R0      ;LOAD BUFFER POINTER
2245 027672 012710 125252      MOV      #125252,(R0)      ;LOAD BAD TARGET LOC.
2246 027676 052737 001400 177572      BIS      #1400,@#SRO      ;ENABLE MAINT. MODE KT-11
2247 027704 005010      CLR      (R0)      ;PRESET THE TARGET LOCATION
2248 027706 042737 001400 177572      BIC      #1400,@#SRO      ;DISABLE MAINT. MODE KT-11
2249
2250      ;NOW GET READY TO DO THE TRANSFER
2251 027714 012777 004000 152034      MOV      #CLRALL,@SFR      ;INIT THE NCV11
2252 027722 012777 010001 152020      MOV      #10001,@OFF      ;LOAD THE EXTENDED ADDRESS BITS
2253 027730 012777 000014 152020      MOV      #TSTDMA!TSTCON,@SFR      ;ENABLE MAINT NCV11 MODE
2254 027736 012777 004036 152002      MOV      #4036,@CSR      ;LOAD MATRIX MODE AND ZB ENABLE
2255 027744 052777 000001 151774      BIS      #BIT0,@CSR      ;ENABLE THE NCV11
2256 027752 052777 000002 151776      BIS      #TESTZ,@SFR      ;ENABLE "TEST Z" PULSES
(1) 027760 042777 000002 151770      BIC      #TESTZ,@SFR      ;DISABLE "TEST Z" PULSES
2257 027766 000240      NOP
2258 027770 000240      NOP
2259 027772 052777 010000 151756      BIS      #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER
2260 030000 000240      NOP
2261 030002 000240      NOP
2262 030004 000240      NOP
2263 030006 012737 000002 041730      MOV      #2,NARROW      ;SAVE EA BITS FOR ERROR TYPEOUT
2264 030014 012737 000400 001124      MOV      #400,$GDDAT      ;LOAD EXPECTED DATA
2265 030022 052737 000001 177572      BIS      #BIT0,@#SRO      ;ENABLE KT-11
2266 030030 111037 001126      MOV      (R0),$BDDAT      ;SAVE ACTUAL DATA
2267 030034 042737 000001 177572      BIC      #BIT0,@#SRO      ;DISABLE KT-11
2268 030042 105037 001126      CLR      $BDDAT      ;MASK OFF LOW BYTE
2269 030046 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA
2270 030054 001401      BEQ      TST161      ;;BR IF SAME
2271 030056 104020      ERROR      20      ;BIT 16 INPUT TO MATRIX MODE ADDER FAILED
  
```

2274
(3)
(3)
(2) 030060 000004
(1) 030062 012737 000001 001160
2275 030070 005737 050024
2276 030074 001402
2277 030076 000137 043550
2278
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 030102
(1) 030102 000004
(1) 030104 005037 001102
(1) 030110 005037 001160
(1) 030114 005237 001202
(1) 030120 042737 100000 001202
(1) 030126 005327
(1) 030130 000001
(1) 030132 003022
(1) 030134 012737
(1) 030136 000001
(1) 030140 030130
(1) 030142 104401 030207
(2) 030146 013746 001202
(2) 030152 104405
(1) 030154 104401 030204
(1) 030160 013700 000042
(1) 030164 001405
(1) 030166 000005
(1) 030170 004710
(1) 030172 000240
(1) 030174 000240
(1) 030176 000240
(1) 030200
(1) 030200 000137
(1) 030202 003360
(1) 030204 377 377 000
(1) 030207 015 042412 042116
(1) 030214 050040 051501 020123
(1) 030222 000043

```
*****  
*TEST 161 DETERMINE IF DIFLIN IS TO BE RUN (F)  
*****  
TST161: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
TST RUNDIF ;;TEST IF DIFLIN IS TO BE RUN  
BEQ $EOP ;;BR IF NOT TO BE RUN  
JMP DIFLIN ;;JUMP AND RUN DIFLIN  
.SBTTL END OF PASS ROUTINE  
  
*****  
*INCREMENT THE PASS NUMBER ($PASS)  
*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM  
*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)  
*IF THERES A MONITOR GO TO IT  
*IF THERE ISN'T JUMP TO LOGIC  
  
$EOP: SCOPE  
CLR $STNM ;;ZERO THE TEST NUMBER  
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS  
INC $PASS ;;INCREMENT THE PASS NUMBER  
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER  
DEC (PC)+ ;;LOOP?  
$EOPCT: .WORD 1  
BGT $DOAGN ;;YES  
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER  
$ENDCT: .WORD 1  
$EOPCT  
TYPE $SENDMG ;;TYPE 'END PASS #'  
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
TYPE $ENULL ;;TYPE A NULL CHARACTER  
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS  
BEQ $DOAGN ;;BRANCH IF NO MONITOR  
RESET ;;CLEAR THE WORLD  
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR  
NOP ;;SAVE ROOM  
NOP ;;FOR  
NOP ;;ACT11  
$DOAGN: JMP @(PC)+ ;;RETURN  
$RTNAD: .WORD LOGIC  
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING  
$SENDMG: .ASCIZ <15><12>/END PASS #/
```

```
2280 .SBTTL ERROR ASCII MESSAGES
2281
2282 030224 034115 031060 004466 EM1: .ASCIZ /M8026 NCV11 BUS ADDRESS TIMEOUT/
      030232 041516 030526 020061
      030240 052502 020123 042101
      030246 051104 051505 020123
      030254 044524 042515 052517
      030262 000124
2283 030264 034115 031060 004466 EM2: .ASCIZ /M8026 COMMAND-STATUS REGISTER ERROR/
      030272 047503 046515 047101
      030300 026504 052123 052101
      030306 051525 051040 043505
      030314 051511 042524 020122
      030322 051105 047522 000122
2284 030330 034115 031060 004466 EM3: .ASCIZ /M8026 SPECIAL FUNCTION REGISTER ERROR/
      030336 050123 041505 040511
      030344 020114 052506 041516
      030352 044524 047117 051040
      030360 043505 051511 042524
      030366 020122 051105 047522
      030374 000122
2285 030376 034115 031060 004466 EM4: .ASCIZ /M8026 WORD COUNT REGISTER ERROR/
      030404 047527 042122 041440
      030412 052517 052116 051040
      030420 043505 051511 042524
      030426 020122 051105 047522
      030434 000122
2286 030436 034115 031060 004466 EM5: .ASCIZ /M8026 BUS ADDRESS REGISTER ERROR/
      030444 052502 020123 042101
      030452 051104 051505 020123
      030460 042522 044507 052123
      030466 051105 042440 051122
      030474 051117 000
2287 030477 115 030070 033062 EM6: .ASCIZ /M8026 OFFSET REGISTER ERROR/
      030504 047411 043106 042523
      030512 020124 042522 044507
      030520 052123 051105 042440
      030526 051122 051117 000
2288 030533 115 030070 033062 EM7: .ASCIZ /M8026 DUAL REGISTER SELECTION ERROR/
      030540 042011 040525 020114
      030546 042522 044507 052123
      030554 051105 051440 046105
      030562 041505 044524 047117
      030570 042440 051122 051117
      030576 000
2289 030577 115 030070 033062 EM10: .ASCIZ /M8026-M8036 LOW 16 BIT Z COUNT ERROR/
      030604 046455 030070 033063
      030612 046011 053517 030440
      030620 020066 044502 020124
      030626 020132 047503 047125
      030634 020124 051105 047522
      030642 000122
2290 030644 034115 031060 026466 EM11: .ASCIZ /M8026-M8036 HIGH 16 BIT Z COUNT ERROR/
      030652 034115 031460 004466
      030660 044510 044107 030440
      030666 020066 044502 020124
```

	030674	020132	047503	047125			
	030702	020124	051105	047522			
	030710	000122					
2291	030712	034115	031060	004466	EM12:	.ASCIIZ /M8026	Z COUNT STATUS ERROR/
	030720	020132	047503	047125			
	030726	020124	052123	052101			
	030734	051525	042440	051122			
	030742	051117	000				
2292	030745	115	030070	033062	EM13:	.ASCIIZ /M8026	Z COUNT INTERRUPT ERROR/
	030752	055011	041440	052517			
	030760	052116	044440	052116			
	030766	051105	052522	052120			
	030774	042440	051122	051117			
	031002	000					
2293	031003	115	030070	033063	EM14:	.ASCIIZ /M8036	JOYSTICK STATUS ERROR/
	031010	045011	054517	052123			
	031016	041511	020113	052123			
	031024	052101	051525	042440			
	031032	051122	051117	000			
2294	031037	115	030070	033063	EM15:	.ASCIIZ /M8036	JOYSTICK DATA ERROR/
	031044	045011	054517	052123			
	031052	041511	020113	040504			
	031060	040524	042440	051122			
	031066	051117	000				
2295	031071	115	030070	033063	FM16:	.ASCIIZ /M8036	DATA INCREMENT ERROR/
	031076	042011	052101	020101			
	031104	047111	051103	046505			
	031112	047105	020124	051105			
	031120	047522	000122				
2296	031124	034115	031460	004466	EM17:	.ASCIIZ /M8036	DATA DECREMENT ERROR/
	031132	040504	040524	042040			
	031140	041505	042522	042515			
	031146	052116	042440	051122			
	031154	051117	000				
2297	031157	115	030070	033062	EM20:	.ASCIIZ /M8026-M8036	MATRIX MODE ADDRESS MAKER DATA ERROR/
	031164	046455	030070	033063			
	031172	046411	052101	044522			
	031200	020130	047515	042504			
	031206	040440	042104	042522			
	031214	051523	046440	045501			
	031222	051105	042040	052101			
	031230	020101	051105	047522			
	031236	000122					
2298	031240	034115	031060	004466	EM21:	.ASCIIZ /M8026	LIST MODE ADDRESS MAKER DATA ERROR/
	031246	044514	052123	046440			
	031254	042117	020105	042101			
	031262	051104	051505	020123			
	031270	040515	042513	020122			
	031276	040504	040524	042440			
	031304	051122	051117	000			
2299	031311	115	030070	033062	EM22:	.ASCIIZ /M8026	LIST MODE TRANSFER BUS ADDRESS ERROR/
	031316	046011	051511	020124			
	031324	047515	042504	052040			
	031332	040522	051516	042506			
	031340	020122	052502	020123			
	031346	042101	051104	051505			

	031354	020123	051105	047522				
	031362	000122						
2300	031364	034115	031060	004466	EM23:	.ASCIIZ	/M8026	LIST MODE TRANSFER WOPD COUNT ERROR/
	031372	044514	052123	046440				
	031400	042117	020105	051124				
	031406	047101	043123	051105				
	031414	053440	051117	020104				
	031422	047503	047125	020124				
	031430	051105	047522	000122				
2301	031436	034115	031060	004466	EM24:	.ASCIIZ	/M8026	LIST MODE TRANSFER OFFSET ERROR/
	031444	044514	052123	046440				
	031452	042117	020105	051124				
	031460	047101	043123	051105				
	031466	047440	043106	042523				
	031474	020124	051105	047522				
	031502	000122						
2302	031504	034115	031060	004466	EM25:	.ASCIIZ	/M8026	TIMEOUT STATUS ERROR/
	031512	044524	042515	052517				
	031520	020124	052123	052101				
	031526	051525	042440	051122				
	031534	051117	000					
2303	031537	115	030070	033062	EM26:	.ASCIIZ	/M8026	TIMEOUT INTERRUPT ERROR/
	031544	052011	046511	047505				
	031552	052125	044440	052116				
	031560	051105	052522	052120				
	031566	042440	051122	051117				
	031574	000						
2304	031575	115	030070	033062	EM27:	.ASCIIZ	/M8026	SET "EVENT" OR "TIME" DATA ERROR/
	031602	051411	052105	021040				
	031610	053105	047105	021124				
	031616	047440	020122	052042				
	031624	046511	021105	042040				
	031632	052101	020101	051105				
	031640	047522	000122					
2305	031644	034115	031060	004466	EM30:	.ASCIIZ	/M8026	CELL INCREMENT DATA ERROR/
	031652	042503	046114	044440				
	031660	041516	042522	042515				
	031666	052116	042040	052101				
	031674	020101	051105	047522				
	031702	000122						
2306	031704	034115	031060	004466	EM31:	.ASCIIZ	/M8026	CELL OVERFLOW STATUS ERROR/
	031712	042503	046114	047440				
	031720	042526	043122	047514				
	031726	020127	052123	052101				
	031734	051525	042440	051122				
	031742	051117	000					
2307	031745	115	030070	033062	EM32:	.ASCIIZ	/M8026	CELL OVERFLOW INTERRUPT ERROR/
	031752	041411	046105	020114				
	031760	053117	051105	046106				
	031766	053517	044440	052116				
	031774	051105	052522	052120				
	032002	042440	051122	051117				
	032010	000						
2308	032011	115	030070	033062	EM33:	.ASCIIZ	/M8026	MATRIX MODE ADDRESS MUX ERROR/
	032016	046411	052101	044522				
	032024	020130	047515	042504				

	032032	040440	042104	042522				
	032040	051523	046440	054125				
	032046	042440	051122	051117				
	032054	000						
2309	032055	115	030070	033062	EM34:	.ASCIIZ	/M8026	"TESTX" FUNCTION ERROR/
	032062	021011	042524	052123				
	032070	021130	043040	047125				
	032076	052103	047511	020116				
	032104	051105	047522	000122				
2310	032112	034115	031060	004466	EM35:	.ASCIIZ	/M8026	DATA ERROR WHEN TRANSFERRING TO EXTENDED MEMORY/
	032120	040504	040524	042440				
	032126	051122	051117	053440				
	032134	042500	020116	051124				
	032142	047101	043123	051105				
	032150	047111	020107	047524				
	032156	042440	052130	047105				
	032164	042504	020104	042515				
	032172	047515	054522	000				
2311	032177	115	030070	033062	EM36:	.ASCIIZ	/M8026	LIST MODE TRANSFER STATUS ERROR/
	032204	046011	051511	020124				
	032212	047515	042504	052040				
	032220	040522	051516	042506				
	032226	020122	052123	052101				
	032234	051525	042440	051122				
	032242	051117	000					
2312	032245	115	030070	033062	EM37:	.ASCIIZ	/M8026	LIST MODE TRANSFER DATA ERROR/
	032252	046011	051511	020124				
	032260	047515	042504	052040				
	032266	040522	051516	042506				
	032274	020122	040504	040524				
	032302	042440	051122	051117				
	032310	000						
2313	032311	112	046525	042520	EM40:	.ASCIIZ	/JUMPER-M8026-M7952	"EVENT" OR "TIME" MARK ERROR/
	032316	026522	034115	031060				
	032324	026466	033515	032471				
	032332	004462	042442	042526				
	032340	052116	020042	051117				
	032346	021040	044524	042515				
	032354	020042	040515	045522				
	032362	042440	051122	051117				
	032370	000						
2314	032371	115	034467	031065	EM41:	.ASCIIZ	/M7952	CLOCK BUS ADDRESS TIMEOUT/
	032376	041411	047514	045503				
	032404	041040	051525	040440				
	032412	042104	042522	051523				
	032420	052040	046511	047505				
	032426	052125	000					
2315	032431	115	031070	033461	EM42:	.ASCIIZ	/M8217	INCORRECT INTERRUPT LEVEL/
	032436	044411	041516	051117				
	032444	042522	052103	044440				
	032452	052116	051105	052522				
	032460	052120	046040	053105				
	032466	046105	000					
2316								
2317	032471	105	051122	041520	DH1:	.ASCIIZ	/ERRPC	ADDR/
	032476	040411	042104	000122				

2318	032504	051105	050122	004503	DH2:	.ASCIZ	/ERRPC	ADDR	GOOD	BAD/
	032512	042101	051104	043411						
	032520	047517	004504	040502						
	032526	000104								
2319	032530	051105	050122	004503	DH3:	.ASCIZ	/ERRPC	ADDR	GOOD	BAD BADADR/
	032536	042101	051104	043411						
	032544	047517	004504	040502						
	032552	004504	040502	040504						
	032560	051104	000							
2320	032563	105	051122	041520	DH4:	.ASCIZ	/ERRPC	ADDR	GOOD	BAD EA ADR BADADR/
	032570	040411	042104	004522						
	032576	047507	042117	041011						
	032604	042101	042411	020101						
	032612	042101	004522	040502						
	032620	040504	051104	000						
2321	032625	015	012	007	OUTRNG:	.BYTE	15,12,7			
2322	032630	047516	021040	021132		.ASCII	/NO 'Z' PULSES OR /			
	032636	050040	046125	042523						
	032644	020123	051117	040						
2323	032651	111	050116	052125		.ASCII	/INPUT VOLTAGE OUT OF RANGE ON CHANNEL #/			
	032656	053040	046117	040524						
	032664	042507	047440	052125						
	032672	047440	020106	040522						
	032700	043516	020105	047117						
	032706	041440	040510	047116						
	032714	046105	021440							
2324	032720	060	015	012	OUTCHN:	.BYTE	60,15,12,7,0			
	032723	007	000							
2325		032726				.EVEN				
2326	032726	001116	001250	000000	DT1:	.WORD	\$ERRPC,\$BASE,0			
2327	032734	001116	001250	001124	DT2:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,0			
	032742	001126	000000							
2328	032746	001116	001250	001124	DT3:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,\$BDADR,0			
	032754	001126	001122	000000						
2329	032762	001116	001250	001124	DT4:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,NARROW,\$BDADR,0			
	032770	001126	041730	001122						
	032776	000000								
2330	033000	001116	001254	000000	DT5:	.WORD	\$ERRPC,\$CDW1,0			
2331	033006	000000			DF0:	.WORD	0			

2336

.SBTTL TTY INPUT ROUTINE

```
(1) .ENABL LSB  
(2) *****  
(1) *****  
(1) *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.  
(1) *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL  
(1) *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL  
(1) *WHEN OPERATING IN TTY FLAG MODE.  
(1) 033010 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?  
(1) 033016 001074 BNE 15$ ;; BRANCH IF NO  
(1) 033020 105777 146120 TSTB @$TKS ;; CHAR THERE?  
(1) 033024 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND  
(1) 033026 117746 146114 MOV B @TKB,-(SP) ;; SAVE THE CHAR  
(1) 033032 042716 177600 BIC #^C177,(SP) ;; STRIP-OFF THE ASCII  
(1) 033036 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?  
(1) 033042 001062 BNE 15$ ;; NO, RETURN TO USER  
(1) 033044 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?  
(1) 033052 001456 BEQ 15$ ;; BRANCH IF YES  
(1) 033054 104401 033535 $GTSWR: TYPE , $CNTLG ;; ECHO THE CONTROL-G (^G)  
(1) 033060 104401 033542 TYPE , $MSWR ;; TYPE CURRENT CONTENTS  
(2) 033064 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT  
(2) 033070 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)  
(1) 033072 104401 033553 TYPE , $MNEW ;; PROMPT FOR NEW SWR  
(1) 033076 005046 19$: CLR -(SP) ;; CLEAR COUNTER  
(1) 033100 005046 CLR -(SP) ;; THE NEW SWR  
(1) 033102 105777 146036 7$: TSTB @$TKS ;; CHAR THERE?  
(1) 033106 100375 BPL 7$ ;; IF NOT TRY AGAIN  
(1) 033110 117746 146032 MOV B @TKB,-(SP) ;; PICK UP CHAR  
(1) 033114 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII  
(1)  
(1)  
(1) 033120 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?  
(1) 033124 001005 BNE 10$ ;; BRANCH IF NOT  
(1) 033126 104401 033530 TYPE , $CNTLU ;; YES, ECHO CONTROL-U (^U)  
(1) 033132 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT  
(1) 033136 000757 BR 19$ ;; LET'S TRY IT AGAIN  
(1)  
(1) 033140 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?  
(1) 033144 001022 BNE 16$ ;; BRANCH IF NO  
(1) 033146 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?  
(1) 033152 001403 BEQ 11$ ;; BRANCH IF YES  
(1) 033154 016677 000002 145756 MOV 2(SP),@SWR ;; SAVE NEW SWR  
(1) 033162 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK  
(1) 033166 104401 001171 14$: TYPE , $CRLF ;; ECHO <CR> AND <LF>  
(1) 033172 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?  
(1) 033200 001003 BNE 15$ ;; BRANCH IF NOT  
(1) 033202 012777 000100 145734 MOV #100,@TKS ;; RE-ENABLE TTY KBD INTERRUPTS  
(1) 033210 000002 15$: RTI ;; RETURN  
(1) 033212 004737 034610 16$: JSR PC,$TYPEC ;; ECHO CHAR  
(1) 033216 021627 000060 CMP (SP),#60 ;; CHAR < 0?
```

```

(1) 033222 002420          BLT      18$          ;;BRANCH IF YES
(1) 033224 021627 000067  CMP      (SP),#67      ;;CHAR > 7?
(1) 033230 003015          BGT      18$          ;;BRANCH IF YES
(1) 033232 042726 000060  BIC      #60,(SP)+    ;;STRIP-OFF ASCII
(1) 033236 005766 000002  TST      2(SP)        ;;IS THIS THE FIRST CHAR
(1) 033242 001403          BEQ      17$          ;;BRANCH IF YES
(1) 033244 006316          ASL      (SP)         ;;NO, SHIFT PRESENT
(1) 033246 006316          ASL      (SP)         ;;CHAR OVER TO MAKE
(1) 033250 006316          ASL      (SP)         ;;ROOM FOR NEW ONE.
(1) 033252 005266 000002  17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
(1) 033256 056616 177776  BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
(1) 033262 000707          BR       7$          ;;GET THE NEXT ONE
(1) 033264 104401 001170  18$: TYPE  $QUES      ;;TYPE ?<CR><LF>
(1) 033270 000720          BR       20$        ;;SIMULATE CONTROL-U
(1) .DSABL  LSB
(1)
(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) * RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1) * RETURN HERE   ;;CHARACTER IS ON THE STACK
(1) *              ;;WITH PARITY BIT STRIPPED OFF
(1)
(1) $RDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC
(1) 033272 011646 000004 000002 MOV      4(SP),2(SP)  ;;SAVE THE PS
(1) 033274 016666 145636 1$: TSTB     @TKS        ;;WAIT FOR
(1) 033302 105777 145636 BPL      1$          ;;A CHARACTER
(1) 033306 100375          MOVB     @TKB,4(SP)  ;;READ THE TTY
(1) 033310 117766 145632 000004 BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 033316 042766 177600 000004 CMP      4(SP),#23   ;;IS IT A CONTROL-S?
(1) 033324 026627 000004 000023 BNE      3$          ;;BRANCH IF NO
(1) 033332 001013          TSTB     @TKS        ;;WAIT FOR A CHARACTER
(1) 033334 105777 145604 2$: BPL      2$          ;;LOOP UNTIL ITS THERE
(1) 033340 100375          MOVB     @TKB,-(SP)  ;;GET CHARACTER
(1) 033342 117746 145600 BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 033346 042716 177600 CMP      (SP)+,#21   ;;IS IT A CONTROL-G?
(1) 033352 022627 000021 BNE      2$          ;;IF NOT DISCARD IT
(1) 033356 001366          BR       1$          ;;YES, RESUME
(1) 033360 000750          CMP      4(SP),#140 ;;IS IT UPPER CASE?
(1) 033362 026627 000004 000140 3$: BLT      4$          ;;BRANCH IF YES
(1) 033370 002407          CMP      4(SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 033372 026627 000004 000175 BGT      4$          ;;BRANCH IF YES
(1) 033400 003003          BIC      #40,4(SP)  ;;MAKE IT UPPER CASE
(1) 033402 042766 000040 000004 4$: RTI          ;;GO BACK TO USER
(1) 033410 000002
(2)
(1) *****
(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)
(1) $RDLIN: MOV      R3,-(SP)  ;;SAVE R3
(1) 033412 010346 033520 1$: MOV      $TTYIN,R3  ;;GET ADDRESS
(1) 033414 012703 033520 2$: CMP      $TTYIN+8.,R3 ;;BUFFER FULL?
(1) 033420 022703 033530

```

(1)	033424	101405			BLOS	4\$::BR IF YES
(1)	033426	104410			RDCHR			::GO READ ONE CHARACTER FROM THE TTY
(1)	033430	112613			MOVB	(SP)+,(R3)		::GET CHARACTER
(1)	033432	122713	000177	10\$:	CMPB	#177,(R3)		::IS IT A RUBOUT
(1)	033436	001003			BNE	3\$::SKIP IF NOT
(1)	033440	104401	001170	4\$:	TYPE	,SQUES		::TYPE A '?'
(1)	033444	000763			BR	1\$::CLEAR THE BUFFER AND LOOP
(1)	033446	111337	033516	3\$:	MOVB	(R3),9\$::ECHO THE CHARACTER
(1)	033452	104401	033516		TYPE	,9\$		
(1)	033456	122723	000015		CMPB	#15,(R3)+		::CHECK FOR RETURN
(1)	033462	001356			BNE	2\$::LOOP IF NOT RETURN
(1)	033464	105063	177777		CLRB	-1(R3)		::CLEAR RETURN (THE 15)
(1)	033470	104401	001172		TYPE	,SLF		::TYPE A LINE FEED
(1)	033474	012603			MOV	(SP)+,R3		::RESTORE R3
(1)	033476	011646			MOV	(SP),-(SP)		::ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	033500	016666	000004	000002	MOV	4(SP),2(SP)		:: FIRST ASCII CHARACTER ON IT
(1)	033506	012766	033520	000004	MOV	#\$TTYIN,4(SP)		
(1)	033514	000002			RTI			::RETURN
(1)	033516	000		9\$:	.BYTE	0		::STORAGE FOR ASCII CHAR. TO TYPE
(1)	033517	000			.BYTE	0		::TERMINATOR
(1)	033520	000010			\$TTYIN:	.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
(1)	033530	052536	005015	000	\$CNTLU:	.ASCIZ	/^U/<15><12>	::CONTROL 'U'
(1)	033535	136	006507	000012	\$CNTLG:	.ASCIZ	/^G/<15><12>	::CONTROL 'G'
(1)	033542	005015	053523	020122	\$MSWR:	.ASCIZ	<15><12>/SWR = /	
(1)	033550	020075	000					
(1)	033553	040	047040	053505	\$MNEW:	.ASCIZ	/ NEW = /	
(1)	033560	036440	000040					

```

2338      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(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)
(1) 033564 011646      $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1) 033566 016666      MOV      4(SP),2(SP)      ;;INPUT NUMBER
(3) 033574 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 033576 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 033600 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(1) 033602 104411      1$:  RDLIN      ;;READ AN ASCII LINE
(1) 033604 012600      MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
(1) 033606 005001      CLR      R1          ;;CLEAR DATA WORD
(1) 033610 005002      CLR      R2
(1) 033612 112046      2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
(1) 033614 001412      BEQ      3$          ;;IF ZERO GET OUT
(1) 033616 006301      ASL      R1          ;;*2
(1) 033620 006102      ROL      R2
(1) 033622 006301      ASL      R1          ;;*4
(1) 033624 006102      ROL      R2
(1) 033626 006301      ASL      R1          ;;*8
(1) 033630 006102      ROL      R2
(1) 033632 042716      BIC      #^C7,(SP)      ;;STRIP THE ASCII JUNK
(1) 033636 062601      ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
(1) 033640 000764      BR       2$          ;;LOOP
(1) 033642 005726      3$:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
(1) 033644 010166      MOV      R1,12(SP)      ;;SAVE THE RESULT
(1) 033650 010237      MOV      R2,$HIOCT
(3) 033654 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
(3) 033656 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 033660 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 033662 000002      RTI          ;;RETURN
(1) 033664 000000      $HIOCT: .WORD 0      ;;HIGH ORDER BITS GO HERE
  
```

```
2340      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)      ;*****
(1)      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)      ;*REPLACED WITH SPACES.
(1)      ;*CALL:
(1)      ;*
(1)      ;*     MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1)      ;*     TYPDS      ;;GO TO THE ROUTINE
(1)
(1)      $TYPDS:
(3)      033666      010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
(3)      033670      010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
(3)      033672      010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
(3)      033674      010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
(3)      033676      010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
(1)      033700      012746      020200      MOV     #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
(1)      033704      016605      000020      MOV     20(SP),R5    ;;GET THE INPUT NUMBER
(1)      033710      100004      BPL     1$           ;;BR IF INPUT IS POS.
(1)      033712      005405      NEG     R5           ;;MAKE THE BINARY NUMBER POS.
(1)      033714      112766      000055      000001      MOVB    #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
(1)      033722      005000      1$:      CLR     R0           ;;ZERO THE CONSTANTS INDEX
(1)      033724      012703      034102      MOV     #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
(1)      033730      112723      000040      MOVB    #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
(1)      033734      005002      2$:      CLR     R2           ;;CLEAR THE BCD NUMBER
(1)      033736      016001      034072      MOV     $DTBL(R0),R1 ;;GET THE CONSTANT
(1)      033742      160105      3$:      SUB     R1,R5        ;;FORM THIS BCD DIGIT
(1)      033744      002402      BLT     4$           ;;BR IF DONE
(1)      033746      005202      INC     R2           ;;INCREASE THE BCD DIGIT BY 1
(1)      033750      000774      BR      3$
(1)      033752      060105      4$:      ADD     R1,R5        ;;ADD BACK THE CONSTANT
(1)      033754      005702      TST     R2           ;;CHECK IF BCD DIGIT=0
(1)      033756      001002      BNE     5$           ;;FALL THROUGH IF 0
(1)      033760      105716      TSTB   (SP)         ;;STILL DOING LEADING 0'S?
(1)      033762      100407      BMI     7$           ;;BR IF YES
(1)      033764      106316      5$:      ASLB   (SP)         ;;MSD?
(1)      033766      103003      BCC     6$           ;;BR IF NO
(1)      033770      116663      000001      177777      MOVB    1(SP),-1(R3)  ;;YES--SET THE SIGN
(1)      033776      052702      000060      6$:      BIS     #'0,R2       ;;MAKE THE BCD DIGIT ASCII
(1)      034002      052702      000040      7$:      BIS     #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)      034006      110223      MOVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)      034010      005720      TST     (R0)+       ;;JUST INCREMENTING
(1)      034012      020027      000010      CMP     R0,#10      ;;CHECK THE TABLE INDEX
(1)      034016      002746      BLT     2$           ;;GO DO THE NEXT DIGIT
(1)      034020      003002      BGT     8$           ;;GO TO EXIT
(1)      034022      010502      MOV     R5,R2       ;;GET THE LSD
(1)      034024      000764      BR      6$          ;;GO CHANGE TO ASCII
(1)      034026      105726      8$:      TSTB   (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
(1)      034030      100003      BPL     9$           ;;BR IF NO
(1)      034032      116663      177777      177776      MOVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1)      034040      105013      9$:      CLRB   (R3)         ;;SET THE TERMINATOR
(3)      034042      012605      MOV     (SP)+,R5    ;;POP STACK INTO R5
(3)      034044      012603      MOV     (SP)+,R3    ;;POP STACK INTO R3
(3)      034046      012602      MOV     (SP)+,R2    ;;POP STACK INTO R2
```


(3)	034050	012601					MOV	(SP)+,R1	::POP STACK INTO R1
(3)	034052	012600					MOV	(SP)+,R0	::POP STACK INTO R0
(1)	034054	104401	034102				TYPE	\$DBLK	::NOW TYPE THE NUMBER
(1)	034060	016666	000002	000004			MOV	2(SP),4(SP)	::ADJUST THE STACK
(1)	034066	012616					MOV	(SP)+,(SP)	
(1)	034070	000002					RTI		::RETURN TO USER
(1)	034072	023420				\$DTBL:	10000.		
(1)	034074	001750					1000.		
(1)	034076	000144					100.		
(1)	034100	000012					10.		
(1)	034102	000004				\$DBLK:	.BLKW 4		
2341						.SBTTL	SCOPE HANDLER ROUTINE		
(1)							*****		
(2)							*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)							*SW09=1	LOOP ON ERROR	
(1)							*SW08=1	LOOP ON TEST IN SWR<7:0>	
(1)							*CALL		
(1)							* SCOPE	::SCOPE=IOT	
(1)							\$SCOPE:		
(1)	034112						CKSWR		::TEST FOR CHANGE IN SOFT-SWR
(1)	034112	104407					JSR	PC,CTRLCG	::TEST FOR CTRL C OR G
(2)	034114	004737	045700				BIT	#BIT14,@SWR	::LOOP ON PRESENT TEST?
(1)	034120	032777	040000	145012		1\$:	BNE	\$OVER	::YES IF SW14=1
(1)	034126	001114					:####START OF CODE FOR THE XOR TESTER####		
(1)	034130	000416				\$XTSTR:	BR	6\$::IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)									::THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1)	034132	013746	000004				MOV	@#ERRVEC,-(SP)	::SAVE THE CONTENTS OF THE ERROR VECTOR
(1)	034136	012737	034156	000004			MOV	#5\$,@#ERRVEC	::SET FOR TIMEOUT
(1)	034144	005737	177060				TST	@#177060	::TIME OUT ON XOR?
(1)	034150	012637	000004				MOV	(SP)+,@#ERRVEC	::RESTORE THE ERROR VECTOR
(1)	034154	000463					BR	\$SVLAD	::GO TO THE NEXT TEST
(1)	034156	022626				5\$:	CMP	(SP)+,(SP)+	::CLEAR THE STACK AFTER A TIME OUT
(1)	034160	012637	000004				MOV	(SP)+,@#ERRVEC	::RESTORE THE ERROR VECTOR
(1)	034164	000423					BR	7\$::LOOP ON THE PRESENT TEST
(1)	034166					6\$::	:####END OF CODE FOR THE XOR TESTER####		
(1)	034166	032777	000400	144744			BIT	#BIT08,@SWR	::LOOP ON SPEC. TEST?
(1)	034174	001404					BEQ	2\$::BR IF NO
(1)	034176	127737	144736	001102			CMPB	@SWR,\$TSTNM	::ON THE RIGHT TEST? SWR<7:0>
(1)	034204	001465					BEQ	\$OVER	::BR IF YES
(1)	034206	105737	001103			2\$:	TSTB	\$ERFLG	::HAS AN ERROR OCCURRED?
(1)	034212	001421					BEQ	3\$::BR IF NO
(1)	034214	123737	001115	001103			CMPB	\$ERMAX,\$ERFLG	::MAX. ERRORS FOR THIS TEST OCCURRED?
(1)	034222	101015					BHI	3\$::BR IF NO
(1)	034224	032777	001000	144706			BIT	#BIT09,@SWR	::LOOP ON ERROR?
(1)	034232	001404					BEQ	4\$::BR IF NO
(1)	034234	013737	001110	001106		7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
(1)	034242	000446					BR	\$OVER	
(1)	034244	105037	001103			4\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG
(1)	034250	005037	001160				CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE

```
(1) 034254 000415          BR      1$          ;;ESCAPE TO THE NEXT TEST
(1) 034256 032777 004000 144654 3$:  BIT      #BIT11,@SWR  ;;INHIBIT ITERATIONS?
(1) 034264 001011          BNE     1$          ;;BR IF YES
(1) 034266 005737 001202          TST     $PASS      ;;IF FIRST PASS OF PROGRAM
(1) 034272 001406          BEQ     1$          ;;      INHIBIT ITERATIONS
(1) 034274 005237 001104          INC     $ICNT      ;;INCREMENT ITERATION COUNT
(1) 034300 023737 001160 001104          CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 034306 002024          BGE     $OVER      ;;BR IF MORE ITERATION REQUIRED
(1) 034310 012737 000001 001104 1$:  MOV     #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
(1) 034316 013737 034374 001160          MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 034324 105237 001102          $SVLAD: INCB    $TSTNM  ;;COUNT TEST NUMBERS
(1) 034330 113737 001102 001200          MOV     $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 034336 011637 001106          MOV     (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 034342 011637 001110          MOV     (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 034346 005037 001162          CLR     $ESCAPE   ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 034352 112737 000001 001115          MOV     #1,$ERMAX  ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 034360 013777 001102 144554 $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 034366 013716 001106          MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 034372 000002          RTI                    ;;FIXES PS
(1) 034374 003720          $MXCNT: 2000.        ;;MAX. NUMBER OF ITERATIONS
```

2342 .SBTTL TYPE ROUTINE

```
(1)
(2)
(1) *****
(1) *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) *
(1) *CALL:
(1) *1) USING A TRAP INSTRUCTION
(1) *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) *OR
(1) *      TYPE
(1) *      MESADR
(1) *
```

```
(1) 034376 105737 001157 $TYPE: TSTB    $TPFLG  ;;IS THERE A TERMINAL?
(1) 034402 100002          BPL     1$          ;;BR IF YES
(1) 034404 000000          HALT                    ;;HALT HERE IF NO TERMINAL
(1) 034406 000430          BR      3$          ;;LEAVE
(1) 034410 010046          1$:  MOV     R0,-(SP)   ;;SAVE R0
(1) 034412 017600 000002          MOV     @2(SP),R0   ;;GET ADDRESS OF ASCIZ STRING
(1) 034416 122737 000001 001214          CMPB   #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 034424 001011          BNE     62$        ;;NO,GO CHECK FOR APT CONSOLE
(1) 034426 132737 000100 001215          BITB   #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 034434 001405          BEQ     62$        ;;NO,GO CHECK FOR CONSOLE
(1) 034436 010037 034446          MOV     R0,61$     ;;SETUP MESSAGE ADDRESS FOR APT
(1) 034442 004737 035450          JSR    PC,$ATY3   ;;SPOOL MESSAGE TO APT
(1) 034446 000000          61$:  .WORD   0        ;;MESSAGE ADDRESS
(1) 034450 132737 000040 001215          62$:  BITB   #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 034456 001003          BNE     60$        ;;YES,SKIP TYPE OUT
(1) 034460 112046          2$:  MOV     (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 034462 001005          BNE     4$          ;;BR IF IT ISN'T THE TERMINATOR
(1) 034464 005726          TST    (SP)+       ;;IF TERMINATOR POP IT OFF THE STACK
(1) 034466 012600          60$:  MOV     (SP)+,R0   ;;RESTORE R0
```

```

(1) 034470 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
(1) 034474 000002 RTI ;;RETURN
(1) 034476 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
(1) 034502 001430 BEQ 8$
(1) 034504 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
(1) 034510 001006 BNE 5$
(1) 034512 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
(1) 034514 104401 TYPE ;;TYPE A CR AND LF
(1) 034516 001171 $CRLF
(1) 034520 105037 034654 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
(1) 034524 000755 BR 2$ ;;GET NEXT CHARACTER
(1) 034526 004737 034610 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
(1) 034532 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 034536 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
(1) 034540 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(1) ;;AND THE NULL CHAR.
(1) 034544 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
(1) 034550 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 034552 004737 034610 JSR PC,$TYPEC ;;GO TYPE A NULL
(1) 034556 105337 034654 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
(1) 034562 000770 BR 7$ ;;LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 034564 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
(1) 034570 004737 034610 9$: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 034574 132737 000007 034654 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
(1) 034602 001372 BNE 9$ ;;TAB STOP
(1) 034604 005726 TST (SP)+ ;;POP SPACE OFF STACK
(1) 034606 000724 BR 2$ ;;GET NEXT CHARACTER
(1) 034610 105777 144334 $TYPEC: TSTB @$TPS ;;WAIT UNTIL PRINTER IS READY
(1) 034614 100375 BPL $TYPEC
(1) 034616 116677 000002 144326 MOVB 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 034624 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 034632 001003 BNE 1$ ;;BRANCH IF NO
(1) 034634 105037 034654 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 034640 000406 BR $TYPEX ;;EXIT
(1) 034642 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 034650 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 034652 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 034654 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
(1) 034656 000207 $TYPEX: RTS PC
(1)

```



```

(1) 035022 005204          4$: INC R4      ;;DON'T SUPPRESS ANYMORE 0'S
(1) 035024 052703 000060  BIS #'0,R3   ;;MAKE THIS DIGIT ASCII
(1) 035030 052703 000040  5$: BIS #' ,R3  ;;MAKE ASCII IF NOT ALREADY
(1) 035034 110337 035100  MOVB R3,8$   ;;SAVE FOR TYPING
(1) 035040 104401 035100  TYPE ,8$     ;;GO TYPE THIS DIGIT
(1) 035044 105337 035102  7$: DECB $OCNT ;;COUNT BY 1
(1) 035050 003347      BGT 2$       ;;BR IF MORE TO DO
(1) 035052 002402      BLT 6$       ;;BR IF DONE
(1) 035054 005204      INC R4       ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 035056 000744      BR 2$        ;;GO DO THE LAST DIGIT
(1) 035060 012605 6$: MOV (SP)+,R5  ;;RESTORE R5
(1) 035062 012604      MOV (SP)+,R4  ;;RESTORE R4
(1) 035064 012603      MOV (SP)+,R3  ;;RESTORE R3
(1) 035066 016666 000002 000004 MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
(1) 035074 012616      MOV (SP)+,(SP)
(1) 035076 000002      RTI             ;;RETURN
(1) 035100 000        8$: .BYTE 0      ;;STORAGE FOR ASCII DIGIT
(1) 035101 000        .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
(1) 035102 000      $OCNT: .BYTE 0  ;;OCTAL DIGIT COUNTER
(1) 035103 000      $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
(1) 035104 000000      $OMODE: .WORD 0  ;;NUMBER OF DIGITS TO TYPE
  
```

2345
 2346

.SBTTL ERROR HANDLER ROUTINE

```

(1)  ;;*****
(2)  ;;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)  ;;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)  ;;*AND GO TO $ERRTYP ON ERROR
(1)  ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)  ;;*SW15=1 HALT ON ERROR
(1)  ;;*SW13=1 INHIBIT ERROR TYPEOUTS
(1)  ;;*SW10=1 BELL ON ERROR
(1)  ;;*SW09=1 LOOP ON ERROR
(1)  ;;*CALL
(1)  ;;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)  $ERROR:
(1) 035106 104407          7$: CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 035106 104407          INCB $ERFLG   ;;SET THE ERROR FLAG
(1) 035110 105237 001103  BEQ 7$     ;;DON'T LET THE FLAG GO TO ZERO
(1) 035114 001775          MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 035116 013777 001102 144016 BIT #BIT10,@SWR ;;BELL ON ERROR?
(1) 035124 032777 002000 144006 BEQ 1$     ;;NO - SKIP
(1) 035132 001402          TYPE ,SBELL   ;;RING BELL
(1) 035134 104401 001164          INC $ERTTL  ;;COUNT THE NUMBER OF ERRORS
(1) 035140 005237 001112          MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 035144 011637 001116          SUB #2,$ERRPC
(1) 035150 162737 000002 001116 MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 035156 117737 143734 001114 BIT #BIT13,@SWR  ;;SKIP TYPEOUT IF SET
(1) 035164 032777 020000 143746 BNE 20$    ;;SKIP TYPEOUTS
(1) 035172 001004          JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 035174 004737 035306          TYPE ,SCRLF
(1) 035200 104401 001171
(1) 035204
(1) 035204 122737 000001 001214 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 035212 001007          BNE 2$     ;;NO,SKIP APT ERROR REPORT
(1) 035214 113737 001114 035226 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
  
```

```
(1) 035222 004737 035460 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 035226 000 21$: .BYTE 0
(1) 035227 000 .BYTE 0
(1) 035230 000777 BR 22$ ;;APT ERROR LOOP
(1) 035232 005777 143702 2$: TST @SWR ;;HALT ON ERROR
(1) 035236 100002 BPL 3$ ;;SKIP IF CONTINUE
(1) 035240 000000 HALT ;;HALT ON ERROR!
(1) 035242 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 035244 032777 001000 143666 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 035252 001402 BEQ 4$ ;;BR IF NO
(1) 035254 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 035260 005737 001162 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 035264 001402 BEQ 5$ ;;BR IF NONE
(1) 035266 013716 001162 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 035272 5$:
(1) 035272 022737 030170 000042 CMP #$ENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 035300 001001 BNE 6$ ;;BRANCH IF NO
(1) 035302 000000 HALT ;;YES
(1) 035304 6$:
(1) 035304 000002 RTI ;;RETURN
```

2347
2348

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 035306 $ERRTYP:
(1) 035306 104401 001171 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035312 010046 MOV RO,-(SP) ;;SAVE RO
(1) 035314 005000 CLR RO ;;PICKUP THE ITEM INDEX
(1) 035316 153700 001114 BISB @#$ITEMB,RO
(1) 035322 001004 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
(1) 035324 013746 001116 MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(2) ;;SAVE $ERRPC FOR TYPEOUT
(2) ;;ERROR ADDRESS
(2) 035330 104402 TYPCC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 035332 000426 BR 6$ ;;GET OUT
(1) 035334 005300 1$: DEC RO ;;ADJUST THE INDEX SO THAT IT WILL
(1) 035336 006300 ASL RO ;; WORK FOR THE ERROR TABLE
(1) 035340 006300 ASL RO
(1) 035342 006300 ASL RO
(1) 035344 062700 001256 ADD #$ERRTB,RO ;;FORM TABLE POINTER
(1) 035350 012037 035360 MOV (RO)+,2$ ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 035354 001404 BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER
(1) 035356 104401 TYPE ;;TYPE THE 'ERROR MESSAGE'
(1) 035360 000000 2$: .WORD 0 ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 035362 104401 001171 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035366 012037 035376 3$: MOV (RO)+,4$ ;;PICKUP 'DATA HEADER' POINTER
(1) 035372 001404 BEQ 5$ ;;SKIP TYPEOUT IF 0
(1) 035374 104401 TYPE ;;TYPE THE 'DATA HEADER'
(1) 035376 000000 4$: .WORD 0 ;;'DATA HEADER' POINTER GOES HERE
(1) 035400 104401 001171 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035404 011000 5$: MOV (RO),RO ;;PICKUP 'DATA TABLE' POINTER
(1) 035406 001004 BNE 7$ ;;GO TYPE THE DATA
```

(1)	035410	012600		6\$:	MOV	(SP)+,R0	::RESTORE R0
(1)	035412	104401	001171		TYPE	, \$CRLF	::'CARRIAGE RETURN' & 'LINE FEED'
(1)	035416	000207			RTS	PC	::RETURN
(1)	035420			7\$:-			
(2)	035420	013046			MOV	@(R0)+,-(SP)	::SAVE @(R0)+ FOR TYPEOUT
(2)	035422	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)	035424	005710			TST	(R0)	::IS THERE ANOTHER NUMBER?
(1)	035426	001770			BEQ	6\$::BR IF NO
(1)	035430	104401	035436		TYPE	,8\$::TYPE TWO(2) SPACES
(1)	035434	000771			BR	7\$::LOOP
(1)	035436	020040	000	8\$:	.ASCIZ	/ /	::TWO(2) SPACES
(1)	035442				.EVEN		

```

2350      .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)
(1) 035442 112737 000001 035706 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) 035450 112737 000001 035704 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
(1) 035456 000403
(1) 035460 112737 000001 035706 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 035466
(3) 035466 010046                MOV    RO,-(SP)      ;;PUSH RO ON STACK
(3) 035470 010146                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
(1) 035472 105737 035704                TSTB  $MFLG          ;;SHOULD TYPE A MESSAGE?
(1) 035476 001450                BEQ   5$             ;;IF NOT: BR
(1) 035500 122737 000001 001214        CMPB  #APTENV,$ENV   ;;OPERATING UNDER APT?
(1) 035506 001031                BNE   3$             ;;IF NOT: BR
(1) 035510 132737 000100 001215        BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 035516 001425                BEQ   3$             ;;IF NOT: BR
(1) 035520 017600 000004                MOV   @4(SP),RO      ;;GET MESSAGE ADDR.
(1) 035524 062766 000002 000004        ADD   #2,4(SP)       ;;BUMP RETURN ADDR.
(1) 035532 005737 001174                1$:  TST  $MSGTYPE    ;;SEE IF DONE W/ LAST XMISSION?
(1) 035536 001375                BNE   1$             ;;IF NOT: WAIT
(1) 035540 010037 001210                MOV   RO,$MSGAD      ;;PUT ADDR IN MAILBOX
(1) 035544 105720                2$:  TSTB (RO)+       ;;FIND END OF MESSAGE
(1) 035546 001376                BNE   2$
(1) 035550 163700 001210                SUB   $MSGAD,RO      ;;SUB START OF MESSAGE
(1) 035554 006200                ASR   RO              ;;GET MESSAGE LNGTH IN WORDS
(1) 035556 010037 001212                MOV   RO,$MSGGLT     ;;PUT LENGTH IN MAILBOX
(1) 035562 012737 000004 001174        MOV   #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.
(1) 035570 000413                BR    5$
(1) 035572 017637 000004 035616 3$:  MOV   @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
(1) 035600 062766 000002 000004        ADD   #2,4(SP)       ;;BUMP RETURN ADDRESS
(3) 035606 013746 177776                MOV   177776 -(SP)   ;;PUSH 177776 ON STACK
(1) 035612 004737 034376                JSR   PC,$J          ;;CALL TYPE MACRO
(1) 035616 000000                4$:  .WORD 0
(1) 035620
(1) 035620 105737 035706                5$:
(1) 035624 001416                10$: TSTB  $FFLG        ;;SHOULD REPORT FATAL ERROR?
(1) 035626 005737 001214                BEQ   12$            ;;IF NOT: BR
(1) 035632 001413                TST  $ENV            ;;RUNNING UNDER APT?
(1) 035634 005737 001174                BEQ   12$            ;;IF NOT: BR
(1) 035640 001375                11$: TST  $MSGTYPE      ;;FINISHED LAST MESSAGE?
(1) 035642 017637 000004 001176        BNE   11$            ;;IF NOT: WAIT
(1) 035650 062766 000002 000004        MOV   @4(SP),$FATAL  ;;GET ERROR #
(1) 035656 005237 001174                ADD   #2,4(SP)       ;;BUMP RETURN ADDR.
(1) 035662 105037 035706                INC   $MSGTYPE       ;;TELL APT TO TAKE ERROR
(1) 035666 105037 035705                12$: CLRB  $FFLG        ;;CLEAR FATAL FLAG
(1) 035672 105037 035704                CLRB  $LFLG          ;;CLEAR LOG FLAG
(3) 035676 012601                CLRB  $MFLG          ;;CLEAR MESSAGE FLAG
(3) 035700 012600                MOV   (SP)+,R1       ;;POP STACK INTO R1
(1) 035702 000207                MOV   (SP)+,RO       ;;POP STACK INTO RO
(1) 035704 000                RTS   PC              ;;RETURN
(1) 035705 000                $MFLG: .BYTE 0        ;;MESSG. FLAG
(1) 035706 000                $LFLG: .BYTE 0        ;;LOG FLAG
(1) 035710 035710                $FFLG: .BYTE 0        ;;FATAL FLAG
(1) 000200                .EVEN
(1) 000001                APTSIZE=200
(1) 000100                APTENV=001
                APTSPOOL=100

```



```
(1) 000040 APTCSUP=040
2351 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1) *****
:POWER DOWN ROUTINE
(1) 035710 012737 036050 000024 $PWRDN: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 035716 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(3) 035724 010046 MCV R0,-(SP) ;;PUSH R0 ON STACK
(3) 035726 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 035730 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 035732 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 035734 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 035736 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 035740 017746 143174 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 035744 010637 036054 MOV SP,$SAVR6 ;;SAVE SP
(1) 035750 012737 035762 000024 MOV #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 035756 000000 HALT
(1) 035760 000776 BR -2 ;;HANG UP
(1)
(2)
(1) *****
:POWER UP ROUTINE
(1) 035762 012737 036050 000024 $PWRUP: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 035770 013706 036054 MOV $SAVR6,SP ;;GET SP
(1) 035774 005037 036054 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
(1) 036000 005237 036054 1$: INC $SAVR6 ;;WAIT FOR THE INC
(1) 036004 001375 BNE 1$ ;;OF WORD
(3) 036006 012677 143126 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
(3) 036012 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 036014 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
(3) 036016 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 036020 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 036022 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(5) 036024 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 036026 012737 035710 000024 MOV #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 036034 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(1) 036042 104401 TYPE ;;REPORT THE POWER FAILURE
(1) 036044 036056 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
(1) 036046 000002 RTI
(1) 036050 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
(1) 036052 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 036054 000000 $SAVR6: 0 ;;PUT THE SP HERE
(1) 036056 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
(1) 036064 000122 .EVEN
```



```

(1) 036300 000421
(1) 036302 042737 100000 036124 $KTNEX: BR $SIZEX
(1) 036310 012737 036340 000004 $SCORE: MOV #100000,$KT11 ;;KT11 NON-EXISTENT
(1) 036316 005002 CLR #SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
(1) 036320 062701 004000 1$: ADD R2 ;;SET UP BANK
(1) 036324 062702 000040 ADD #4000,R1 ;;INCREMENT BY 1K
(1) 036330 005711 ADD #40,R2 ;;1K STEP
(1) 036332 022701 177776 TST (R1) ;;TRAP ON TIME OUT
(1) 036336 001370 CMP #177776,R1 ;;LAST ONE
(1) 036340 162701 004000 BNE 1$ ;;NO--TRY AGAIN
(1) 036344 162702 000040 $SCROUT: SUB #4000,R1
(1) 036350 010006 $SIZEX: SUB #40,R2 ;;DROP BACK
(1) 036352 012637 000006 MOV RO,SP ;;RESTORE THE STACK
(1) 036356 012637 000004 MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
(1) 036362 010137 036404 MOV (SP)+,@#ERRVEC
(1) 036366 010237 036406 MOV R1,$LSTAD ;;LAST ADDRESS
(1) 036372 012603 MOV R2,$LSTBK ;;LAST BANK
(1) 036374 012602 MOV (SP)+,R3 ;;RESTORE R3
(1) 036376 012601 MOV (SP)+,R2 ;;RESTORE R2
(1) 036400 012600 MOV (SP)+,R1 ;;RESTORE R1
(1) 036402 000207 MOV (SP)+,R0 ;;RESTORE R0
(1) 036404 000000 RTS PC
(1) 036406 000000 $LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
2354 036410 000000 $LSTBK: .WORD 0 ;;CONTAINS THE LAST BANK
2355 036412 000776 KTERR: HALT ;;KT11 FAILURE
BR KTERR
  
```

2357

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

```
*****  
*SAVE R0-R5  
*CALL:  
* SAVREG  
*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:  
*  
*TOP---(+16)  
* +2---(+18)  
* +4---R5  
* +6---R4  
* +8---R3  
*+10---R2  
*+12---R1  
*+14---R0
```

(1) 036414
(3) 036414 010046
(3) 036416 010146
(3) 036420 010246
(3) 036422 010346
(3) 036424 010446
(3) 036426 010546
(1) 036430 016646 000022
(1) 036434 016646 000022
(1) 036440 016646 000022
(1) 036444 016646 000022
(1) 036450 000002

```
$SAVREG:  
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 R4,-(SP) ;;PUSH R4 ON STACK  
MOV R5,-(SP) ;;PUSH R5 ON STACK  
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW  
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW  
MOV 22(SP),-(SP) ;;SAVE PS OF CALL  
MOV 22(SP),-(SP) ;;SAVE PC OF CALL  
RTI
```

(1)
(1)
(1)
(1)
(1) 036452
(1) 036452 012666 000022
(1) 036456 012666 000022
(1) 036462 012666 000022
(1) 036466 012666 000022
(3) 036472 012605
(3) 036474 012604
(3) 036476 012603
(3) 036500 012602
(3) 036502 012601
(3) 036504 012600
(1) 036506 000002

```
*RESTORE R0-R5  
*CALL:  
* RESREG  
$RESREG:  
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW  
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW  
MOV (SP)+,R5 ;;POP STACK INTO R5  
MOV (SP)+,R4 ;;POP STACK INTO R4  
MOV (SP)+,R3 ;;POP STACK INTO R3  
MOV (SP)+,R2 ;;POP STACK INTO R2  
MOV (SP)+,R1 ;;POP STACK INTO R1  
MOV (SP)+,R0 ;;POP STACK INTO R0  
RTI
```

2359

.SBTTL INTEGER MULTIPLY ROUTINE

```

(1) (1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(3) 036510 010046
(3) 036512 010146
(3) 036514 010246
(1) 036516 005046
(1) 036520 016601 000012
(1) 036524 100002
(1) 036526 005216
(1) 036530 005401
(1) 036532 016602 000014
(1) 036536 100002
(1) 036540 005316
(1) 036542 005402
(1) 036544 012746 000021
(1) 036550 005000
(1) 036552 103001
(1) 036554 060200
(1) 036556 006000
(1) 036560 006001
(1) 036562 005316
(1) 036564 001372
(1) 036566 022616
(1) 036570 001403
(1) 036572 005400
(1) 036574 005401
(1) 036576 005600
(1) 036600 005726
(1) 036602 010066 000012
(1) 036606 010166 000010
(3) 036612 012602
(3) 036614 012601
(3) 036616 012600
(1) 036620 000207

```

```

*****
:CALL
:*
:*   MOV     MULTIPLIER,-(SP)
:*   MOV     MULTIPLICAND,-(SP)
:*   JSR     PC,@#$MULT
:*   RETURN  ;;PRODUCT IS ON THE STACK
:*
:*   STACK  PRODUCT
:*   -----
:*   TOP    LSB'S
:*   +2     MSB'S

```

```

$MULT:
MOV     R0,-(SP)          ;;PUSH R0 ON STACK
MOV     R1,-(SP)          ;;PUSH R1 ON STACK
MOV     R2,-(SP)          ;;PUSH R2 ON STACK
CLR     -(SP)             ;;CLEAR THE SIGN KEY
MOV     12(SP),R1         ;;GET THE MULTIPLICAND
BPL     1$                ;;BR IF PLUS
INC     (SP)              ;;SET THE SIGN KEY
NEG     R1                ;;MAKE THE MULTIPLICAND POSTIVE
1$:     MOV     14(SP),R2  ;;GET THE MULTIPLIER
BPL     2$                ;;BR IF PLUS
DEC     (SP)              ;;UPDATE THE SIGN KEY
NEG     R2                ;;MAKE THE MULTIPLIER POSTIVE
2$:     MOV     #17,-(SP)  ;;SET THE LOOP COUNT
CLR     R0                ;;SETUP FOR THE MULTIPLY LOOP
3$:     BCC     4$         ;;DON'T ADD IF MULTIPLICAND = 0
ADD     R2,R0
4$:     ROR     R0         ;;POSITION THE PARITIAL PRODUCT AND
ROR     R1                ;;THE MULTIPLICAND
DEC     (SP)              ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
BNE     3$                ;;BR IF NO
CMP     (SP)+,(SP)        ;;SHOULD PRODUCT BE NEGATIVE?
BEQ     5$                ;;GO TO EXIT IF NO
NEG     R0                ;;YES--SO MAKE IT SO
NEG     R1
SBC     R0
5$:     TST     (SP)+      ;;CLEAR SIGN INFO. OFF OF STACK
MOV     R0,12(SP)         ;;PUT THE PRODUCT ON THE STACK (MSB'S)
MOV     R1,10(SP)         ;;LSB'S
MOV     (SP)+,R2         ;;POP STACK INTO R2
MOV     (SP)+,R1         ;;POP STACK INTO R1
MOV     (SP)+,R0         ;;POP STACK INTO R0
RTS     PC

```


(1)	036742	010300			MOV	R3,R0	::REMAINDER AFTER THIS LOOP
(1)	036744	006101		6\$:	ROL	R1	::QUOTIENT BIT ENTERS HERE
(1)	036746	005316			DEC	(SP)	::DONE?
(1)	036750	001370			BNE	5\$::BR IF NO
(1)	036752	005701			TST	R1	::OVERFLOW?
(1)	036754	100005			BPL	8\$::BR IF NO
(1)	036756	052766	000002	000014	BIS	#2,14(SP)	::SET 'V' IN RETURN STATUS WORD
(1)	036764	005000			CLR	R0	::SET REMAINDER TO ALL ZEROS
(1)	036766	010001		7\$:	MOV	R0,R1	::COPY REMAINDER INTO QUOTIENT
(1)	036770	005726		8\$:	TST	(SP)+	::CLEAR COUNTER FROM STACK
(1)	036772	005716			TST	(SP)	::REMAINDER SIGN CORRECTION NEEDED?
(1)	036774	002004			BGE	9\$::BR IF NO
(1)	036776	005400			NEG	R0	::NEGATE REMAINDER
(1)	037000	105066	000001		CLRB	1(SP)	::CLEAR SIGN
(1)	037004	005316			DEC	(SP)	::BUT DON'T FORGET QUOTIENT
(1)	037006	005726		9\$:	TST	(SP)+	::QUOTIENT SIGN CORRECTION NEEDED?
(1)	037010	001401			BEQ	10\$::BR IF NO
(1)	037012	005401			NEG	R1	::NEGATE QUOTIENT
(1)	037014	010166	000020	10\$:	MOV	R1,20(SP)	::RETURN QUOTIENT AND
(1)	037020	010066	000016		MOV	R0,16(SP)	::REMAINDER TO USER
(3)	037024	012603			MOV	(SP)+,R3	::POP STACK INTO R3
(3)	037026	012602			MOV	(SP)+,R2	::POP STACK INTO R2
(3)	037030	012601			MOV	(SP)+,R1	::POP STACK INTO R1
(3)	037032	012600			MOV	(SP)+,R0	::POP STACK INTO R0
(1)	037034	012666	000002		MOV	(SP)+,2(SP)	::SETUP TO RETURN CONDITION CODES
(1)	037040	000002			RTI		::RETURN


```

(1) 037210 000000      0
(1) 037212 000012      12          ;;1.0E01
(1) 037214 000000      0
(1) 037216 000001      1          ;;1.0E00
(1) 037220 000000      0
(1) 037222 000014      $DECVL: .BLKB 12.          ;;RESERVE STORAGE FOR ASCII STRING
2364 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
(1)
(2) ;*****
(1) ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
(1) ;*UNSIGNED DECIMAL ASCII NUMBER.
(1) ;*CALL
(1) ;*   MOV     NUMBER,-(SP)      ;;PUT BINARY NUMBER ON THE STACK
(1) ;*   JSR     PC,@#$SB2D       ;;CALL
(1) ;*   RETURN                    ;;ADDRESS OF THE 1ST ASCII CHAR.IS ON THE STACK
(1)
(1) (1) 037236 016637 000002 037266 $SB2D: MOV     2(SP),1$      ;;SAVE BINARY NUMBER
(1) (1) 037244 012746 037266      MOV     #1$,-(SP)      ;;SET POINTER
(1) (1) 037250 004737 037042      JSR     PC,@#$SB2D     ;;CALL DOUBLE LENGTH CONVERT
(1) (1) 037254 062716 000005      ADD     #5,(SP)       ;;ONLY ALLOW FIVE CHARACTERS
(1) (1) 037260 012666 000002      MOV     (SP)+,2(SP)   ;;PICKUP POINTER
(1) (1) 037264 000207 000000      RTS     PC            ;;RETURN
(1) (1) 037266 000000 000000      1$:   .WORD 0,0

```

2366

.SBTTL TRAP DECODER

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)

*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) 037272 010046
(1) 037274 016600 000002
(1) 037300 005740
(1) 037302 111000
(1) 037304 006300
(1) 037306 016000 037326
(1) 037312 000200

\$TRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

(1)
(1)
(1)

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

(1) 037314 011646
(1) 037316 016666 000004 000002
(1) 037324 000002

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

(1)
(3)
(3)

.SBTTL TRAP TABLE

(3)
(3)
(3)
(3)
(3)

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

(3) 037326 037314
(3) 037330 034376
(3) 037332 034704
(3) 037334 034660
(3) 037336 034720
(3) 037340 033666
(1)
(3) 037342 033060
(1)
(3) 037344 033010
(3) 037346 033272
(3) 037350 033412
(3) 037352 033564
(3) 037354 036414
(3) 037356 036452

ROUTINE

\$TRPAD: .WORD \$TRAP2
\$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
\$GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
\$CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
\$RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
\$SAVREG ;;CALL=SAVREG TRAP+13(104413) SAVE R0-R5 ROUTINE
\$RESREG ;;CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE

2370

```

2372 .SBTTL A TO D FIELD SITE AND ADJUSTMENT LOOP
2373 037360 004737 045700 FSITE: JSR PC,CTRLCG ;TEST FOR CTRL C OR G
2374 ;CAMERA #0
2375 037364 032777 000100 141546 BIT #SW06,@SWR ;TEST IF CAM 0 G=1
2376 037372 001004 BNE 1$
2377 037374 004537 040106 JSR R5,CONVRT ;CONVERT
2378 037400 000000 0 ;CAMERA 0
2379 037402 040444 CAMOG1 ;STORE RESULTS
2380 037404 032777 000001 141526 1$: BIT #SW00,@SWR ;TEST IF CAM 0 G=2
2381 037412 001004 BNE 2$
2382 037414 004537 040106 JSR R5,CONVRT ;CONVERT
2383 037420 002000 BIT10 ;GAIN = 2
2384 037422 040446 CAMOG2
2385 ;CAMERA #1
2386 037424 032777 000200 141506 2$: BIT #SW7,@SWR ;TEST IF CAM 1 G=1
2387 037432 001004 BNE 3$
2388 037434 004537 040106 JSR R5,CONVRT ;CONVERT
2389 037440 000400 BIT8 ;CAMERA 1
2390 037442 040450 CAM1G1 ;STORE RESULTS
2391 037444 032777 000002 141466 3$: BIT #SW01,@SWR ;TEST IF CAM 1 G=2
2392 037452 001004 BNE 4$
2393 037454 004537 040106 JSR R5,CONVRT ;CONVERT
2394 037460 002400 BIT10!BIT8 ;CAMERA 1 GAIN = 2
2395 037462 040452 CAM1G2 ;RESULTS
2396 ;CAMERA #2
2397 037464 032777 000400 141446 4$: BIT #SW8,@SWR ;TEST IF CAM 2 G=1
2398 037472 001004 BNE 5$
2399 037474 004537 040106 JSR R5,CONVRT ;CONVERT
2400 037500 001000 BIT9 ;CAMERA #2
2401 037502 040454 CAM2G1 ;RESULTS
2402 037504 032777 000004 141426 5$: BIT #SW2,@SWR ;TEST IF CAM 2 G=2
2403 037512 001004 BNE 6$
2404 037514 004537 040106 JSR R5,CONVRT ;CONVERT
2405 037520 003000 BIT10!BIT9 ;GAIN = 2
2406 037522 040456 CAM2G2 ;RESULTS
2407 ;CAMERA #3
2408 037524 032777 001000 141406 6$: BIT #SW9,@SWR ;TEST IF CAM 3 G=1
2409 037532 001004 BNE 7$
2410 037534 004537 040106 JSR R5,CONVRT ;CONVERT
2411 037540 001400 BIT9!BIT8 ;CAMERA #3
2412 037542 040460 CAM3G1 ;RESULTS
2413 037544 032777 000010 141366 7$: BIT #SW3,@SWR ;TEST IF CAM 3 G=2
2414 037552 001004 BNE 10$
2415 037554 004537 040106 JSR R5,CONVRT ;CONVERT
2416 037560 003400 BIT10!BIT9!BIT8 ;GAIN = 2
2417 037562 040462 CAM3G2 ;RESULTS
2418 ;JOYSTICK
2419 037564 032777 000020 141346 10$: BIT #SW4,@SWR ;TEST IF JOYSTICK
2420 037572 001011 BNE CALRPT
2421 037574 052777 000001 142154 BIS #BIT0,@SFR ;ASK FOR JOYSTICK
2422 037602 105777 142150 11$: TSTB @SFR ;WAIT FOR JOY DONE
2423 037606 100375 BPL 11$
2424 037610 017737 142146 040464 MOV @JOY,JOYG1 ;SAVE THE RESULT

```

```
2426  
2427  
2428 037616 004537 040064  
2429 037622 000100  
2430 037624 000405  
2431 037626 104401 040466  
2432 037632 004537 040362  
2433 037636 040444  
2434 037640 004537 040064  
2435 037644 000001  
2436 037646 000405  
2437 037650 104401 040505  
2438 037654 004537 040362  
2439 037660 040446  
2440  
2441 037662 004537 040064  
2442 037666 000200  
2443 037670 000405  
2444 037672 104401 040524  
2445 037676 004537 040362  
2446 037702 040450  
2447 037704 004537 040064  
2448 037710 000002  
2449 037712 000405  
2450 037714 104401 040543  
2451 037720 004537 040362  
2452 037724 040452  
2453  
2454 037726 004537 040064  
2455 037732 000400  
2456 037734 000405  
2457 037736 104401 040562  
2458 037742 004537 040362  
2459 037746 040454  
2460 037750 004537 040064  
2461 037754 000004  
2462 037756 000405  
2463 037760 104401 040601  
2464 037764 004537 040362  
2465 037770 040456  
2466  
2467 037772 004537 040064  
2468 037776 001000  
2469 040000 000405  
2470 040002 104401 040620  
2471 040006 004537 040362  
2472 040012 040460  
2473 040014 004537 040064  
2474 040020 000010  
2475 040022 000405  
2476 040024 104401 040637  
2477 040030 004537 040362  
2478 040034 040462  
2479  
2480 040036 004537 040064  
2481 040042 000020
```

;NOW TEST IS TYPEOUT IS ENABLED

```
CALRPT: JSR R5,CKTSWR ;TEST BIT 6  
BIT6  
BR 1$ ;BR IF SET  
TYPE, CAMOTX ;REPORT CAMERA #0  
JSR R5,CAMUNP ;REPORT VALUES  
CAMOG1  
1$: JSR R5,CKTSWR ;TEST BIT 0  
BIT0  
BR 2$  
TYPE, CAMOTW ;REPORT G=2  
JSR R5,CAMUNP  
CAMOG2  
2$: JSR R5,CKTSWR ;TEST BIT 7  
BIT7  
BR 3$ ;BR IF SET  
TYPE, CAM1TX ;REPORT CAMERA #1  
JSR R5,CAMUNP ;REPORT VALUES  
CAM1G1  
3$: JSR R5,CKTSWR ;TEST BIT 1  
BIT1  
BR 4$  
TYPE, CAM1TW ;REPORT CAM 1 G=2  
JSR R5,CAMUNP  
CAM1G2  
4$: JSR R5,CKTSWR ;TEST BIT 8  
BIT8  
BR 5$ ;BR IF SET  
TYPE, CAM2TX ;REPORT CAMERA #2  
JSR R5,CAMUNP ;REPORT VALUES  
CAM2G1  
5$: JSR R5,CKTSWR ;TEST BIT 2  
BIT2  
BR 6$  
TYPE, CAM2TW ;REPORT CAMERA #2 G=2  
JSR R5,CAMUNP  
CAM2G2  
6$: JSR R5,CKTSWR ;TEST BIT 9  
BIT9  
BR 7$ ;BR IF SET  
TYPE, CAM3TX ;REPORT CAMERA #3  
JSR R5,CAMUNP ;REPORT VLAUES  
CAM3G1  
7$: JSR R5,CKTSWR ;TEST BIT 3  
BIT3  
BR 10$  
TYPE, CAM3TW ;REPORT CAMERA #3 G=2  
JSR R5,CAMUNP  
CAM3G2  
10$: JSR R5,CKTSWR ;TEST BIT 4  
BIT4
```

CZNCCC NCV11 DIAGNOSTIC
CZNCCC.P11 29-SEP-80 09:35

MACY11 30G(1063) 29-SEP-80 10:33 H 10 PAGE 82-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0124

2482	040044	000405		BR	11\$	
2483	040046	104401	040656	TYPE,	JOYTW	
2484	040052	004537	040362	JSR	R5,CAMUNP	;REPORT JOYSTICK
2485	040056	040464		JOYG1		
2486	040060	000137	037360	11\$: JMP	FSITE	
2487						
2488	040064	032577	141050	CKTSWR: BIT	(R5)+,@SWR	;TEST IF INHIBIT THIS CAMERA
2489	040070	001005		BNE	1\$;BR IF YES
2490	040072	032777	020000 141040	BIT	#SW13,@SWR	;TEST INHIBIT TYPE-OUT
2491	040100	001001		BNE	1\$;BR IF YES
2492	040102	005725		TST	(R5)+	;BUMP EXIT POINTER
2493	040104	000205		1\$: RTS	R5	;EXIT

```

2495 ;SUBROUTINE TO TAKE 8 CONVERSIONS AND AVERAGE FOUR OF THEM
2496 040106 012537 040356 CONVRT: MOV (R5)+,11$ ;SAVE CAMERA CHANNEL
2497 040112 012777 004000 141636 4$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
2498 040120 013777 040356 141620 MOV 11$,@CSR ;SELECT CAMERA AND GAIN
2499 040126 012777 060000 141620 MOV #BUFO,@BAR ;LOAD BUS ADDRESS
2500 040134 012777 177770 141610 MOV #-10,@WCR ;LOAD WORD COUNT
2501 040142 013777 040360 141606 MOV INOUTZ,@SFR ;SET Z INPUTS IF ENABLED
2502 040150 012700 000004 MOV #4.,R0 ;LOAD COUNTER
2503 040154 005001 CLR R1 ;FOR OUT OF RANGE VOLTAGE
2504 040156 052777 000001 141562 BIS #BIT0,@CSR ;ENABLE THE DEVICE
2505 040164 105777 141556 1$: TSTB @CSR ;WAIT FOR IDLE
2506 040170 100432 BMI 3$
2507 040172 005301 DEC R1 ;DELAY
2508 040174 001373 BNE 1$
2509 040176 004737 045700 JSR PC,CTRLCG ;TEST FOR CTRL G OR C
2510 040202 005300 DEC R0 ;FINISHED OUT OF RANGE TIMER
2511 040204 001367 BNE 1$
2512 040206 032777 020000 140724 BIT #SW13,@SWR ;TEST IF INHIBIT TYPEOUT
2513 040214 001012 BNE 5$ ;BR IF INHIBIT
2514 040216 113700 040357 MOVB 11$,+1,R0 ;GET CAMERA #
2515 040222 042700 177774 BIC #177774,R0 ;MASK OFF EXTRA BITS
2516 040226 062700 000060 ADD #60,R0 ;MAKE INTO ASCII
2517 040232 110037 032720 MOVB R0,OUTCHN ;INSERT INTO ASCII ERROR MESSAGE
2518 040236 104401 032625 TYPE, OUTRNG ;TELL OPERATOR
2519 040242 032777 100000 140670 5$: BIT #SW15,@SWR ;TEST IF HALT ON NO CONVERSION ERROR
2520 040250 001401 BEQ 6$ ;BR IF CLEARED
2521 040252 000000 HALT ;NO CONVERT FLAG - INPUT VOLTAGE WAS OUT OF RANGE OR NO "Z" PULSES
2522 040254 000716 6$: BR 4$
2523 040256 012700 060000 3$: MOV #BUFO,R0 ;LOAD POINTER
2524 040262 005003 CLR R3 ;CLEAR RESULT
2525 040264 005004 CLR R4
2526 040266 012737 000004 040354 MOV #4,10$ ;LOAD COUNTER
2527 040274 112002 2$: MOVB (R0)+,R2 ;GET LOW BYTE
2528 040276 112001 MOVB (R0)+,R1 ;GET HIGH BYTE
2529 040300 042702 177400 BIC #177400,R2
2530 040304 042701 177400 BIC #177400,R1
2531 040310 060104 ADD R1,R4 ;UPDATE RESULT
2532 040312 060203 ADD R2,R3 ;UPDATE RESULT
2533 040314 005337 040354 DEC 10$ ;FINISHED ?
2534 040320 001365 BNE 2$
2535 040322 006203 ASR R3
2536 040324 006203 ASR R3
2537 040326 006204 ASR R4
2538 040330 006204 ASR R4
2539 040332 010437 040354 MOV R4,10$ ;SAVE IT
2540 040336 105037 040355 CLR B 10$,+1 ;CLEAR HIGH BYTE
2541 040342 110337 040355 MOV B R3,10$,+1 ;LOAD HIGH BYTE
2542 040346 013735 040354 MOV 10$,@(R5)+ ;SAVE THE RESULT
2543 040352 000205 RTS R5
2544 040354 000000 10$: 0
2545 040356 000000 11$: 0
2546 040360 000002 INOUTZ: TESTZ ;0= USE CUSTOMER Z PULSES, 1= MAINT Z PULSES
2547
2548 ;SUBROUTINE TO TYPE THE CAMERA MESSAGE
2549 040362 013537 040440 CAMUNP: MOV @(R5)+,10$ ;GET RESULT
2550 040366 113737 040441 040442 MOV B 10$,+1,11$

```

2551	040374	105037	040441		CLRB	10\$+1	
2552	040400	105037	040443		CLRB	11\$+1	
2553	040404	013746	040440		MOV	10\$,-(SP)	
2554	040410	104403			TYPOS		
2555	040412	003	001		.BYTE	3,1	;TYPE 3 DIGIT NUMBER
2556	040414	104401	040674		TYPE,	G1B	;AND TRALING ASCII
2557	040420	013746	040442		MOV	11\$,-(SP)	
2558	040424	104403			TYPOS		
2559	040426	003	001		.BYTE	3,1	;TYPE 3 DIGIT NUMBER
2560	040430	000240			NOP		
2561	040432	000240			NOP		
2562	040434	000240			NOP		
2563	040436	000205			RTS	R5	;EXIT
2564	040440	000000			10\$:	0	
2565	040442	000000			11\$:	0	
2566	040444	000000			CAM0G1:	0	
2567	040446	000000			CAM0G2:	0	
2568	040450	000000			CAM1G1:	0	
2569	040452	000000			CAM1G2:	0	
2570	040454	000000			CAM2G1:	0	
2571	040456	000000			CAM2G2:	0	
2572	040460	000000			CAM3G1:	0	
2573	040462	000000			CAM3G2:	0	
2574	040464	000000			JOYG1:	0	
2575							
2576	040466	015	012		CAM0TX:	.BYTE 15,12	
2577	040470	040503	030115	020060	.ASCIZ	/CAM00 G=1 X=/	
	040476	036507	020061	036530			
	040504	000					
2578	040505	015	012		CAM0TW:	.BYTE 15,12	
2579	040507	103	046501	030060	.ASCIZ	/CAM00 G=2 X=/	
	040514	043440	031075	054040			
	040522	000075					
2580	040524	015	012		CAM1TX:	.BYTE 15,12	
2581	040526	040503	030115	020061	.ASCIZ	/CAM01 G=1 X=/	
	040534	036507	020061	036530			
	040542	000					
2582	040543	015	012		CAM1TW:	.BYTE 15,12	
2583	040545	103	046501	030460	.ASCIZ	/CAM01 G=2 X=/	
	040552	043440	031075	054040			
	040560	000075					
2584	040562	015	012		CAM2TX:	.BYTE 15,12	
2585	040564	040503	030115	020062	.ASCIZ	/CAM02 G=1 X=/	
	040572	036507	020061	036530			
	040600	000					
2586	040601	015	012		CAM2TW:	.BYTE 15,12	
2587	040603	103	046501	031060	.ASCIZ	/CAM02 G=2 X=/	
	040610	043440	031075	054040			
	040616	000075					
2588	040620	015	012		CAM3TX:	.BYTE 15,12	
2589	040622	040503	030115	020063	.ASCIZ	/CAM03 G=1 X=/	
	040630	036507	020061	036530			
	040636	000					
2590	040637	015	012		CAM3TW:	.BYTE 15,12	
2591	040641	103	046501	031460	.ASCIZ	/CAM03 G=2 X=/	
	040646	043440	031075	054040			

	040654	000075				
2592	040656				JOYTX:	
2593	040656	015	012		JOYTW:	.BYTE 15,12
2594	040660	047512	051531	044524		.ASCIZ /JOYSTICK X=/
	040666	045503	054040	000075		
2595	040674	054440	000075		G1B:	.ASCIZ / Y=/
2596	040700	005015	044504	043106	GAIN:	.ASCII <15><12>/DIFFERENTIAL LINEARITY:/
	040706	051105	047105	044524		
	040714	046101	046040	047111		
	040722	040505	044522	054524		
	040730	072				
2597	040731	040	020040	040507	GAINX:	.ASCII / GAIN = /
	040736	047111	036440	040		
2598	040743	061	040	040	GAIN1:	.BYTE 61,40,40,0
	040746	000				
2599	040747	040	026455	000040	DASH:	.ASCIZ / -- /
2600	040754	046040	041123	005015	LSBMSG:	.ASCIZ / LSB/<15><12>
	040762	000				
2601	040763	040	045523	050111	SKPMSG:	.ASCIZ / SKIPPED STATE(S)/<15><12>
	040770	042520	020104	052123		
	040776	052101	024105	024523		
	041004	005015	000			
2602	041007	040	025052	051105	ERMSG:	.ASCIZ / **ERROR**/<15><12>
	041014	047522	025122	006452		
	041022	000012				
2603	041024	020040	020040	047440	OKMSG:	.ASCIZ / OK/<15><12>
	041032	006513	000012			
2604	041036	047040	051101	047522	NARMSG:	.ASCIZ # NARROW STATE(S)#<15><12>
	041044	020127	052123	052101		
	041052	024105	024523	005015		
	041060	000				
2605	041061	040	044527	042504	WIDMSG:	.ASCIZ # WIDE STATE(S)#<15><12>
	041066	051440	040524	042524		
	041074	051450	006451	000012		
2606	041102	052123	052101	026505	STATE:	.ASCIZ /STATE-- WIDTH/<15><12>
	041110	020055	044527	052104		
	041116	006510	000012			
2607	041122	046040	041123	046440	LINEA:	.ASCIZ / LSB MAXIMUS AT /
	041130	054101	046511	051525		
	041136	040440	020124	000		
2608	041143	057	000		SLASH:	.ASCIZ #/#
2609	041145	122	046105	052101	MSG21:	.ASCIZ /RELATIVE ACCURACY:/<15><12>
	041152	053111	020105	041501		
	041160	052503	040522	054503		
	041166	006472	000012			
2610	041172	005015	047522	020115	COMP:	.ASCIZ <15><12>/ROM BLASTING COMPLETED/
	041200	046102	051501	044524		
	041206	03516	041440	046517		
	041214	035120	052105	042105		
	041222	000				
2611	041223	015	052012	046511	TIMEO:	.ASCII <15><12>/TIMEOUT FROM BLASTER CHECK SWITCHES ON BLASTER/
	041230	047505	052125	043040		
	041236	047522	020115	046102		
	041244	051501	042524	020122		
	041252	020040	044103	041505		
	041260	020113	053523	052111		

2612	041266	044103	051505	047440	
	041274	020116	046102	051501	
	041302	042524	122		
	041305	015	020012	051117	.ASCIZ <15><12>/ GR REPLACE ROM IN BLASTER/<15><12>
	041312	051040	050105	040514	
	041320	042503	051040	046517	
	041326	044440	020116	046102	
	041334	051501	042524	006522	
	041342	000012			
2613	041344	015	012		FIELDI: .BYTE 15,12
2614	041346	047111	042524	047122	.ASCIZ /INTERNAL MAINT. Z PULSES <Y FOR YES> ? /
	041354	046101	046440	044501	
	041362	052116	020056	020132	
	041370	052520	051514	051505	
	041376	036040	020131	047506	
	041404	020122	042531	037123	
	041412	037440	000040		
2615	041416	005015	053101	051105	AVRGO: .ASCIZ <15><12>/AVERAGE OF 128 STATES = /
	041424	043501	020105	043117	
	041432	030440	034062	051440	
	041440	040524	042524	020123	
	041446	020075	000		
2616	041451	015	012	007	UNFIX: .BYTE 15,12,7
2617	041454	054105	042503	042105	.ASCIZ /EXCEEDED CORRECTION COUNTER FOR /
	041462	042105	041440	051117	
	041470	042522	052103	047511	
	041476	020116	047503	047125	
	041504	042524	020122	047506	
	041512	020122	000		
2618	041515	015	012	007	ABORTB: .BYTE 15,12,7
2619	041520	047522	020115	046102	.ASCII /ROM BLASTING ABORTED/
	041526	051501	044524	043516	
	041534	040440	047502	052122	
	041542	042105			
2620	041544	015	012	007	.BYTE 15,12,7,0
	041547	000			
2621	041550	020040	026455	020040	SGNVAL: .ASCII / -- /
2622	041556	053	040		SGNVL1: .BYTE 53,40
2623	041560	060	060	056	PERTXT: .BYTE 60,60,56,60,45,15,12,0
	041563	060	045	015	
	041566	012	000		
2624	041570	015	012	000	RTN: .BYTE 15,12,0,0
	041573	000			
2625	041574	041040	046105	053517	BELMSG: .ASCIZ / BELOW LIMIT/<15><12>
	041602	046040	046511	052111	
	041610	005015	000		
2626	041613	040	047516	046522	NORMSG: .ASCIZ / NORMAL STATES/<15><12>
	041620	046101	051440	040524	
	041626	042524	006523	000012	
2627	041634	040440	047502	042526	ABOMSG: .ASCIZ / ABOVE LIMIT/<15><12>
	041642	046040	046511	052111	
	041650	005015	000		
2628	041653	040	040520	051523	PASMSG: .ASCIZ / PASSED/<15><12>
	041660	042105	005015	000	
2629	041665	040	040506	046111	FAIMSG: .ASCIZ / FAILED/<15><12>
	041672	042105	005015	000	

2630		041700		
2631	041700	000000	LSBAVG: 0	.EVEN
2632	041702	000000	LSBSVQ: 0	
2633	041704	000000	LSBSVR: 0	
2634	041706	000000	LSBSVW: 0	
2635	041710	000000	DIFEX1: 0	
2636	041712	042134	ROMPNT: ROMVAL	
2637	041714	027340	NOMIAL: 12000.	; AVERAGE COUNT
2638				
2639			: THESE LOC.'S CLEARED BY DIFLIN	
2640	041716	000000	AVGVAL: 0	
2641	041720	000000	DIFERR: 0	
2642	041722	000000	BELOW: 0	
2643	041724	000000	ABOVE: 0	
2644	041726	000000	NORMAL: 0	
2645	041730	000000	NARROW: 0	
2646	041732	000000	NARA: 0	
2647	041734	000000	NARB: 0	
2648	041736	000000	WIDA: 0	
2649	041740	000000	WIDB: 0	
2650	041742	000000	DIF: 0	
2651	041744	000000	OUT: 0	
2652	041746	000000	FIRST: 0	

```

2654
2655 ;KEYBOARD CONVERSATION TO THE BLASTER SELECT TEST "T"
2656 ;APPLY POWER TO BLASTER AND DEPRESS I/O AND EXECUTE BUTTONS
2657 ;ALL INDICATORS SHOULD LIGHT THE BLASTER SHOULD SEND A ">" CHARACTER
2658 ;TYPE THE COMMANDS FOLLOWED BY A "CR", SOME WILL ECHO THE CR AND OTHERS
2659 ;WILL NOT TYPE "ESC"/"AL" KEYS TO ABORT CURRNET COMMAND
2660 ;TYPE KF2/2 ;THIS SETS OCTAL COMMAND MODE
2661 ;TYPE FM30 ;THIS SET OCTAL INPUT/OUTPUT
2662 ;TYPE SP[0037/8] ;THIS SIZES THE PROM -- 0037/8 IS REPORTED BY THE BLASTER
2663 ;TYPE LD ;THIS READS THE PROM INTO THE RAM
2664 ;TYPE D00/37 ;THIS OUTPUTS THE RAM CONTENTS TO THE TTY
2665 ;TYPE ZP ;THIS EXITS COMPUTER CONTROL TO THE BUTTON MODE
2666
2667 041750 105777 137722 LOOPC: TSTB @DLICSR ;BLASTER INPUT ?
2668 041754 100007 BPL LOOPD ;NO
2669 041756 000240 NOP
2670 041760 000240 NOP
2671 041762 000240 NOP
2672 041764 017720 137710 MOV @DLIBD,(R0)+ ;SAVE THE CHARACTER
2673 041770 005237 042132 INC BLICNT ;UPDATE COUNTER
2674
2675 041774 105777 137150 LOOPD: TSTB @STPS ;PRINTER READY ?
2676 042000 100363 BPL LOOPC ;BR IF NOT
2677 042002 005737 042132 TST BLICNT ;ANY CHARACTERS
2678 042006 001413 BEQ LOOPE ;BR IF NOT
2679 042010 012177 137136 MOV (R1)+,@STPB ;PRINT THE CHAR.
2680 042014 005337 042132 DEC BLICNT
2681 042020 001353 BNE LOOPC ;BR IF MORE DATA
2682 042022 012700 060000 BTALK: MOV #BUFO,R0
2683 042026 010001 MOV R0,R1
2684 042030 005037 042132 CLR BLICNT
2685 042034 000745 BR LOOPC
2686 ;COME HERE IS NO BLASTER DATA TO BE TYPED
2687 042036 105777 137102 LOOPE: TSTB @STKS ;KEYBOARD INPUT
2688 042042 100342 BPL LOOPC ;BR IF NOT
2689 042044 017702 137076 MOV @STKB,R2 ;READ CHAR
2690 042050 042702 177600 BIC #177600,R2 ;MASK THE DATA
2691 042054 022702 000003 CMP #3,R2 ;TEST FOR CTRL C
2692 042060 001413 BEQ 2$ ;BR IF CTRL C
2693 042062 022702 000177 CMP #177,R2 ;TEST FOR RUBOUT
2694 042066 001412 BEQ LOOPF ;BR IF ESC
2695 042070 020227 000140 CMP R2,#140 ;TEST IF LOWER CASE
2696 042074 103402 BLO 1$ ;BR IF NOT
2697 042076 042702 000040 BIC #40,R2 ;MAKE UPPER CASE
2698 042102 010277 137576 1$: MOV R2,@DLODB ;LOAD BLASTER DATA OUT
2699 042106 000720 BR LOOPC
2700 042110 000137 002646 2$: JMP RBEGO ;JUMP TO RESTART
2701 042114 012777 000033 137562 LOOPF: MOV #33,@DLODB ;LOAD ESC <CANCLE>
2702 042122 012777 000134 137022 MOV #' \,@STPB
2703 042130 000707 BR LOOPC
2704 042132 000000 BLICNT: 0

```

```

2706
2707      ;A017 CORRECTION PROM VALUES    <MEMORY RAM>
2708
2709 042134      200      200      200 ROMVAL: .BYTE  200,200,200,200,200,200,200,200 ;GAIN = 1 VALUE
      042137      200      200      200
      042142      200      200
2710 042144      200      200      200      .BYTE  200,200,200,200,200,200,200,200
      042147      200      200      200
      042152      200      200
2711 042154      200      200      200      .BYTE  200,200,200,200,200,200,200,200 ;GAIN = 2 VALUE
      042157      200      200      200
      042162      200      200
2712 042164      200      200      200      .BYTE  200,200,200,200,200,200,200,200
      042167      200      200      200
      042172      200      200
2713
2714      ;A017 CORRECTION PRAM VALUES    <RAM MEMORY>
2715      ;ROMVAL MUST BE IN COMPLEMENTED AND REVERSED ORDER <LSB=MSB>
2716
2717 042174      376      376      376 RAMVAL: .BYTE  376,376,376,376,376,376,376,376 ;GAIN = 1 VALUE
      042177      376      376      376
      042202      376      376
2718 042204      376      376      376      .BYTE  376,376,376,376,376,376,376,376
      042207      376      376      376
      042212      376      376
2719 042214      376      376      376      .BYTE  376,376,376,376,376,376,376,376 ;GAIN = 2 VALUE
      042217      376      376      376
      042222      376      376
2720 042224      376      376      376      .BYTE  376,376,376,376,376,376,376,376
      042227      376      376      376
      042232      376      376
2721
2722      ;M8036 PROGRAM PROM VALUES
2723
2724
2725 042234      215      244      116 PROROM: .BYTE  215,244,116,144,215,244,116,144 ;M8036 PROGRAM PROM
      042237      144      215      244
      042242      116      144
2726 042244      215      244      116      .BYTE  215,244,116,144,215,244,116,144
      042247      144      215      244
      042252      116      144
2727 042254      106      144      314      .BYTE  106,144,314,304,377,377,377,377
      042257      304      377      377
      042262      377      377
2728 042264      377      377      377      .BYTE  377,377,377,377,377,377,377,377
      042267      377      377      377
      042272      377      377

```

```

2730      ;BLASTER SECTION
2731 042274 104401 050764      BLAST: TYPE, PRIMO      ;TELL OPERATOR ABOUT THE CABLE
2732 042300 104401 051376      TYPE, PRIM5      ;AND SWITCHES
2733 042304 042777 000001 137364 BIC #BIT0,@DLICSR ;ENSURE INIT. ROM CONTROL LINE
2734 042312 004537 043064      JSR R5,RDBLST    ;TELL THE BLASTER TO INITILIZE
2735 042316 052562      ESCP      ;REALLY WE ARE WAITING FOR THE OPERATOR
2736 042320 004537 043064      JSR R5,RDBLST    ;TELL THE BLASTER THE KEYBOARD INPUT MODE
2737 042324 052564      KFO
2738 042326 004537 043064      JSR R5,RDBLST    ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
2739 042332 052573      FMO
2740      ;PRIME THE MEMORY RAM BUFFER
2741 042334 012700 042134      MOV #ROMVAL,R0    ;GET POINTER
2742 042340 112720 000200 1$: MOVB #200,(R0)+ ;LOAD THE BUFFER
2743 042344 022700 042174      CMP #ROMVAL+40,R0 ;TEST IF DONE
2744 042350 001373      BNE 1$           ;BR IF BUFFER NOT PRIMED
2745 042352 012737 000036 046004 MOV #36,PRIME1    ;LOAD GAIN AND RESOL. VALUES
2746 042360 004537 046010      JSR R5,LIMITS    ;LOAD DIFLIN ERROR LIMIT VALUES
2747 042364 046074      G1LIMO          ;G1 LIMIT - USERS
2748 042366 046114      G1LIM1         ;G1 LIMIT - OPTION CHECKOUT
2749 042370 112737 000061 040743 MOVB #'1,GAIN1    ;LOAD GAIN TYPEOUT VALUE
2750 042376 012737 042135 041712 MOV #ROMVAL+1,ROMPNT ;LOAD RAM POINTER
2751 042404 004737 042450      JSR PC,4$       ;TEST THAT GAIN
2752 042410 012737 002036 046004 MOV #2036,PRIME1 ;LOAD GAIN AND RESOL. VALUES
2753 042416 004537 046010      JSR R5,LIMITS    ;LOAD DIFLIN ERROR LIMIT VALUES
2754 042422 046134      G2LIMO          ;G2 LIMIT - USERS
2755 042424 046154      G2LIM1         ;G2 LIMIT - OPTION CHECKOUT
2756 042426 112737 000062 040743 MOVB #'2,GAIN1    ;LOAD GAIN TYPEOUT VALUE
2757 042434 012737 042155 041712 MOV #ROMVAL+21,ROMPNT ;LOAD RAM POINTER
2758 042442 004737 042450      JSR PC,4$       ;TEST GAIN OF 2
2759 042446 000436      BR 10$
2760 042450 012737 000003 042620 4$: MOV #3,100$      ;LOAD LOOP COUNTER
2761 042456 012737 000001 041710 5$: MOV #1,DIFEX1    ;LOAD DIFLIN EXIT FLAG
2762 042464 012737 042502 046006 MOV #6$,DIFEXO   ;LOAD DIFLIN EXIT RETURN
2763 042472 004737 042622      JSR PC,ZAPRAM    ;ENSURE BLASTER RAM HAS BEEN LOADED WITH
2764      ;THE 'RAMVAL' VALUE
2765 042476 000137 043722      JMP DIFLNO       ;RUN 'DIFLIN'
2766 042502 004737 043362 6$: JSR PC,ADJFIX   ;ADJUST AND FIX THE CORRECTION WEIGHTS
2767 042506 005337 042620      DEC 100$        ;HAS THIS SECTION BEEN EXECITED N TIMES
2768 042512 001361      BNE 5$          ;BR IF NOT
2769 042514 005737 041720      TST DIFERR      ;TEST IF ERROR OCCURRED?
2770 042520 001410      BEQ 8$          ;BR IF NOT
2771 042522 104401 041451 7$: TYPE, UNFIX     ;TELL THE OPERATOR
2772 042526 104401 040731      TYPE, GAINX     ;ABOUT IT
2773 042532 104401 041515      TYPE, ABORTB    ;TELL OPER. THE BLASTING IS ABORTED
2774 042536 000137 002646      JMP RBEGO       ;RESTART
2775 042542 000207 8$: RTS PC ;EXIT

```

```

2777      ;THE RAM DATA IS NOW CORRECT - BLAST THE BLANK ROM
2778 042544 004537 043064 10$: JSR R5,RDBLST ;TELL THE BLASTER TO COMP. THE DATA
2779 042550 053013 CMO
2780 042552 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER TO CHECK-SUM DATA
2781 042556 053023 CCO
2782 042560 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER TO BLAST THE ROM
2783 042564 053026 PGO
2784 042566 005237 043244 INC NOEXIT ;SET FLAG TO EXIT WITHOUT RECPT OF '>' CHAR FROM BLASTER
2785 042572 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
2786 042576 053036 ZPO
2787      ;INFORM THE OPERATOR ABOUT THE BLASTER ROM
2788 042600 104401 051767 TYPE, PRIM1 ;TELL OPERATOR TO REMOVE RAM CABLE AND INSTERT ROM INTO A017
2789 042604 104411 RDLIN
2790 042606 012600 MOV (SP)+,R0 ;CLEAN STACK
2791 042610 104401 041172 TYPE, COMP ;COMPLETED
2792 042614 000137 002646 JMP RBEGO
2793 042620 000000
2794      100$: 0
;SUBROUTINE TO CONVERT THE MEMORY RAM VALUE INTO RAM MEMORY VALUE
2795 042622 012703 042174 ZAPRAM: MOV #RAMVAL,R3 ;LOAD OUTPUT POINTER
2796 042626 012700 042134 MOV #ROMVAL,R0 ;LOAD INPUT POINTER
2797 042632 112001 1$: MOV (R0)+,R1 ;GET A BYTE
2798 042634 105101 COMB R1 ;INVERT VALUE
2799 042636 005002 CLR R2 ;CLEAR RESULT
2803 042640 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042642 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042644 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042646 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042650 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042652 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042654 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042656 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042660 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042662 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042664 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042666 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042670 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042672 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042674 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042676 006102 ROL R2 ;MOVE CARRY INTO BIT 0
2804 042700 110223 MOV R2,(R3)+ ;SAVE OUTPUT BYTE
2805 042702 020027 042174 CMP R0,#ROMVAL+40 ;TEST IF END
2806 042706 001351 BNE 1$ ;BR IF NOT
2807      ;SUBROUTINE TO LOAD MEMORY RAM INTO RAM MEMORY
2808 042710 004537 043246 JSR R5,CONROM ;CONVERT 'ROMVAL' TABLE INTO BLASTER FORMAT
2809 042714 042174 RAMVAL
2810 042716 052777 000001 136752 BIS #BIT0,@DLICSR ;ENABLE RAM MEMORY TO BE LOADED
2811 042724 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER THE NEW RAM DATA
2812 042730 052601 DIO
2813 042732 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER TO CHECKSUM THE DATA
2814 042736 053023 CCO
2815 042740 004537 043064 JSR R5,RDBLST ;TELL THE BLASTER TO PROGRAM THE RAM-PAK
2816 042744 053026 PGO
2817 042746 042777 000001 136722 BIC #BIT0,@DLICSR ;RE-ENTER EMMULATE MODE
2818 042754 000207 RTS ;EXIT

```

2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842

042756 104401 051540
042762 042777 000001
042770 004537 043064
042774 052562
042776 004537 043064
043002 052564
043004 004537 043064
043010 052573
043012 004537 043246
043016 042234
043020 004537 043064
043024 052601
043026 004537 043064
043032 053023
043034 004537 043064
043040 053026
043042 005237 043244
043046 004537 043064
043052 053036
043054 104401 041172
043060 000137 002646

136706

;BLASTING THE PROGRAM PROM (M8036)

PBLAST: TYPE, PRIM3 ;INFORM THE OPERATOR
BIC #BIT0,@DLICSR ;ENSURE ROM MODE
JSR R5,RDBLST ;TELL THE BLASTER TO INITILIZE
ESCP
JSR R5,RDBLST ;TELL THE BLASTER THE KEYBOARD INPUT MODE
KFO
JSR R5,RDBLST ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
FMO
JSR R5,CONROM ;CONVERT THE "PROROM" TABLE INTO BLASTER FORMAT
PROROM
JSR R5,RDBLST ;TELL THE BLASTER THE PROM DATA
DIO
JSR R5,RDBLST ;TELL THE BLASTER TO CHECKSUM THE DATA
CCO
JSR R5,RDBLST ;TELL THE BLASTER TO BLAST THE PROM
PGO
INC NOEXIT ;SET FLAG TO EXIT WITHOUT RECPT OF ">" FROM BLASTER
JSR R5,RDBLST ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
ZPO
TYPE, COMP ;INFORM THE OPERATOR IT'S DONE
JMP RBEGO

```

2844
2845 ;COME THERE TO OUTPUT A MESSAGE TO THE BLASTER
2846 ;EXIT ONLY WHEN THE BLASTER HAS ECHOED AN '>'
2847 ;REPORT AN ERROR UPON TIMEOUT UNLESS 'NOEXIT' IS NON-ZERO
2848 043064 010046 RDBLST: MOV R0, -(SP) ;SAVE R0
2849 043066 012500 MOV (R5)+, R0 ;GET ARG.
2850 043070 012737 000012 043242 MOV #10, 13$ ;LOAD DELAY FOR ERROR COUNTER
2851 043076 111037 043234 MOV (R0), 10$ ;LOAD A CHAR
2852 043102 017737 136572 043236 MOV @DLIBD, 11$ ;FALSE READ TO CLEAR READY
2853 043110 105777 136566 1$: TSTB @DLOCSR ;WAIT FOR OUTPUT READY
2854 043114 100375 BPL 1$
2855 043116 105737 043234 TSTB 10$ ;TEST IF FINISHED ?
2856 043122 001406 BEQ 2$
2857 043124 112037 043234 MOV (R0)+, 10$ ;GET A CHAR.
2858 043130 001403 BEQ 2$ ;BR IF TERM
2859 043132 113777 043234 136544 MOV 10$, @DLODB ;OUTPUT THE CHARACTER
2860 043140 105777 136532 2$: TSTB @DLICSR ;WAIT FOR INPUT
2861 043144 100415 BMI 4$ ;BR IF INPUT
2862 043146 005337 043240 DEC 12$ ;DELAY
2863 043152 001372 BNE 2$
2864 043154 005337 043242 DEC 13$ ;DELAY
2865 043160 001367 BNE 2$
2866 043162 005737 043244 TST NOEXIT ;TEST IF EXIT IS ALLOWED WITHOUT '>'
2867 043166 001016 BNE 5$ ;BR IF YES
2868 043170 104401 041223 TYPE, TIMEO
2869 043174 000137 002646 JMP RBEGC
2870 043200 017737 136474 043236 4$: MOV @DLIBD, 11$ ;READ CHAR
2871 043206 042737 177600 043236 BIC #177600, 11$ ;MASK OFF BITS
2872 043214 022737 000076 043236 CMP #'>, 11$ ;TEST FOR >
2873 043222 001332 BNE 1$ ;TRY NEXT CHAR.
2874 043224 012600 5$: MOV (SP)+, R0 ;RESTORE R0
2875 043226 005037 043244 CLR NOEXIT
2876 043232 000205 RTS R5 ;EXIT
2877
2878 043234 000000 10$: 0
2879 043236 000000 11$: 0
2880 043240 000000 12$: 0
2881 043242 000000 13$: 0
2882 043244 000000 NOEXIT: 0
  
```



```
2884  
2885  
2886 043246 012700 052614  
2887 043252 012501  
2888 043254 010137 043324  
2889 043260 062737 000040 043324  
2890 043266 111102  
2891 043270 004737 043346  
2892 043274 111102  
2893 043276 004737 043340  
2894 043302 112102  
2895 043304 004737 043326  
2896 043310 062700 000007  
2897 043314 020137 043324  
2898 043320 001362  
2899 043322 000205  
2900 043324 000000  
2901  
2902  
2903  
2904 043326 042702 177477  
2905 043332 006202  
2906 043334 006202  
2907 043336 006202  
2908 043340 006202  
2909 043342 006202  
2910 043344 006202  
2911 043346 042702 177770  
2912 043352 062702 000060  
2913 043356 110240  
2914 043360 000207  
2915
```

```
;  
CONROM: MOV #DIDATA+3,R0 ;LOAD RESULT POINTER  
MOV (R5)+,R1 ;LOAD INITIAL VALUE POINTER  
MOV R1,10$ ;COPY THE START  
ADD #40,10$ ;MAKE THE END ADDRESS  
1$: MOVB (R1),R2 ;GET A BYTE OF DATA  
JSR PC,SHUF0 ;CONVERT IT TO ASCII  
MOVB (R1),R2 ;GET SAME BYTE AGAIN  
JSR PC,SHUF3 ;CONVERT 2ND DIGIT  
MOVB (R1)+,R2 ;GET MSD  
JSR PC,SHUF6 ;CONVERT IT  
ADD #7,R0 ;UPDATE RESULT POINTER  
CMP R1,10$ ;TEST IF DONE  
BNE 1$  
RTS R5 ;EXIT  
10$: G
```

```
;  
;ROTATE THE DATA RIGHT AND MAKE ASCII CHARACTER.  
SHUF6: BIC #177477,R2 ;MASK OFF BITS  
ASR R2  
ASR R2  
ASR R2  
SHUF3: ASR R2  
ASR R2  
ASR R2  
SHUF0: BIC #177770,R2 ;MASK OFF BITS  
ADD #60,R2 ;MAKE ASCII  
MOVB R2,-(R0) ;LOAD RESULT BYTE  
RTS PC ;EXIT
```

```

2917
2918 043362 012700 062036
2919 043366 013702 041712
2920 043372 010204
2921 043374 062704 000017
2922 043400 012001
2923 043402 061001
2924 043404 163701 041716
2925 043410 163701 041716
2926 043414 010137 041706
2927 043420 010146
2928 043422 012746 000316
2929 043426 004737 036510
2930 043432 012637 043544
2931 043436 012637 043546
2932
2933 043442 013746 043544
2934 043446 013746 043546
2935 043452 013746 041716
2936 043456 004737 036622
2937 043462 102004
2938 043464 000000
2939 043466 000000
2940 043470 000240
2941 043472 000240
2942 043474 012637 041704
2943 043500 012637 041702
2944
2945
2946
2947 043504 111203
2948 043506 000240
2949 043510 042703 177400
2950 043514 163703 041702
2951 043520 110322
2952
2953
2954
2955 043522 062700 000036
2956 043526 011001
2957 043530 063701 041706
2958 043534 010110
2959 043536 020204
2960 043540 001317
2961
2962 043542 000207
2963
2964 043544 000000
2965 043546 000000

;SUBROUTINE TO ADJUST THE LSB VALUES
ADJFIX: MOV #BUF1+36,R0 ;GET BUFFER POINTER
        MOV ROMPNT,R2 ;GET ROM POINTER
        MOV R2,R4
        ADD #17,R4 ;MAKE END VALUE
10$: MOV (R0)+,R1 ;GET LSB (17) M-1
     ADD (R0),R1 ;ADD LSB (00) M
     SUB AVGVAL,R1 ;SUB 2 LSB AVG.
     SUB AVGVAL,R1
     MOV R1,LSBSVW ;SAVE FOR LATER USAGE
     MOV R1,-(SP) ;SAVE ON STACK
     MOV #316,-(SP) ;MULTI BY 316 (8)
     JSR PC,$MULT
     MOV (SP)+,100$ ;SAVE LSW
     MOV (SP)+,101$ ;SAVE MSW
     MOV 100$,-(SP) ;
     MOV 101$,-(SP)
     MOV AVGVAL,-(SP)
     JSR PC,$DIV
     BVC 1$ ;BR IF NO ERROR
     HALT ;MATH ERROR
     HALT ; VALUE OUT OF RANGE
     NOP
     NOP
1$: MOV (SP)+,LSBSVR ;SAVE INTGR. REMAINDER
   MOV (SP)+,LSBSVQ ;SAVE QUOTIENT

;NOW ADD THE QUOTIENT TO THE MEMORY RAM VALUE
        MOVB (R2),R3 ;GET CURRENT MEMORY RAM VALUE
        NOP
        BIC #177400,R3 ;CLEAR OFF UPPER BITS
        SUB LSBSVQ,R3 ;ADD THE QUOTIENT
        MOVB R3,(R2)+ ;RELOAD MEMORY RAM VALUE

;NOW UPDATE 'M' VALUE POINTER
        ADD #36,R0 ;UPDATE POINTER
        MOV (R0),R1 ;GET VALUE
        ADD LSBSVW,R1 ;CORRECT THE VALUE (#10)
        MOV R1,(R0) ;RESTORE FOR NEXT USAGE
        CMP R2,R4 ;TEST WHEN FINISHED
        BNE 10$ ;BR UNTIL DONE
        RTS PC ;EXIT

100$: 0
101$: 0

```

```

2967
2968
2969 043550 005037 041710      ;:DIFFERENTIAL LINEARITY
2970 043554 112737 000061 040743 DIFLIN: CLR DIFEX1      ;CLEAR EXIT FLAG
2971 043562 105737 001134      MOVB #'1,GAIN1      ;LOAD GAIN MESSAGE VALUE
2972 043566 001007      TSTB $AUTOB        ;TEST IF UNDER MONITOR
2973 043570 104401 040700      BNE 10$            ;BR IF YES
2974 043574 005737 043550      TYPE, GAIN
2975 043600 100402      TST DIFLIN        ;TEST BIT 15
2976 043602 104401 050764      BMI 10$            ;DONT TELL OPER. ABOUT A017
2977 043606 012737 000036 046004 10$: MOV #36,PRIME1      ;TELL OPERATOR ABOUT A017
2978 043614 004537 046010      JSR R5,LIMITS      ;LOAD RESOLUTION AND GAIN
2979 043620 046074      G1LIMO             ;LOAD DIFLIN TOLERANCE
2980 043622 046114      G1LIM1             ;G1 USER LIMIT
2981 043624 012737 043634 046006 MOV #1$,DIFEXO      ;G1 OPTION AREA
2982 043632 000433      BR DIFLNO          ;LOAD RETURN ADDRESS
2983 043634 112737 000062 040743 1$: MOVB #'2,GAIN1      ;LOAD GAIN MESSAGE VALUE
2984 043642 012737 002036 046004 MOV #2036,PRIME1    ;LOAD RESOLUTION AND GAIN
2985 043650 004537 046010      JSR R5,LIMITS      ;LOAD DIFLIN TOLERANCE
2986 043654 046134      G2LIMO             ;G2 USER LIMIT
2987 043656 046154      G2LIM1             ;G2 OPTION AREA
2988 043660 012737 043702 046006 MOV #2$,DIFEXO      ;LOAD RETURN ADDRESS
2989 043666 105737 001134      TSTB $AUTOB        ;RUNNING UNDER MONITOR
2990 043672 001002      BNE 11$            ;BR IF YES
2991 043674 104401 040700      TYPE, GAIN
2992 043700 000410      BR DIFLNO          ;ENTER VIA 'F' SELECTION ?
2993 043702 022737 177777 050024 2$: CMP #-1,RUNDIF    ;BR IF NOT
2994 043710 001002      BNE 3$             ;YES REPORT END OF PASS
2995 043712 000137 030102      JMP $EOP
2996 043716 000137 002646      JMP RBEGO
2997
2998
2999
3000 043722 012700 041716      ;DIFLIN ROUTINE
3001 043726 005020      DIFLNO: MOV #AVGVAL,RO ;LOAD CLEARING POINTER
3002 043730 022700 041750      DIFLNX: CLR (RO)+
3003 043734 001374      CMP #FIRST+2,RO   ;FINISHED ?
3004 043736 012700 060000      BNE DIFLNX        ;BR IF NOT DONE
3005 043742 005020      MOV #BUFO,RO      ;LOAD BUFFER POINTER
3006 043744 022700 064000      1$: CLR (RO)+      ;CLEAR THE DATA BUFFER
3007 043750 001374      CMP #BUF2,RO      ;TEST IF FINISHED
3008      BNE 1$           ;BR IF NOT
3009 043752 004737 045632      ;LOOK FOR THE DATA REGION FOR TO
3010      JSR PC,STDATA    ;FIND THE NON-ZERO TO ZERO TRANS. DATA
3011 043756 012777 000000 135760 ;DATA HAS NOW GONE INTO THE GOOD 0 DATA REGION
3012 043764 012777 000025 135746 MOV #0,@KWPSR      ;INIT THE CLOCK PRESET
3013      MOV #25,@KWCSR  ;ENABLE THE CLOCK
3014 043772 004737 045476      ;NOW TIME 'T1' LENGTH
3015 043776 022702 000000      2$: JSR PC,LISTDT   ;GET CURRENT DATA VALUE
3016 044002 001773      CMP #0,R2         ;TEST FOR EXIT OF 0 DATA REGION
3017 044004 105077 135732      BEQ 2$           ;BR UNTILL DONE
3018 044010 152777 000002 135724 CLRB @KWCSR1
3019 044016 017737 135722 045670 BISB #BIT1,@KWCSR1 ;FIRE ST2
3020      MOV @KWPSR,DIFT1 ;SAVE COUNTER VALUE
3021 044024 004737 045476      ;NOW TIME 'T2' LENGTH
3022 044030 022702 000377      3$: JSR PC,LISTDT   ;GET CURRENT DATA VALUE
      CMP #377,R2     ;TEST FOR ENTRY INTO MAX DATA REGION

```

3023	044034	001373			BNE	3\$:BR IF NOT
3024	044036	105077	135700		CLRB	@KWCSR1		
3025	044042	152777	000002	135672	BISB	#BIT1,@KWCSR1		:FIRE ST2
3026	044050	017737	135670	045672	MCV	@KWPSR,DIFT2		:SAVE COUNTER VALUE
3027					:NOW TIME 'T3' LENGTH			
3028	044056	004737	045476		4\$: JSR	PC,LISTDT		:GET CURRENT DATA VALUE
3029	044062	022702	000377		CMP	#377,R2		:TEST FOR EXIT OF MAX DATA REGION
3030	044066	001773			BEQ	4\$:BR IF NOT
3031	044070	105077	135646		CLRB	@KWCSR1		
3032	044074	152777	000002	135640	BISB	#BIT1,@KWCSR1		:FIRE ST2
3033	044102	017737	135636	045674	MOV	@KWPSR,DIFT3		:SAVE COUNTER VALUE
3034	044110	005077	135624		CLR	@KWCSR		:STOP CLOCK
3035					:NOW DETERMINE THE AVG TIME (T3+T2)/2 AND SAVE IN 'HT2T3'			
3036	044114	013700	045672		MOV	DIFT2,R0		:GET T2 VALUE
3037	044120	063700	045674		ADD	DIFT3,R0		:ADD T3 VALUE
3038	044124	006000			ROR	R0		:/2
3039	044126	010037	045676		MOV	R0,HT2T3		:SAVE FOR LATER USE
3040					:NOW RETURN TO LIST MODE AND LOOK FOR THE NON-ZERO TO ZERO EDGE AGAIN			
3041	044132	004737	045632		DIFLN1: JSR	PC,STDATO		:FIND THE EDGE
3042					:NOW START THE CLOCK USING THE AVG TIME			
3043					:AND DO AN MATRIX MODE COLLECTION			
3044	044136	013700	045676		MOV	HT2T3,R0		:GET AVG CLOCK
3045	044142	005400			NEG	R0		:FIX FOR CLOCK PRESET REG
3046	044144	010077	135574		MOV	R0,@KWPSR		:LOAD CLOCK PRESET
3047	044150	012777	000020	135562	MOV	#20,@KWCSR		:PRIME CLOCK
3048	044156	012777	004000	135572	MOV	#CLRALL,@SFR		:INIT THE NCV11
3049	044164	012777	060000	135556	MOV	#BUFO,@OFF		:LOAD OFFSET
3050	044172	005077	135554		CLR	@WCR		:CLEAR Z COUNTER
3051	044176	005077	135552		CLR	@BAR		:
3052	044202	013777	046004	135536	MOV	PRIME1,@CSR		:SET MODE AND RES.
3053	044210	052777	000022	135540	BIS	#TESTZ!BIT4,@SFR		:SET TESTZ AND TESTX
3054	044216	052777	000001	135522	BIS	#BIT0,@CSR		:ENABLE NCV11
3055	044224	052777	000001	135506	BIS	#BIT0,@KWCSR		:ENABLE CLOCK
3056	044232	105777	135502		1\$: TSTB	@KWCSR		:WAIT FOR CLOCK FLAG
3057	044236	100375			BPL	1\$		
3058	044240	052777	000400	135510	BIS	#ENDDMA,@SFR		:STOP DMA XFR

```

3060 ;THE MATRIX MODE XFR IF NOW COMPLETED
3061 ;NOW ADD THE CURRENT BYTE DATA COLLECTION INTO A TOTAL INCREMENT BUFFER
3062 044246 005037 050016 CLR $TEMP1
3063 044252 012700 060000 MOV #BUFO,R0
3064 044256 012701 062000 MOV #BUF1,R1
3065
3066 044262 111037 050016 2$: MOVB (R0),$TEMP1 ;GET THE WORD
3067 044266 105020 CLR (R0)+ ;CLEAR THE BYTE TABLE ENTRY
3068 044270 063721 050016 ADD $TEMP1,(R1)+ ;UPDATE THE LIST
3069 044274 103003 BCC 3$ ;BR IF NO OVERFLOW
3070 044276 005741 TST -(R1) ;POSITON POINTER
3071 044300 012721 177777 MOV #-1,(R1)+ ;SET TO 177777
3072 044304 022701 064000 3$: CMP #BUF2,R1 ;FINISHED?
3073 044310 001364 BNE 2$ ;BR IF NOT
3074 ;GET THE AVERAGE OF THE 128 CENTER STATES
3075 044312 012700 062100 MOV #BUF1+64.,R0 ;GET INITIAL POINTER
3076 044316 005001 CLR R1
3077 044320 005002 CLR R2
3078 044322 062001 4$: ADD (R0)+,R1 ;GET A VALUE
3079 044324 103001 BCC 5$ ;BR IF NO CARRY OVERFLOW
3080 044326 005202 INC R2 ;UPDATE MSW
3081 044330 022700 062300 5$: CMP #BUF1+192.,R0 ;FINISHED THE BUFFER
3082 044334 001372 BNE 4$ ;BR IF NOT DONE AVERAGE
3087 044336 000241 CLC
(1) 044340 006002 ROR R2 ;ROTATE MSW
(1) 044342 006001 ROR R1 ;INTO LSW
(1) 044344 000241 CLC
(1) 044346 006002 ROR R2 ;ROTATE MSW
(1) 044350 006001 ROR R1 ;INTO LSW
(1) 044352 000241 CLC
(1) 044354 006002 ROR R2 ;ROTATE MSW
(1) 044356 006001 ROR R1 ;INTO LSW
(1) 044360 000241 CLC
(1) 044362 006002 ROR R2 ;ROTATE MSW
(1) 044364 006001 ROR R1 ;INTO LSW
(1) 044366 000241 CLC
(1) 044370 006002 ROR R2 ;ROTATE MSW
(1) 044372 006001 ROR R1 ;INTO LSW
(1) 044374 000241 CLC
(1) 044376 006002 ROR R2 ;ROTATE MSW
(1) 044400 006001 ROR R1 ;INTO LSW
3088 044402 010137 041716 MOV R1,AVGVAL ;SAVE THE AVERAGE
3089 044406 004737 045700 JSR PC,CTRLCG ;TEST FOR ^C OR ^G
3090 044412 023737 041716 041714 CMP AVGVAL,NOMIAL ;TEST IF AVG. IF >
3091 044420 103644 BLO DIFLN1 ;BR IF NOT
3092

```

```

3094 ;READ THE TOTAL INCREMENT BUFFER AND DETERMINE IF ANY VALUES OUT OF RANGE
3095
3096 044422 013737 041716 045116 READ1: MOV AVGVAL,101$ ;GET AVERAGE
3097 044430 012700 000001 MOV #1,R0
3098 044434 012701 062002 MOV #BUF1+2,R1
3099 044440 012137 045114 1$: MOV (R1)+,100$ ;GET A WORD
3100 044444 163737 045116 045114 SUB 101$,100$ ;REMOVE THE AVERAGE
3101 044452 100006 BPL 2$
3102 044454 005137 045114 COM 100$
3103 044460 112737 000055 041556 MOVB #'-,SGNVL1 ;INSERT '-' SIGN
3104 044466 000403 BR 3$
3105 044470 112737 000053 041556 2$: MOVB #'+,SGNVL1 ;INSERT '+' SIGN
3106 044476 013746 045114 3$: MOV 100$,-(SP)
3107 044502 012746 001750 MOV #1000.,-(SP)
3108 044506 004737 036510 JSR PC,$MULT ;MULTIPLY NUM-AVG BY 1000.
3109 044512 012637 045132 MOV (SP)+,107$ ;GET RESULT
3110 044516 012637 045134 MOV (SP)+,110$
3111 044522 013746 045132 MOV 107$,-(SP) ;DIVIDE RESULT
3112 044526 013746 045134 MOV 110$,-(SP)
3113 044532 013746 045116 MOV 101$,-(SP) ;DIVIDE AGAIN
3114 044536 004737 036622 JSR PC,$DIV
3115 044542 102003 BVC 4$
3116 044544 000000 HALT ;FATAL ERROR DURING CAL. OF ERROR TOLERANCES
3117 044546 000240 NOP
3118 044550 000240 NOP
3119 044552 012637 045124 4$: MOV (SP)+,104$ ;GET REMAINDER
3120 044556 012637 045126 MOV (SP)+,105$ ;GET QOUT
3121 044562 013702 045116 MOV 101$,R2 ;ROUND UP IF NEEDED
3122 044566 006202 ASR R2
3123 044570 023702 045124 CMP 104$,R2 ;COMPARE RESULT
3124 044574 002402 BLT 5$
3125 044576 005237 045126 INC 105$ ;ROUND UP
3126 044602 000240 5$: NOP
3127 ;DIFLIN ERROR CHECK
3128
3129 044604 023737 045126 046056 6$: CMP 105$,NORTOL ;TEST AGAINST NORMAL
3130 044612 103446 BLO 50$ ;BR IF OK
3131 044614 023737 045126 046062 CMP 105$,NARTOL ;TEST AGAINST WIDE/NARROW TOLERANCE
3132 044622 103416 BLO 20$ ;BR IF OK
3133 044624 023737 045126 046066 CMP 105$,NURTOL ;TEST AGAINST LARGE/SMALL TOLERANCE
3134 044632 103424 BLO 21$ ;BR IF OK
3135 044634 122737 000053 041556 CMPB #'+,SGNVL1 ;TEST IF +
3136 044642 001403 BEQ 7$
3137 044644 005237 041722 INC BELOW ;COUNT THE LOWER VALUE
3138 044650 000435 BR 51$
3139 044652 005237 041724 7$: INC ABOVE ;COUNT THE HIGHER VALUE
3140 044656 000432 BR 51$
3141 044660 122737 000053 041556 20$: CMPB #'+,SGNVL1 ;TEST IF +
3142 044666 001403 BEQ 22$
3143 044670 005237 041734 INC NARB ;COUNT THE LOWER
3144 044674 000423 BR 51$
3145 044676 005237 041740 22$: INC WIDB ;COUNT THE HIGHER
3146 044702 000420 BR 51$
3147 044704 122737 000053 041556 21$: CMPB #'+,SGNVL1 ;TEST IF +
3148 044712 001403 BEQ 23$
3149 044714 005237 041732 INC NARA ;COUNT THE LOWER

```

```

3150 044720 000411
3151 044722 005237 041736      23$:  BR      51$
3152 044726 000406              INC      WIDA      ;COUNT THE HIGHER
3153 044730 005237 041726      50$:  BR      51$
3154 044734 122737 000115 052412  INC      NORMAL    ;COUNT THE NORMAL
3155 044742 001054              CMPB     #'M,RUNIT  ;TEST IF FORCE REPORT
3156 044744 105737 001134      51$:  BNE      77$
3157 044750 001051              TSTB     $AUTOB    ;TEST IF MONITOR ?
3158              BNE      77$      ;BR IF YES
3159 044752 005737 041746      ;REPORT THE STATE INFORMATION
3160 044756 001013              TST      FIRST     ;FIRST TYPEOUT ?
3161 044760 005237 041746      BNE      10$       ;BR IF YES
3162 044764 104401 041416      INC      FIRST     ;SET FLAG
3163 044770 013746 041716      TYPE    ,AVRGO    ;TELL THE OPERATOR THE AVERG.
3164 044774 104405              MOV      AVGVAL,-(SP)
3165 044776 104401 041570      TYPDS
3166 045002 104401 041102      TYPE,   RTN
3167 045006 010046      10$:  TYPE,   STATE    ;ADD HEADER INFO
3168 045010 104403              MOV      RO,-(SP)  ;LOAD STATE #
3169 045012 003      001
3170 045014 122737 000115 052412  .BYTE   3,1
3171 045022 001005              CMPB     #'M,RUNIT  ;TEST IF EXPAND OUTPUT SELECTED
3172 045024 104401 040747      BNE      11$
3173 045030 013746 045114      TYPE,   DASH      ;INSERT SPACING
3174 045034 104405              MOV      100$,-(SP) ;REPORT DIFFERENCE
3175 045036 013746 045126      11$:  TYPDS
3176 045042 004737 037236      MOV      105$,-(SP)
3177 045046 012602              JSR      PC,$SB2D  ;CONVERT TO ASCII
3178 045050 062702 000002      MOV      (SP)+,R2  ;GET RESULT POINTER TO MESSAGE
3179 045054 112237 041560      ADD      #2,R2     ;ADJUST POINTER
3180 045060 112237 041561      MOVB     (R2)+,PERTXT
3181 045064 112237 041563      MOVB     (R2)+,PERTXT+1 ;LOAD THE PERCENT REPORT
3182 045070 104401 041550      MOVB     (R2)+,PERTXT+3
3183              TYPE,   SGNVAL
3184 045074 004737 045700      77$:  JSR      PC,CTRLCG ;TEST IF CTRL C/G
3185 045100 005200              INC      RO
3186 045102 022700 000377      CMP      #255.,RO  ;TEST IF DONE
3187 045106 001414              BEQ      READ      ;BR IF DONE
3188 045110 000137 044440      JMP      1$        ;TRY NEXT VALUE
3189
3190 045114 000000      100$:  0
3191 045116 000000      101$:  0
3192 045120 000000      102$:  0
3193 045122 000000      103$:  0
3194 045124 000000      104$:  0
3195 045126 000000      105$:  0
3196 045130 000000      106$:  0
3197 045132 000000      107$:  0
3198 045134 000000      110$:  0
3199 045136 000000      111$:  0

```

```

3201
3202
3203 045140 005037 041720
3204 045144 013702 041724
3205 045150 063702 041722
3206 045154 120237 046070
3207 045160 101026
3208 045162 013702 041732
3209 045166 063702 041736
3210 045172 120237 046064
3211 045176 101017
3212 045200 013702 041734
3213 045204 063702 041740
3214 045210 120237 046060
3215 045214 101010
3216 045216 123737 046054 041726
3217 045224 101004
3218 045226 012737 041653 045354
3219 045234 000405
3220 045236 012737 041665 045354 1$:
3221 045244 005237 041720
3222 045250 013746 041722 2$:
3223 045254 104405
3224 045256 104401 041574
3225 045262 013702 041732
3226 045266 063702 041734
3227 045272 010246
3228 045274 104405
3229 045276 104401 041036
3230 045302 013746 041726
3231 045306 104405
3232 045310 104401 041613
3233 045314 013702 041736
3234 045320 063702 041740
3235 045324 010246
3236 045326 104405
3237 045330 104401 041061
3238 045334 013746 041724
3239 045340 104405
3240 045342 104401 041634
3241
3242
3243
3244 045346 104401 040700
3245 045352 104401
3246 045354 041653
3247

;NOW ACCOUNT FOR ALL THE ERRORS
READ: CLR DIFERR ;CLEAR FLAG
MOV ABOVE,R2 ;GET HIGH VALUE
ADD BELOW,R2 ;ADD LOW
CMPB R2,OBSCNT ;TEST IF EXCEEDS LIMIT <OBEASE ABOVE/BELOW>
BHI 1$ ;BR IF ERRORS
MOV NARA,R2 ;GET LOW VALUE
ADD WIDA,R2 ;ADD HIGH
CMPB R2,NURCNT ;TEST IF EXCEEDS LIMIT <LARGE ABOVE/BELOW>
BHI 1$
MOV NARB,R2 ;GET LOW VALUE
ADD WIDB,R2 ;ADD HIGH
CMPB R2,NARCNT ;TEST IF EXCEEDS LIMIT <SMALL ABOVE/BELOW>
BHI 1$
CMPB NORCNT,NORMAL ;TEST IF MIN. NORMAL COUNT HAS BEEN HIT
BHI 1$
MOV #PASMSG,READ7 ;LOAD MESSAGE POINTER
BR 2$
MOV #FAIMSG,READ7 ;LOAD MESSAGE POINTER
INC DIFERR ;SET ERROR FLAG
MOV BELOW,-(SP) ;GET NO. OF STATES BELOW LIMITS
TYPDS ;REPORT VALUE
TYPE ,BELMSG ;TYPE MESSAGE
MOV NARA,R2 ;GET NARROW LOW
ADD NARB,R2 ;ADD NARROW HIGH
MOV R2,-(SP) ;GET NO. OF NARROW STATES
TYPDS
TYPE ,NARMSG ;TYPE MESSAGE
MOV NORMAL,-(SP) ;GET NORMAL COUNT
TYPDS
TYPE ,NORMSG ;REPORT NORMAL TEXT
MOV WIDA,R2 ;GET WIDE LOW
ADD WIDB,R2 ;ADD WIDE HIGH
MOV R2,-(SP)
TYPDS
TYPE ,WIDMSG ;TYPE MESSAGE
MOV ABOVE,-(SP) ;TYPE NO. OF STATES ABOVE LIMIT
TYPDS
TYPE ,ABOMSG ;TYPE MESSAGE

;REPORT PASS/FAIL MESSAGE
TYPE , GAIN ;RE-TYPE THE GAIN MESSAGE
TYPE
READ7: PASMSG ;REPORT PASS/FAIL
  
```



```

3249
3250 ;TYPE OUT MEMORY RAM AND RAM MEMORY VALUES IF SELECTED
3251 045356 032777 004000 133554 READXX: BIT #SW11,@SWR ;TEST IF THIS IS WANTED
3252 045364 001440 BEQ 4$ ;BR IF NOT
3253 045366 005737 041710 TST DIFEX1 ;TEST IF BLASTING MODE
3254 045372 001435 BEQ 4$ ;BR IF NOT
3255
3256 045374 012700 042134 MOV #ROMVAL,R0 ;LOAD INITIAL POINTER
3257 045400 004737 045430 1$: JSR PC,2$ ;TYPE OUT LINE
3258 045404 022700 042234 CMP #ROMVAL+100,R0 ;TEST IF ALL DONE
3259 045410 001373 BNE 1$ ;BR IF NOT
3260 045412 104401 041570 TYPE, RTN
3261 045416 104401 041570 TYPE, RTN
3262 045422 104401 041570 TYPE, RTN
3263 045426 000417 BR 4$
3264
3265 045430 012701 000020 2$: MOV #16.,R1 ;LOAD WIDE COUNTER
3266 045434 104401 041570 TYPE, RTN
3267 045440 112002 3$: MOVB (R0)+,R2 ;GET A BYTE
3268 045442 042702 177400 BIC #177400,R2 ;MASK OFF HIGHER BITS
3269 045446 010246 MOV R2,-(SP) ;LOAD VALUE
3270 045450 104403 TYPOS
3271 045452 003 001 .BYTE 3,1
3272 045454 104401 040752 TYPE, DASH+3
3273 045460 005301 DEC R1 ;FINISHED ?
3274 045462 001366 BNE 3$
3275 045464 000207 RTS PC ;EXIT
3276
3277
3278 045466 005037 041710 4$: CLR DIFEX1
3279 045472 000177 000310 JMP @DIFEXO ;EXIT THIS CRAZYNESS
3280
  
```

```

;DIF LIN SUBROUTINE LIST MODE DATA COLLECTOR
3282
3283
3284 045476 012777 004000 134252 LISTDT: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3285 045504 005077 134240 CLR @OFF ;PRIME CFFSET REG.
3286 045510 012777 177770 134234 MOV #-8,@WCR ;LOAD WC.
3287 045516 012777 060000 134230 MOV #BUFO,@BAR ;LOAD BUS ADDR.
3288 045524 012777 000000 134214 MOV #0,@CSR ;SET RES. AND MODE
3289 045532 032737 002000 046004 BIT #2000,PRIME1 ;TEST GAIN BIT
3290 045540 001403 BEQ 3$ ;BR IF CLEARED
3291 045542 052777 002000 134176 BIS #2000,@CSR ;SELECT GAIN = 2
3292 045550 052777 000002 134200 3$: BIS #TESTZ,@SFR ;SET Z PULSES
3293 045556 052777 000001 134162 BIS #BIT0,@CSR ;ENABLE THE DEVICE
3294
3295 045564 105777 134156 1$: TSTB @CSR ;WAIT FOR IDLE
3296 045570 100375 BPL 1$
3297 045572 012700 060000 MOV #BUFO,R0 ;LOAD POINTER TO NEW DATA
3298 045576 005002 CLR R2
3299 045600 012703 000010 MOV #8.,R3 ;LOAD COUNTER
3300 045604 011001 2$: MOV (R0),R1 ;GET DATA WORD
3301 045606 005020 CLR (R0)+ ;CLR BUFFER WORD
3302 045610 042701 177400 BIC #177400,R1 ;MAKE OFF BITS
3303 045614 060102 ADD R1,R2 ;UPDATE TOTAL
3304 045616 005303 DEC R3 ;FINISHED ?
3305 045620 001371 BNE 2$ ;BR IF NOT
3306 045622 006202 ASR R2
3307 045624 006202 ASR R2
3308 045626 006202 ASR R2
3309 045630 000207 RTS PC ;EXIT
3310 ;SUBROUTINE TO FIND THE TRUE EDGE OF ZERO DATA
3311 045632 012737 177777 045666 STDATO: MOV #-1,10$ ;SET ENTRY FLAG
3312 045640 004737 045476 1$: JSR PC,LISTDT ;GET DATA
3313 045644 005702 TST R2 ;CHECK DATA
3314 045646 001403 BEQ 2$ ;BR IF ZERO
3315 045650 005037 045666 CLR 10$ ;CLEAR ALREADY NON-ZERO DATA FLAG
3316 045654 000771 BR 1$
3317 ;DATA WAS ZERO - CHECK IF WE STARTED IN 0 DATA REGION
3318 045656 005737 045666 2$: TST 10$ ;TEST FLAG
3319 045662 001366 BNE 1$ ;BR IF NOT A GOOD TIME TO EXIT
3320 ;DATA HAS NOW GONE TO A GOOD ZERO DATA REGION
3321 ;NOW EXIT
3322 045664 000207 RTS PC ;BYEBYE
3323 045666 000000 10$: 0
3324 045670 000000 DIFT1: 0
3325 045672 000000 DIFT2: 0
3326 045674 000000 DIFT3: 0
3327 045676 000000 HT2T3: 0

```

```

3329 ;SUBROUTINE TO TEST IF OPERATOR TYPED CTRL C OR G
3330 045700 105777 133240 CTRLCG: TSTB @STKS ;INPUT FLAG
3331 045704 100033 BPL 2$ ;BR IF NON
3332 045706 017737 133234 045776 MOV @STKB,CTRCHA ;GET CHARACTER
3333 045714 042737 177600 045776 BIC #177600,CTRCHA ;MASK OFF BITS
3334 045722 022737 000003 045776 CMP #3,CTRCHA ;TEST IF CTRL C
3335 045730 001014 BNE 1$ ;BR IF NOT ^C
3336 045732 052777 000400 134016 BIS #ENDDMA,@SFR ;STOP NPR'S
3337 045740 000240 NOP
3338 045742 000240 NOP
3339 045744 000240 NOP
3340 045746 052777 004000 134002 BIS #CLRALL,@SFR ;CLEAR THE DEVICE
3341 045754 005726 TST (SP)+ ;CLEAN STACK
3342 045756 000137 002014 JMP RTRT ;RE-START
3343 045762 022737 000007 045776 1$: CMP #7,CTRCHA ;TEST IF CTRL G
3344 045770 001001 BNE 2$ ;BR IF NOT
3345 045772 104406 GTSWR ;GET SWR VALUE
3346 045774 000207 2$: RTS PC ;EXIT
3347 045776 000000 CTRCHA: 0
3348 046000 177777 PRIME: -1.
3349 046002 177400 PRIME0: 177400
3350 046004 000036 PRIME1: 36
3351 046006 000000 DIFEX0: 0
3352
3353 ;SUBROUTINE TO DETERMINE ERROR LIMITS FOR DIFLIN
3354 046010 012500 LIMITS: MOV (R5)+,R0 ;GET 1ST ARG.
3355 046012 005737 050026 TST WFMODE ;TEST IF ON TESTER
3356 046016 001402 BEQ 1$ ;BR IF NOT
3357 046020 012500 MOV (R5)+,R0 ;GET TESTER LIMITS
3358 046022 000401 BR 2$
3359 046024 005725 1$: TST (R5)+ ;BUMP ADDRESS
3360 046026 012701 046054 2$: MOV #NORCNT,R1 ;GET POINTER
3363 046032 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046034 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046036 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046040 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046042 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046044 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046046 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 046050 012021 MOV (R0)+,(R1)+ ;GET VALUE
3364 046052 000205 RTS R5 ;EXIT

```

```
3366 ;ACTUAL VALUES FOR DIF LIN ERROR TOLERANCE
3367
3368
3369 046054 000342 NORCNT: .WORD 226. ;'NORMAL'' MIN COUNT VALUE
3370 046056 000025 NORTOL: .WORD 21. ;'NORMAL'' TOLERANCE VALUE
3371 046060 000024 NARCNT: .WORD 20. ;'NARROW/WIDE'' MAX. COUNT VALUE
3372 046062 000063 NARTOL: .WORD 51. ;'NARROW/WIDE'' TOLERANCE VALUE
3373 046064 000010 NURCNT: .WORD 8. ;'LARGE/SMALL'' MAX. COUNT VALUE
3374 046066 000145 NURTOL: .WORD 101. ;'LARGE/SMALL'' TOLERANCE VALUE
3375 046070 000000 OBSCNT: .WORD 0. ;'OBESITY'' MAX. COUNT VALUE
3376 046072 000377 OBSTOL: .WORD 377 ;'OBESITY'' TOLERANCE VALUE
3377
3378 ;DIFLIN ERROR TOLERANCE GAIN = 1 USER
3379
3380 046074 000342 000025 G1LIMO: .WORD 226.,21. ;226 MIN. COUNT, =<2.0%
3381 046100 000024 000063 .WORD 20.,51. ;20 MAX. COUNT, =<5.0%
3382 046104 000010 000145 .WORD 8.,101. ;8 MAX. COUNT, =<10.%
3383 046110 000000 000377 .WORD 0.,377 ;0 MAX COUNT, >10%
3384
3385 ;DIFLIN ERROR TOLERANCE GAIN = 1 OPTION AREA
3386
3387 046114 000342 000024 G1LIM1: .WORD 226.,20. ;226 MIN COUNT, =<1.9%
3388 046120 000024 000061 .WORD 20.,49. ;20 MAX. COUNT, =<4.8%
3389 046124 000010 000140 .WORD 8.,96. ;8 MAX COUNT, =<9.5%
3390 046130 000000 000377 .WORD 0.,377 ;0 MAX COUNT, >9.5%
3391
3392 ;DIFLIN ERROR TOLERANCE GAIN = 2 USER
3393
3394 046134 000330 000031 G2LIMO: .WORD 216.,25. ;216. MIN COUNT, =<2.4%
3395 046140 000036 000075 .WORD 30.,61. ;30. MAX COUNT, =<6.0%
3396 046144 000010 000171 .WORD 8.,121. ;8. MAX COUNT, =<12.0%
3397 046150 000000 000377 .WORD 0.,377 ;0 MAX COUNT, >12.0%
3398
3399 ;DIFLIN ERROR TOLERANCE GAIN = 2 OPTION AREA
3400
3401 046154 000330 000025 G2LIM1: .WORD 216.,21. ;216. MIN COUNT, =<2.0%
3402 046160 000036 000063 .WORD 30.,51. ;30. MAX COUNT, =<5.0%
3403 046164 000010 000145 .WORD 8.,101. ;8. MAX COUNT, =<10.0%
3404 046170 000000 000377 .WORD 0.,377 ;0 MAX COUNT, >10.0%
3405
```

3407	046174	015	012	
3408	046176	020111	020075	047111
	046204	052111	040511	020114
	046212	030101	033461	040440
	046220	045104	051525	046524
	046226	047105	020124	051525
	046234	047111	020107	044124
	046242	020105	047101	046101
	046250	043517	052040	051505
	046256	042524	122	
3409	046261	015	012	
3410	046263	102	036440	041040
	046270	040514	052123	047111
	046276	020107	043117	052040
	046304	042510	046040	047111
	046312	040505	044522	054524
	046320	041440	051117	042522
	046326	052103	047511	020116
	046334	051120	046517	
3411	046340	015	012	
3412	046342	020103	020075	047503
	046350	052116	047522	020114
	046356	051120	043517	040522
	046364	020115	051120	046517
	046372	041040	040514	052123
	046400	047111	000107	
3413	046404	015	012	
3414	046406	020114	020075	047514
	046414	044507	020103	042524
	046422	052123	047440	020106
	046430	034115	031060	020066
	046436	047101	020104	034115
	046444	031460	066	
3415	046447	015	012	
3416	046451	104	036440	042040
	046456	043111	042506	042522
	046464	052116	040511	020114
	046472	044514	042516	051101
	046500	052111	020131	043117
	046506	052040	042510	040440
	046514	030460	067	
3417	046517	015	012	
3418	046521	106	036440	043040
	046526	047111	046101	040440
	046534	041503	050105	040524
	046542	041516	020105	046050
	046550	040440	042116	042040
	046556	051		
3419	046557	015	012	
3420	046561	101	036440	040440
	046566	045104	051525	046524
	046574	047105	020124	043117
	046602	040440	030460	020067
	046610	052101	043040	042511
	046616	042114	051440	052111
	046624	105		

LISTO: .BYTE 15,12
 .ASCII /I = INITIAL A017 ADJUSTMENT USING THE ANALOG TESTER/

.BYTE 15,12
 .ASCII /B = BLASTING OF THE LINEARITY CORRECTION PROM/

.BYTE 15,12
 .ASCII /C = CONTROL PROGRAM PROM BLASTING/

LIST: .BYTE 15,12
 .ASCII /L = LOGIC TEST OF M8026 AND M8036/

.BYTE 15,12
 .ASCII /D = DIFFERENTIAL LINEARITY OF THE A017/

.BYTE 15,12
 .ASCII /F = FINAL ACCEPTANCE (L AND D)/

.BYTE 15,12
 .ASCII /A = ADJUSTMENT OF A017 AT FIELD SITE/

CZNCCC NCV11 DIAGNOSTIC
CZNCCC.P11 29-SEP-80 09:35

MACY11 30G(1063) 29-SEP-80 10:33 PAGE 100-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

G 12

SEQ 0149

3421	046625	015	012
3422	046627	117	036440
	046634	042520	040522
	046642	053440	052111
	046650	052124	020131
	046656	040440	042104
	046664	051523	047040
	046672	047116	116
3423	046675	015	012
3424	046677	!23	036440
	046704	043117	053524
	046712	020105	053523
	046720	044103	051040
	046726	051511	042524
	046734	044103	047101
3425	046742	015	012
3426	046744	020110	020075
	046752	050114	052040
	046760	047440	042520
	046766	047524	020122
	046774	020104	042522
	047002	042520	052040
	047010	020123	044514
	047016	000	052123
3427	047017	015	012
3428	047021	056	000
3429			
3430	047024		
3431			

.BYTE 15,12
.ASCII /O = OPERATE WITH TTY AT ADDRESS NNNNNN/

.BYTE 15,12
.ASCII /S = SOFTWARE SWITCH REGISTER CHANGE/

.BYTE 15,12
.ASCII /H = HELP THE OPERATOR AND RETYPE THIS LIST/

LIST1: .BYTE 15,12
DOT: .ASCII /./

.EVEN

```

3433
3434 047024 000005          POTIME: RESET          ;CLEAR THE WORLD
3435 047026 104401 052160  TYPE, PRIM2          ;TELL OPERATOR ABOUT CABLES
3436 047032 012777 006035 132646  MOV #6035,@VTCHAR      ;HOME AND ERASE SCREEN
3437 047040 005077 132644  CLR @VTYPOS          ;CLEAR THE MAP P.C.
3438 047044 005037 045776  CLR CTRCHA          ;CLEAR TTY CHARACTER
3439
3440          ;ERASE THE SCREEN MAP
3441 047050 105777 132642  1$: TSTB @VTCSR          ;WAIT FOR READY
3442 047054 100375          BPL 1$              ;
3443 047056 052777 001000 132632  BIS #BIT9,@VTCSR      ;CLEAR THE MAP
3444 047064 105777 132626  2$: TSTB @VTCSR          ;WAIT FOR READY AGAIN
3445 047070 100375          BPL 2$              ;
3446 047072 042777 001000 132616  BIC #BIT9,@VTCSR      ;CLEAR THE ERASE BIT
3447          ;NOW LOAD THE REF. PATTERN INTO THE MAP
3448 047100 012701 010421  MOV #10421,R1          ;LOAD THE PIXEL VALUE
3449 047104 012700 003740  MOV #3740,R0           ;LOAD THE PIXEL ADDRESS
3450 047110 004737 047762  3$: JSR PC,DISPLY      ;LOAD THE DATA
3451 047114 005200          INC R0              ;UPDATE THE ADDRESS
3452 047116 022700 004000  CMP #4000,R0          ;FINISHED ?
3453 047122 001372          BNE 3$              ;BR UNTIL DONE
3454
3455 047124 012700 000000  MOV #0,R0            ;LOAD PIXEL ADDRESS
3456 047130 004737 047762  4$: JSR PC,DISPLY      ;LOAD THE DATA
3457 047134 005200          INC R0              ;UPDATE THE PIXEL ADDRESS
3458 047136 022700 000040  CMP #40,R0           ;TEST IF DONE
3459 047142 001372          BNE 4$              ;BR IF DONE THE VERTICAL
3460
3461 047144 012700 007740  MOV #7740,R0          ;LOAD PIXEL ADDRESS
3462 047150 004737 047762  5$: JSR PC,DISPLY      ;LOAD THE DATA
3463 047154 005200          INC R0              ;UPDATE THE PIXEL ADDRESS
3464 047156 022700 010000  CMP #10000,R0        ;TEST IF DONE
3465 047162 001372          BNE 5$              ;BR IF DONE
3466
3467          ;POSITION THE MAP AND DISPLAY
3468
3469 047164 012777 000005 132524  MOV #5,@VTCSR        ;LOAD ORGIN VALUE
3470 047172 012777 000060 132524  MOV #60,@VTINT       ;LOAD LUT 0
3471 047200 012777 000463 132516  MOV #463,@VTINT      ;LOAD LUT 1
3472 047206 052777 000400 132502  BIS #BIT8,@VTCSR     ;ENABLE THE MAP
  
```

```
3474 ;NOW SAMPLE THE CAMERA CHANNELS
3475 047214 SAMCAM:
3476 ;CAMERA 0, X INPUT
3477 047214 004537 047340 JSR R5,ADJCAM
3478 047220 050044 ADMSG0 ;CAM 0 MESSAGE
3479 047222 001730 DAC0 ;DAC ADDRESS
3480 047224 000 000 .BYTE 0,0 ;CAMERA 0, X AXIS
3481 ;DO CAMERA 1, X INPUT
3482 047226 004537 047340 JSR R5,ADJCAM
3483 047232 050230 ADMSG2 ;CAM 1, X MESSAGE
3484 047234 001732 DAC1 ;DAC ADDRESS
3485 047236 001 000 .BYTE 1,0 ;CAMERA 1, X AXIS
3486 ;DO CAMERA 2, X INPUT
3487 047240 004537 047340 JSR R5,ADJCAM
3488 047244 050414 ADMSG4 ;CAM 2, X MESSAGE
3489 047246 001734 DAC2 ;DAC ADDRESS
3490 047250 002 000 .BYTE 2,0 ;CAMERA 2, X AXIS
3491 ;DO CAMERA 3, X INPUT
3492 047252 004537 047340 JSR R5,ADJCAM
3493 047256 050600 ADMSG6 ;CAM 3, X MESSAGE
3494 047260 001736 DAC3 ;DAC ADDRESS
3495 047262 003 000 .BYTE 3,0 ;CAMERA 3, X AXIS
3496 ;NOW SAMPLE THE Y CAMERA CHANNELS
3497 ;DO CAMERA 0 Y INPUT
3498 047264 004537 047340 JSR R5,ADJCAM
3499 047270 050136 ADMSG1 ;CAM 0 Y MESSAGE
3500 047272 001732 DAC1 ;DAC ADDRESS
3501 047274 000 001 .BYTE 0,1 ;CAMERA 0, Y AXIS
3502 ;DO CAMERA 1, Y INPUT
3503 047276 004537 047340 JSR R5,ADJCAM
3504 047302 050322 ADMSG3 ;CAM 1, Y MESSAGE
3505 047304 001730 DAC0 ;DAC ADDRESS
3506 047306 001 001 .BYTE 1,1 ;CAMERA 1, Y AXIS
3507 ;DO CAMERA 2, Y INPUT
3508 047310 004537 047340 JSR R5,ADJCAM
3509 047314 050506 ADMSG5 ;CAM 2, Y MESSAGE
3510 047316 001736 DAC3 ;DAC ADDRESS
3511 047320 002 001 .BYTE 2,1 ;CAMERA 2, Y AXIS
3512 ;DO CAMERA 3, Y INPUT
3513 047322 004537 047340 JSR R5,ADJCAM
3514 047326 050672 ADMSG7 ;CAM 3, Y MESSAGE
3515 047330 001734 DAC2 ;DAC ADDRESS
3516 047332 003 001 .BYTE 3,1 ;CAMERA 3 Y AXIS
3517
3518 047334 000137 002646 JMP RBEGO
```



```

3520 ;REPORT THE MESSAGE FIRST
3521
3522 047340 012500 ADJCAM: MOV (R5)+,R0 ;GET ASCII POINTER
3523 047342 005777 132340 3$: TST @VTCHAR ;WAIT FOR READY
3524 047346 100375 BPL 3$
3525 047350 012777 012000 132336 1$: MOV #12000,@VTCXY ;POSITION THE CAHRACTERS
3526 047356 005777 132324 TST @VTCHAR ;WAIT FOR CHAR. READY
3527 047362 100375 BPL 1$
3528 047364 112077 132316 MOV (R0)+,@VTCHAR ;LOAD THE CHAR.
3529 047370 105710 TSTB (R0) ;TEST IF TERM.
3530 047372 001371 BNE 1$ ;BR IF NOT
3531
3532 047374 013537 050022 MOV @ (R5)+,DACADR ;SAVE ADDRESS
3533 047400 012537 050030 MOV (R5)+,CAMVAL ;SAVE CAMERA DATA
3534
3535 047404 012777 002315 000410 2$: MOV #2315,@DACADR ;PRESET DAC TO - GAIN INPUT
3536 047412 004737 047514 JSR PC,SAMDAT
3537 047416 010137 050036 MOV R1,VAL0 ;SAVE RESULTS
3538 047422 004737 047740 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3539 047426 012777 004000 000366 MOV #4000,@DACADR ;PRESET DAC TO OFFSET
3540 047434 004737 047514 JSR PC,SAMDAT ;SAMPLE THE DATA
3541 047440 010137 050040 MOV R1,VAL1 ;SAVE AVERAGE RESULTS
3542 047444 004737 047740 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3543 047450 012777 005463 000344 MOV #5463,@DACADR ;PRESET DAC TO + GAIN INPUT
3544 047456 004737 047514 JSR PC,SAMDAT ;SAMPLE THE DATA
3545 047462 010137 050042 MOV R1,VAL2 ;SAVE AVERAGE RESULTS
3546 047466 004737 047740 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3547 047472 004737 045700 JSR PC,CTRLCG ;TEST FOR CTRL C/G
3548 047476 122737 000103 045776 CMPB #'C,CTRCHA ;CHECK IF CHARACTER WAS A \C\
3549 047504 001337 BNE 2$ ;BR IF NOT
3550 047506 005037 045776 CLR CTRCHA ;CLEAR CHAR
3551 047512 000205 RTS R5 ;EXIT
3552
3553 ;COLLECT 64 SAMPLES FROM THE SELECTED CAMERA
3554 ;R1 CONTAINS THE AVERAGE
3555
3556 047514 012700 060000 SAMDAT: MOV #BUFO,R0
3557 047520 012701 062000 MOV #BUF1,R1
3558 047524 005020 1$: CLR (R0)+ ;CLEAR THE BUFFER
3559 047526 020001 CMP R0,R1 ;FINISHED ?
3560 047530 001375 BNE 1$ ;BR IF NOT
3561
3562 047532 012777 177700 132212 MOV #-100,@WCR ;LOAD WORD COUNT
3563 047540 012777 060000 132206 MOV #BUFO,@BAR ;LOAD ADDRESS
3564 047546 113777 050030 132212 MOVB CAMVAL,@CSRHB ;SELECT THE CAMERA
3565 047554 052777 000002 132174 BIS #TESTZ,@SFR ;SET TEST Z ENABLE
3566 047562 052777 000001 132156 BIS #BIT0,@CSR ;ENABLE NCV11
3567
3568 047570 105777 132152 2$: TSTB @CSR ;WAIT UNTIL DONE
3569 047574 100375 BPL 2$
3570
3571 047576 012777 004000 132152 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3572 047604 005037 050016 CLR $TEMP1
3573 047610 005037 050020 CLR $TEMP2
3574 047614 005001 CLR R1
3575 047616 012700 060000 MOV #BUFO,R0 ;LOAD BUFFER POINTER

```

```

3576 047622 105737 050031      TSTB    CAMVAL+1      ;TEST IF X OR Y DATA
3577 047626 001413              BEQ     5$            ;BR IF X
3578 047630 012037 050016      3$:    MOV     (R0)+,$TEMP1 ;GET THE DATA
3579 047634 113737 050017 050020  MOVB   $TEMP1+1,$TEMP2 ;GET HIGH BYTE DATA
3580 047642 063701 050020      ADD     $TEMP2,R1    ;UPDATE AVERAGE
3581 047646 022700 060200      CMP    #BUFO+200,R0 ;FINISHED DATA ?
3582 047652 001366              BNE    3$            ;BR IF NOT
3583 047654 000412              BR     10$           ;
3584 047656 012037 050016      5$:    MOV     (R0)+,$TEMP1 ;GET DATA
3585 047662 113737 050016 050020  MOVB   $TEMP1,$TEMP2
3586 047670 063701 050020      ADD     $TEMP2,R1    ;UPDATE THE AVERAGE
3587 047674 022700 060200      CMP    #BUFO+200,R0 ;FINISHED DATA ?
3588 047700 001366              BNE    5$            ;BR IF NOT
3589 047702 000240      10$:   NOP
3590 047704 000240      NOP
3594 047706 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047710 000240      NOP
(1) 047712 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047714 000240      NOP
(1) 047716 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047720 000240      NOP
(1) 047722 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047724 000240      NOP
(1) 047726 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047730 000240      NOP
(1) 047732 006201      ASR    R1            ;AVERAGE THE DATA
(1) 047734 000240      NOP
3595 047736 000207      RTS     PC            ;EXIT
3596
3597      ;SUBROUTINE TO LOAD X OR Y CROSSHAIRS
3598 047740 006201      FIXLIN: ASR    R1
3599 047742 000240      NOP
3600 047744 000240      NOP
3601 047746 000240      NOP
3602 047750 062701 000100      ADD     #100,R1
3603 047754 110177 131732      MOVB   R1,@VTXPOS
3604 047760 000207      RTS     PC
3605
3606      ;SUBROUTINE TO LOAD PIXEL DATA
3607 047762 042777 000400 131726  DISPLY: BIC    #400,@VTCSR ;DISABLE MAP
3608 047770 010077 131724      MOV    R0,@VTMAP    ;LOAD MAP PC
3609 047774 105777 131716      1$:    TSTB   @VTCSR    ;WAIT FOR MAP READY
3610 050000 100375      BPL    1$
3611 050002 010177 131714      MOV    R1,@VTPX    ;LOAD PIXEL DATA
3612 050006 052777 000400 131702  BIS    #400,@VTCSR ;ENABLE MAP
3613 050014 000207      RTS     PC
3614
3615 050016 000000      $TEMP1: 0
3616 050020 000000      $TEMP2: 0
3617 050022 000000      DACADR: 0
3618 050024 000000      RUNDIF: 0
3619 050026 000000      WFMODE: 0 ;WF OPTION AREA FLAG
3620 050030 000000      CAMVAL: 0
3621 050032 000000      CURENT: 0
3622 050034 000000      AGAN: 0
3623 050036 000000      VALO: 0

```

CZNCCC NCV11 DIAGNOSTIC
CZNCCC.P11 29-SEP-80 09:35

MACY11 30G(1063) 29-SEP-80 10:33 L 12 PAGE 103-2
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0154

3624	050040	000000			VAL1: 0
3625	050042	000000			VAL2: 0
3626	050044	040503	042515	040522	ADMSG0: .ASCII /CAMERA 00/
	050052	030040	060		
3627	050055	015	012		.BYTE 15,12
3628	050057	101	045104	051525	.ASCII /ADJUST R09 FOR X OFFSET/
	050064	020124	030122	020071	
	050072	047506	020122	020130	
	050100	043117	051506	052105	
3629	050106	015	012		.BYTE 15,12
3630	050110	042101	052512	052123	.ASCII /ADJUST R24 FOR X GAIN/
	050116	051040	032062	043040	
	050124	051117	054040	043440	
	050132	044501	000116		
3631	050136	040503	042515	040522	ADMSG1: .ASCII /CAMERA 00/
	050144	030040	060		
3632	050147	015	012		.BYTE 15,12
3633	050151	101	045104	051525	.ASCII /ADJUST R13 FOR Y OFFSET/
	050156	020124	030522	020063	
	050164	047506	020122	020131	
	050172	043117	051506	052105	
3634	050200	015	012		.BYTE 15,12
3635	050202	042101	052512	052123	.ASCII /ADJUST R20 FOR Y GAIN/
	050210	051040	030062	043040	
	050216	051117	054440	043440	
	050224	044501	000116		
3636	050230	040503	042515	040522	ADMSG2: .ASCII /CAMERA 01/
	050236	030040	061		
3637	050241	015	012		.BYTE 15,12
3638	050243	101	045104	051525	.ASCII /ADJUST R10 FOR X OFFSET/
	050250	020124	030522	020060	
	050256	047506	020122	020130	
	050264	043117	051506	052105	
3639	050272	015	012		.BYTE 15,12
3640	050274	042101	052512	052123	.ASCII /ADJUST R23 FOR X GAIN/
	050302	051040	031462	043040	
	050310	051117	054040	043440	
	050316	044501	000116		
3641	050322	040503	042515	040522	ADMSG3: .ASCII /CAMERA 01/
	050330	030040	061		
3642	050333	015	012		.BYTE 15,12
3643	050335	101	045104	051525	.ASCII /ADJUST R14 FOR Y OFFSET/
	050342	020124	030522	020064	
	050350	047506	020122	020131	
	050356	043117	051506	052105	
3644	050364	015	012		.BYTE 15,12
3645	050366	042101	052512	052123	.ASCII /ADJUST R19 FOR Y GAIN/
	050374	051040	034461	043040	
	050402	051117	054440	043440	
	050410	044501	000116		
3646	050414	040503	042515	040522	ADMSG4: .ASCII /CAMERA 02/
	050422	030040	062		
3647	050425	015	012		.BYTE 15,12
3648	050427	101	045104	051525	.ASCII /ADJUST R11 FOR X OFFSET/
	050434	020124	030522	020061	
	050442	047506	020122	020130	

3649	050450	043117	051506	052105		
3650	050456	015	012		.BYTE	15,12
	050460	042101	052512	052123	.ASCIZ	/ADJUST R22 FOR X GAIN/
	050466	051040	031062	043040		
	050474	051117	054040	043440		
3651	050502	044501	000116			
	050506	040503	042515	040522	ADMSG5: .ASCII	/CAMERA 02/
	050514	030040	062			
3652	050517	015	012		.BYTE	15,12
3653	050521	101	045104	051525	.ASCII	/ADJUST R15 FOR Y OFFSET/
	050526	020124	030522	020065		
	050534	047506	020122	020131		
	050542	043117	051506	052105		
3654	050550	015	012		.BYTE	15,12
3655	050552	042101	052512	052123	.ASCIZ	/ADJUST R18 FOR Y GAIN/
	050560	051040	034061	043040		
	050566	051117	054440	043440		
	050574	044501	000116			
3656	050600	040503	042515	040522	ADMSG6: .ASCII	/CAMERA 03/
	050606	030040	063			
3657	050611	015	012		.BYTE	15,12
3658	050613	101	045104	051525	.ASCII	/ADJUST R12 FOR X OFFSET/
	050620	020124	030522	020062		
	050626	047506	020122	020130		
	050634	043117	051506	052105		
3659	050642	015	012		.BYTE	15,12
3660	050644	042101	052512	052123	.ASCIZ	/ADJUST R21 FOR X GAIN/
	050652	051040	030462	043040		
	050660	051117	054040	043440		
	050666	044501	000116			
3661	050672	040503	042515	040522	ADMSG7: .ASCII	/CAMERA 03/
	050700	030040	063			
3662	050703	015	012		.BYTE	15,12
3663	050705	101	045104	051525	.ASCII	/ADJUST R16 FOR Y OFFSET/
	050712	020124	030522	020066		
	050720	047506	020122	020131		
	050726	043117	051506	052105		
3664	050734	015	012		.BYTE	15,12
3665	050736	042101	052512	052123	.ASCIZ	/ADJUST R17 FOR Y GAIN/
	050744	051040	033461	043040		
	050752	051117	054440	043440		
	050760	044501	000116			
3666						
3667	050764	015	012		PRIMO: .BYTE	15,12
3668	050766	044124	020105	042523	.ASCII	/THE SELF-TEST CONNECTOR MUST BE INSTALLED ON A017/
	050774	043114	052055	051505		
	051002	020124	047503	047116		
	051010	041505	047524	020122		
	051016	052515	052123	041040		
	051024	020105	047111	052123		
	051032	046101	042514	020104		
	051040	047117	040440	030460		
	051046	067				
3669	051047	015	012		.BYTE	15,12
3670	051051	124	042510	051440	.ASCII	/THE SWITCH ON A017 MUST BE IN 'MAINT.' POSITION (TOWARD THE I/O CONNECT
	051056	044527	041524	020110		

	051064	047117	040440	030460
	051072	020067	052515	052123
	051100	041040	020105	047111
	051106	021040	040515	047111
	051114	027124	020042	047520
	051122	044523	044524	047117
	051130	024040	047524	040527
	051136	042122	052040	042510
	051144	044440	047457	041440
	051152	047117	042516	052103
	051160	051117	051	
3671	051163	015	012	000
3672	051166	015	012	
3673	051170	044124	020105	042523
	051176	043114	052055	051505
	051204	020124	047503	047116
	051212	041505	047524	020122
	051220	052515	052123	041040
	051226	020105	042522	047515
	051234	042526	020104	051106
	051242	046517	052040	042510
	051250	040440	030460	067
3674	051255	015	012	
3675	051257	124	042510	051440
	051264	044527	041524	020110
	051272	047117	040440	030460
	051300	020067	052515	052123
	051306	041040	020105	047111
	051314	021040	050117	051105
	051322	021056	050040	051517
	051330	052111	047511	020116
	051336	040450	040527	020131
	051344	051106	046517	052040
	051352	042510	044440	047457
	051360	041440	047117	042516
	051366	052103	051117	051
3676	051373	015	012	000
3677	051376	047111	042523	052122
	051404	051040	046501	050055
	051412	045501	041440	041101
	051420	042514	044440	052116
	051426	020117	044124	020105
	051434	047523	045503	052105
	051442	047440	020116	044124
	051450	020105	030101	033461
3678	051456	015	012	
3679	051460	042523	020124	040522
	051466	026515	040520	020113
	051474	053523	052111	044103
	051502	051505	052040	020117
	051510	044442	021116	020054
	051516	047042	051117	021115
	051524	040440	042116	021040
	051532	043117	021106	
3680	051536	015	012	
3681	051540	047111	042523	052122

PRIM6: .BYTE 15,12,0
 .BYTE 15,12
 .ASCII /THE SELF-TEST CONNECTOR MUST BE REMOVED FROM THE A017/

.BYTE 15,12
 .ASCII \THE SWITCH ON A017 MUST BE IN 'OPER.'" POSITION (AWAY FROM THE I/O CONNE

PRIM5: .BYTE 15,12,0
 .ASCII /INSERT RAM-PAK CABLE INTO THE SOCKET ON THE A017/

.BYTE 15,12
 .ASCII /SET RAM-PAK SWITCHES TO 'IN', 'NORM' AND 'OFF'/'

PRIM3: .BYTE 15,12
 .ASCII /INSERT BLANK ROM INTO ROM BLASTER SOCKET/

	051546	041040	040514	045516	
	051554	051040	046517	044440	
	051562	052116	020117	047522	
	051570	020115	046102	051501	
	051576	042524	020122	047523	
	051604	045503	052105		
3682	051610	015	012		.BYTE 15,12
3683	051612	046120	040505	042523	.ASCII \PLEASE DEPRESS THE "I/O" AND "EXECUTE" BUTTONS TOGETHER\
	051620	042040	050105	042522	
	051626	051523	052040	042510	
	051634	021040	027511	021117	
	051642	040440	042116	021040	
	051650	054105	041505	052125	
	051656	021105	041040	052125	
	051664	047524	051516	052040	
	051672	043517	052105	042510	
	051700	122			
3684	051701	015	012		.BYTE 15,12
3685	051703	124	042510	032040	.ASCII \THE 4 PROCESS LEDS WILL LIGHT WHEN DONE CORRECTLY\
	051710	050040	047522	042503	
	051716	051523	046040	042105	
	051724	020123	044527	046114	
	051732	046040	043511	052110	
	051740	053440	042510	020116	
	051746	047504	042516	041440	
	051754	051117	042522	052103	
	051762	054514			
3686	051764	015	012	000	.BYTE 15,12,0
3687	051767	015	012		PRIM1: .BYTE 15,12
3688	051771	120	042514	051501	.ASCII /PLEASE REMOVE RAM-PAK CABLE/
	051776	020105	042522	047515	
	052004	042526	051040	046501	
	052012	050055	045501	041440	
	052020	041101	042514		
3689	052024	015	012		.BYTE 15,12
3690	052026	040440	042116	044440	.ASCII / AND INSERT THE BLASTED ROM INTO SOCKET ON A017 MODULE/
	052034	051516	051105	020124	
	052042	044124	020105	046102	
	052050	051501	042524	020104	
	052056	047522	020115	047111	
	052064	047524	051440	041517	
	052072	042513	020124	047117	
	052100	040440	030460	020067	
	052106	047515	052504	042514	
3691	052114	015	012		.BYTE 15,12
3692	052116	042504	051120	051505	.ASCII /DEPRESS THE "CR" KEY WHEN READY/
	052124	020123	044124	020105	
	052132	041442	021122	045440	
	052140	054505	053440	042510	
	052146	020116	042522	042101	
	052154	131			
3693	052155	015	012	000	.BYTE 15,12,0
3694	052160	015	012		PRIM2: .BYTE 15,12
3695	052162	050042	030512	020042	.ASCII /"PJ1" CABLE MUST BE CONNECTED TO THE A017 CONNECTOR/
	052170	040503	046102	020105	
	052176	052515	052123	041040	

	052204	020105	047503	047116		
	052212	041505	042524	020104		
	052220	047524	052040	042510		
	052226	040440	030460	020067		
	052234	047503	047116	041505		
	052242	047524	122			
3696	052245	015	012	000		
3697	052250	015	012		PRIM4:	.BYTE 15,12,0
3698	052252	050042	031512	020042		.BYTE 15,12
	052260	040503	046102	020105		.ASCII /"PJ3" CABLE MUST BE CONNECTED TO THE M8036 CONNECTOR/
	052266	052515	052123	041040		
	052274	020105	047503	047116		
	052302	041505	042524	020104		
	052310	047524	052040	042510		
	052316	046440	030070	033063		
	052324	041440	047117	042516		
	052332	052103	051117			
3699	052336	015	012	000		.BYTE 15,12,0
3700		052342				.EVEN
3701	052342	015	012	007	FATALO:	.BYTE 15,12,7 ;"RUNIT" MUST BE ON A WORD BOUNDRY
3702	052345	106	052101	046101		.ASCII /FATAL BUS TRAP WHEN PERFORMING TEST "/
	052352	041040	051525	052040		
	052360	040522	020120	044127		
	052366	047105	050040	051105		
	052374	047506	046522	047111		
	052402	020107	042524	052123		
	052410	021040				
3703	052412	021103	040411	020124	RUNIT:	.ASCIZ /C" AT LOCATION / ;"RUNIT" MUST BE A WORD BOUNDRY
	052420	047514	040503	044524		
	052426	047117	000040			
3704	052432	005015	051120	043517	RUNITA:	.ASCIZ <15><12>/PROGRAM RESTARTING/ ;"RUNIT" MUST BE A WORD BOUNDRY
	052440	040522	020115	042522		
	052446	052123	051101	044524		
	052454	043516	000			
3705	052457	015	012	007		.BYTE 15,12,7,0
	052462	000				
3706	052463	015	012		MSGMEM:	.BYTE 15,12
3707	052465	122	047125	044516		.ASCIZ /RUNNING WITH /
	052472	043516	053440	052111		
	052500	020110	000			
3708	052503	040	020113	043117	MSGK:	.ASCIZ / K OF MEMORY/<15><12>
	052510	046440	046505	051117		
	052516	006531	000012			
3709	052522	015	012	007	WARNO:	.BYTE 15,12,7
3710	052525	117	052103	046101		.ASCIZ /OCTAL ADDRESS OF TERMINAL = /
	052532	040440	042104	042522		
	052540	051523	047440	020106		
	052546	042524	046522	047111		
	052554	046101	036440	000040		
3711	052562	033	000		ESCP:	.BYTE 33,0
3712	052564	043113	027462	062	KFO:	.ASCII \KF2/2\
3713	052571	015	000			.BYTE 15,0
3714	052573	106	031515	060	FMO:	.ASCII \FM30\
3715	052577	015	000			.BYTE 15,0
3716	052601	104	030111	031457	DIO:	.ASCII \DIO/37\
	052606	067				

3717	052607	015	002			.BYTE	15,2
3718	052611						
3721	052611	060	060	060	DIDATA:	.BYTE	60,60,60,40
(1)	052614	040					
(1)	052615	060	060	060		.BYTE	60,60,60,40
(1)	052620	040					
(1)	052621	060	060	060		.BYTE	60,60,60,40
(1)	052624	040					
(1)	052625	060	060	060		.BYTE	60,60,60,40
(1)	052630	040					
(1)	052631	060	060	060		.BYTE	60,60,60,40
(1)	052634	040					
(1)	052635	060	060	060		.BYTE	60,60,60,40
(1)	052640	040					
(1)	052641	060	060	060		.BYTE	60,60,60,40
(1)	052644	040					
(1)	052645	060	060	060		.BYTE	60,60,60,40
(1)	052650	040					
(1)	052651	060	060	060		.BYTE	60,60,60,40
(1)	052654	040					
(1)	052655	060	060	060		.BYTE	60,60,60,40
(1)	052660	040					
(1)	052661	060	060	060		.BYTE	60,60,60,40
(1)	052664	040					
(1)	052665	060	060	060		.BYTE	60,60,60,40
(1)	052670	040					
(1)	052671	060	060	060		.BYTE	60,60,60,40
(1)	052674	040					
(1)	052675	060	060	060		.BYTE	60,60,60,40
(1)	052700	040					
(1)	052701	060	060	060		.BYTE	60,60,60,40
(1)	052704	040					
(1)	052705	060	060	060		.BYTE	60,60,60,40
(1)	052710	040					
(1)	052711	060	060	060		.BYTE	60,60,60,40
(1)	052714	040					
(1)	052715	060	060	060		.BYTE	60,60,60,40
(1)	052720	040					
(1)	052721	060	060	060		.BYTE	60,60,60,40
(1)	052724	040					
(1)	052725	060	060	060		.BYTE	60,60,60,40
(1)	052730	040					
(1)	052731	060	060	060		.BYTE	60,60,60,40
(1)	052734	040					
(1)	052735	060	060	060		.BYTE	60,60,60,40
(1)	052740	040					
(1)	052741	060	060	060		.BYTE	60,60,60,40
(1)	052744	040					
(1)	052745	060	060	060		.BYTE	60,60,60,40
(1)	052750	040					
(1)	052751	060	060	060		.BYTE	60,60,60,40
(1)	052754	040					
(1)	052755	060	060	060		.BYTE	60,60,60,40
(1)	052760	040					
(1)	052761	060	060	060		.BYTE	60,60,60,40
(1)	052764	040					


```

(1) 052765 060 060 060 .BYTE 60,60,60,40
(1) 052770 040
(1) 052771 060 060 060 .BYTE 60,60,60,40
(1) 052774 040
(1) 052775 060 060 060 .BYTE 60,60,60,40
(1) 053000 040
(1) 053001 060 060 060 .BYTE 60,60,60,40
(1) 053004 040
(1) 053005 060 060 060 .BYTE 60,60,60,40
(1) 053010 040
3722 053011 003 000 .BYTE 3,0
3723 053013 103 030115 031457 CM0: .ASCII \CM0/37\
      053020 067
3724 053021 015 000 .BYTE 15,0
3725 053023 103 000103 CC0: .ASCIZ /CC/
3726 053026 043520 027460 033463 PGO: .ASCII \PG0/37\
3727 053034 015 000 .BYTE 15,0
3728 053036 050132 ZP0: .ASCII /ZP/
3729 053040 015 000 .BYTE 15,0
3730 .EVEN
3731
3732 ;BUFFER AREA
3733
3734 060000 . =60000
3735 060000 001000 BUFO: .BLKW 512.
3736 062000 001000 BUF1: .BLKW 512.
3737 064000 000240 BUF2: NOP
3738
3739 000001 .END
  
```


BIT15 = 100000	13#	1141	1170	1178										
BIT2 = 000004	13#	129	1408	1423	1477	1490	1513	1533	1612	1613	1615	1616	1618	
	1619	1621	1622	1624	1625	1627	1628	2461						
BIT3 = 000010	13#	128	2474											
BIT4 = 000020	13#	433	563	565	574	597	608	618	631	642	650	660	685	
	691	716	723	738	750	756	797	801	1374	1408	1423	1434	1448	
	1458	1477	1490	1513	1533	1612	1613	1615	1616	1618	1619	1621	1622	
	1624	1625	1627	1628	1691	1770	1783	1806	1829	1851	1871	1891	2112	
	2223	2481	3053											
BIT5 = 000040	13#	1513	1533											
BIT6 = 000100	13#	774	1198	2429										
BIT7 = 000200	13#	311	388	400	409	574	581	705	738	801	826	845	850	
	1141	1150	1170	1179	1497	1533	2442							
BIT8 = 000400	13#	127	621	663	1316	1851	1891	2389	2394	2411	2416	2455	3472	
BIT9 = 001000	13#	1293	1360	1871	1891	2400	2405	2411	2416	2468	3443	3446		
BLAST 042274	205	2731#												
BLICNT 042132	2673*	2677	2680*	2684*	2704#									
BPTVEC= 000014	13#													
BRLEV 002002	118#	167*	1542	1566										
BTALK 042022	233	2682#												
BUFO 060000	1031*	1034	1053	1070	1072	1079	1084	1116	1122	1265	1267*	1277	1286	
	1288*	1298	1310	1312*	1320	1331	1333*	1343	1353	1355*	1365	1372*	1373	
	1385	1407	1411*	1412*	1416	1433	1441*	1442*	1464	1476	1484*	1502	1512	
	1520*	1550	1553*	1638	1643*	1644*	1649	1654	1659	1669	1675*	1677	1686	
	1692*	1693*	1699	1702	1706	1717*	1719	1733*	1735	1755*	1757	1769	1777*	
	1778*	1789	1909	1913	1942	1970	1979	1983	2009	2037	2044	2048	2057	
	2072	2091	2138	2186	2227*	2229	2244	2499	2523	2682	3004	3049	3063	
	3287	3297	3556	3563	3575	3581	3587	3735#						
BUF1 062000	1081	1105	2918	3064	3075	3081	3098	3557	3736*					
BUF2 064000	3006	3072	3737#											
BUSTRP 003212	152	258#												
CALRPT 037616	2420	2428#												
CAMUNP 040362	2432	2438	2445	2451	2458	2464	2471	2477	2484	2549#				
CAMVAL 050030	3533*	3564	3576	3620#										
CAMOG1 040444	2379	2433	2566#											
CAMOG2 040446	2384	2439	2567#											
CAMOTW 040505	2437	2578#												
CAMOTX 040466	2431	2576#												
CAM1G1 040450	2390	2446	2568#											
CAM1G2 040452	2395	2452	2569#											
CAM1TW 040543	2450	2582#												
CAM1TX 040524	2444	2580#												
CAM2G1 040454	2401	2459	2570#											
CAM2G2 040456	2406	2465	2571#											
CAM2TW 040601	2463	2586#												
CAM2TX 040562	2457	2584#												
CAM3G1 040460	2412	2472	2572#											
CAM3G2 040462	2417	2478	2573#											
CAM3TW 040637	2476	2590#												
CAM3TX 040620	2470	2588#												
CCO 053023	2781	2814	2835	3725#										
CKSWR = 104407	2341	2346	2366#											
CKTSWR 040064	2428	2434	2441	2447	2454	2460	2467	2473	2480	2488#				
CLRALL= 004000	126#	408	465	476	489	502	514	546	550	552	558	583	593	
	614	638	656	670	681	704	712	746	780	786	792	818	837	
	848	875	880	885	890	895	900	908	913	919	925	931	936	

CZNCCC NCV11 DIAGNOSTIC
CZNCCC.P11 29-SEP-80 09:35

MACY11 30G(1063) 29-SEP-80 10:33 PAGE 104-11
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0172

TST123	021260	1613	1615#					
TST124	021410	1615	1616#					
TST125	021540	1616	1618#					
TST126	021670	1618	1619#					
TST127	022020	1619	1621#					
TST13	004700	485#						
TST130	022150	1621	1622#					
TST131	022300	1622	1624#					
TST132	022430	1624	1625#					
TST133	022560	1625	1627#					
TST134	022710	1627	1628#					
TST135	023040	1628	1633#					
TST136	023226	1653	1658	1664#				
TST137	023356	1681	1684#					
TST14	004766	498#						
TST140	023544	1701	1705	1710#				
TST141	023674	1723	1726#					
TST142	024024	1739	1743#					
TST143	024164	1749	1761	1765#				
TST144	024370	1792	1795#					
TST145	024536	1797	1815	1818#				
TST146	024704	1820	1838	1842#				
TST147	025030	1844	1858	1862#				
TST15	005054	512#						
TST150	025154	1864	1878	1882#				
TST151	025300	1884	1898	1902#				
TST152	025670	1941	1951	1964	1966	1969	1974#	
TST153	026250	2008	2018	2025	2031	2033	2036	2042#
TST154	026472	2071	2079	2084#				
TST155	026766	2086	2125	2127	2132#			
TST156	027264	2134	2172	2174	2179#			
TST157	027500	2184	2211	2214#				
TST16	005314	548#						
TST160	027640	2221	2233	2237#				
TST161	030060	2242	2270	2274#				
TST17	005364	554	557#					
TST2	004022	366	374	381#				
TST20	005556	586	592#					
TST21	005730	604	611	613#				
TST22	006114	626	634	637#				
TST23	006254	649	653	655#				
TST24	006370	666	669#					
TST25	006462	677	680#					
TST26	006654	701	708	711#				
TST27	007054	734	741	745#				
TST3	004124	397#						
TST30	007324	791#						
TST31	007466	806	810	817#				
TST32	007574	833	836#					
TST33	007740	859	874#					
TST34	010034	876	879#					
TST35	010130	881	884#					
TST36	010224	886	889#					
TST37	010320	891	894#					
TST4	004172	403	406#					
TST40	010414	896	899#					

\$MAMS4	001240	32#															
\$MBADR	001002	31#															
\$MFLG	035704	2350#*															
\$MNEW	033553	2336#															
\$MSGAD	001210	32#	2350*														
\$MSGLG	001212	32#	2350*														
\$MSGTY	001174	32#	2350*														
\$MSWR	033542	2336#															
\$MTYP1	001225	32#															
\$MTYP2	001231	32#															
\$MTYP3	001235	32#															
\$MTYP4	001241	32#															
\$MULT	036510	2359#	2929	3108													
\$MXCNT	034374	2341#															
\$NULL	001154	32#	2342														
\$NWTST=	000001	350#	381#	397#	406#	416#	431#	445#	455#	463#	472#	485#	498#	512#			
		548#	557#	592#	613#	637#	655#	669#	680#	711#	745#	791#	817#	836#			
		874#	879#	884#	889#	894#	899#	907#	912#	918#	924#	930#	935#	956#			
		957#	958#	960#	961#	962#	964#	965#	966#	968#	969#	970#	972#	973#			
		974#	976#	977#	978#	980#	981#	982#	984#	998#	1012#	1029#	1077#	1125#			
		1155#	1184#	1217#	1238#	1262#	1283#	1305#	1326#	1350#	1370#	1393#	1429#	1472#			
		1508#	1541#	1612#	1613#	1615#	1616#	1618#	1619#	1621#	1622#	1624#	1625#	1627#			
		1628#	1633#	1664#	1684#	1710#	1726#	1743#	1765#	1795#	1818#	1842#	1862#	1882#			
		1902#	1974#	2042#	2084#	2132#	2179#	2214#	2237#	2274#							
\$OCNT	035102	2344#*															
\$OMODE	035104	2344#*															
\$OVER	034360	2341#															
\$PASS	001202	32#	147*	299	1963	2030	2124	2171	2278*	2341							
\$PASTM	001006	31#															
\$POWER	036056	2351#															
\$PWRDN	035710	147	2351#														
\$PWARMG	036044	2351#															
\$PWRUP	035762	2351#															
\$QUES	001170	32#	253	2336	2342	2346											
\$RDCHR	033272	2336#	2366														
\$RDDEC=	***** U	2366															
\$RDLIN	033412	2336#	2366														
\$RDOCT	033564	2338#	2366														
\$RDSZ =	000010	2336#															
\$RESRE	036452	2357#	2366														
\$RTNAD	030202	2278#															
\$R2A =	***** U	2366															
\$SAVRE	036414	2357#	2366														
\$SAVR6	036054	2351#*															
\$SB2D	037236	2364#	3176														
\$SCOPE	034112	147	2341#														
\$SETUP=	000137	139#	147	149	2278	2336	2341	2346									
\$SIZE	036066	305	312	2353#													
\$SIZEX	036344	2353#															
\$STUP =	177777	139#															
\$SVLAD	034324	2341#															
\$SVPC =	000106	29#															
\$SWR =	167400	5#	12	21	32	147	350	381	397	406	416	431	445	455			
		463	472	485	498	512	548	557	592	613	637	655	669	680			
		711	745	791	817	836	874	879	884	889	894	899	907	912			
		918	924	930	935	956	957	958	960	961	962	964	965	966			

COMMEN ENDCOM ERROR	13# 13# 13#															
	525	530	535	540	545	555	571	578	587	603	607	612	625	630	635	
	648	654	667	678	700	709	733	742	766	781	804	811	827	835	846	
	853	860	877	882	887	892	897	902	910	915	921	927	933	938	956	
	957	958	960	961	962	964	965	966	968	969	970	972	973	974	976	
	977	978	980	981	982	995	1009	1023	1050	1056	1063	1068	1075	1097	1102	
	1108	1114	1120	1144	1148	1154	1173	1177	1183	1208	1236	1260	1281	1302	1324	
	1347	1369	1389	1419	1424	1451	1461	1468	1493	1500	1506	1529	1536	1577	1582	
	1612	1613	1615	1616	1618	1619	1621	1622	1624	1625	1627	1628	1656	1661	1682	
	1704	1708	1724	1740	1762	1787	1793	1814	1837	1859	1879	1899	1940	1950	1957	
	2007	2017	2024	2070	2078	2123	2170	2212	2234	2271						
ESCAPE	13#															
GETPRI	13#	2353	2361													
GETSWR	13#	149#														
JOYMAC	862#	875	880	885	890	895	900	908	913	919	925	931	936			
MULT	13#															
NEWTST	13#	350	381	397	406	416	431	445	455	463	472	485	498	512	548	
	557	592	613	637	655	669	680	711	745	791	817	836	874	879	884	
	889	894	899	907	912	918	924	930	935	956	957	958	960	961	962	
	964	965	966	968	969	970	972	973	974	976	977	978	980	981	982	
	984	998	1012	1029	1077	1125	1155	1184	1217	1238	1262	1283	1305	1326	1350	
	1370	1393	1429	1472	1508	1541	1612	1613	1615	1616	1618	1619	1621	1622	1624	
	1625	1627	1628	1633	1664	1684	1710	1726	1743	1765	1795	1818	1842	1862	1882	
	1902	1974	2042	2084	2132	2179	2214	2237	2274							
OFFBIT	1591#	1612	1613	1615	1616	1618	1619	1621	1622	1624	1625	1627	1628			
POP	13#	2338	2340	2350	2351	2357	2359	2361								
PUSH	13#	2338	2340	2350	2351	2357	2359	2361								
REPORT	13#															
RESLM	941#	956	957	958	960	961	962	964	965	966	968	969	970	972	973	
	974	976	977	978	980	981	982									
SCOPE	13#	350	381	397	406	416	431	445	455	463	472	485	498	512	548	
	557	592	613	637	655	669	680	711	745	791	817	836	874	879	884	
	889	894	899	907	912	918	924	930	935	956	957	958	960	961	962	
	964	965	966	968	969	970	972	973	974	976	977	978	980	981	982	
	984	998	1012	1029	1077	1125	1155	1184	1217	1238	1262	1283	1305	1326	1350	
	1370	1393	1429	1472	1508	1541	1612	1613	1615	1616	1618	1619	1621	1622	1624	
	1625	1627	1628	1633	1664	1684	1710	1726	1743	1765	1795	1818	1842	1862	1882	
	1902	1974	2042	2084	2132	2179	2214	2237	2274	2278						
SETPRI	13#															
SETTRA	2366#															
SETUP	13#	147														
SETZ	135#	599	620	644	662	674	688	719	753	799	1037	1088	1133	1162	1199	
	1227	1248	1377	1413	1437	1480	1516	1555	1612	1613	1615	1616	1618	1619	1621	
	1622	1624	1625	1627	1628	1642	1690	1716	1732	1754	1773	1995	2115	2160	2198	
	2226	2256														
SKIP	13#	300	323	340	366	374	390	403	412	422	429	439	451	460	468	
	480	493	506	524	529	534	539	544	554	569	577	586	602	604	606	
	611	626	629	634	649	653	666	677	701	708	734	741	767	782	803	
	806	810	833	852	859	876	881	886	891	896	901	909	914	920	926	
	932	937	956	957	958	960	961	962	964	965	966	968	969	970	972	
	973	974	976	977	978	980	981	982	994	1008	1022	1049	1051	1055	1062	
	1067	1074	1096	1098	1101	1107	1113	1119	1123	1145	1147	1153	1174	1176	1182	
	1209	1220	1235	1241	1259	1280	1301	1308	1323	1329	1346	1368	1388	1418	1450	
	1460	1467	1492	1499	1505	1530	1535	1578	1583	1612	1613	1615	1616	1618	1619	

CZNCCC NCV11 DIAGNOSTIC
CZNCCC.P11 29-SEP-80 09:35

MACY11 30G(1063) 29-SEP-80 10:33 PAGE 105-2
CROSS REFERENCE TABLE -- MACRO NAMES

N 14

SEQ 0182

.\$APT8	11#	32#	
.\$APTH	11#	31	
.\$APTY	11#	2350	
.\$CATC	7#	22	
.\$CKSW	9#		
.\$CMTA	7#	32	
.\$DB2D	10#	2363	
.\$DIV	10#	2361	
.\$EOP	7#	2278	
.\$ERRO	7#	2346	
.\$ERRT	9#	2348	
.\$MULT	10#	2359	
.\$PARM	8#		
.\$POWE	8#	2351	
.\$RDOC	10#	2338	
.\$READ	8#	2336	
.\$SAVE	8#	10#	2357
.\$SB2D	10#	2364	
.\$SCOP	8#	2341	
.\$SIZE	10#	2353	
.\$SPAC	8#		
.\$SWDO	8#		
.\$STRAP	8#	2366	
.\$TYPB	9#		
.\$TYPD	9#	2340	
.\$TYPE	7#	8#	2342
.\$TYPO	7#	2344	

. ABS. 064002 000 CON RW ABS LCL I

ERRORS DETECTED: 0

CZNCCC,CZNCCC/CRF=CZNCCC
RUN-TIME: 33 24 2 SECONDS
RUN-TIME RATIO: 286/60=4.7
CORE USED: 30K (59 PAGES)