

NCV11

NCV-11 DIAG
CZNCCB0

AH-E772B-MC

COPYRIGHT 78-80

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

The main body of the document consists of a 10x10 grid of small diagrams or tables. Each cell in the grid contains a small schematic or data table, which is too small to read clearly. The diagrams appear to be related to the NCV-11 diagnostic tool mentioned in the header.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-E771B-MC
PRODUCT NAME: CZNCCBO NCV-11 DIAGNOSTIC TEST
DATE: AUGUST 1979
MAINTAINER: DIAGNOSTIC ENGINEERING

(COPYRIGHT (C) 1978, 1979
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 '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

THE NCV11 DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST 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 OPTIGN 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      STARTING ADDRESS OF THE PROGRAM
          204      RESTART ADDRESS OF THE PROGRAM
          210      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

SEQ 0005

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

SEQ 0006

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																
RESO	REST	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	C	B	Z	G	Z	G	1	0	G	Z	G	Z	G	1	0	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	0	B	G	Z	G	Z	G	G	Z	G	Z	G	0
0	0	1	0	0	0	0	0	0	B	G	Z	G	Z	G	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 SE 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	DESCRIPTION
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 2 INPUTS CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN MATRIX MODE
613	T21 VERIFY 2 INPUTS CAUSE THE LOW 16 BITS OF 32 BIT COUNTER TO CHANGE
637	T22 VERIFY 2 INPUTS CAUSE THE LOW 24 BITS OF 32 BIT COUNTER TO CHANGE
655	T23 VERIFY 2 INPUTS CAUSE THE HIGH 8 BITS OF THE 32 BIT COUNTER TO CHANGE
669	T24 VERIFY 2 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 MAKER - 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	
1973	T152	DYNAMIC LIST MODE ADDRESS	
2041	T153	DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K	
2083	T154	ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY	
2131	T155	ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY	
2178	T156	VERIFY BIT 15 MATRIX ADDER INPUT	
2213	T157	VERIFY HIGH BYTE OPERATION OF THE 'TEST X'	
2236	T160	VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER	
2273	T161	DETERMINE IF DIFLIN IS TO BE RUN (F)	
2277		END OF PASS ROUTINE	
2279		ERROR ASCII MESSAGES	
2335		TTY INPUT ROUTINE	
2337		READ AN OCTAL NUMBER FROM THE TTY	
2339		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE	
2340		SCOPE HANDLER ROUTINE	
2341		TYPE ROUTINE	
2343		BINARY TO OCTAL (ASCII) AND TYPE	
2345		ERROR HANDLER ROUTINE	
2347		ERROR MESSAGE TYPEOUT ROUTINE	
2349		APT COMMUNICATIONS ROUTINE	
2350		POWER DOWN AND UP ROUTINES	
2352		ROUTINE TO SIZE MEMORY	
2356		SAVE AND RESTORE R0-R5 ROUTINES	
2358		INTEGER MULTIPLY ROUTINE	
2360		INTEGER DIVIDE ROUTINE	
2362		DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE	
2363		SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE	
2365		TRAP DECODER	
(3)		TRAP TABLE	
2371		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      167400      $SWR=167400
6      000001      $TN=1
12     .TITLE CZNCCB NCV11 DIAGNOSTIC
(1)    .*COPYRIGHT (C) 1979
(1)    .*DIGITAL EQUIPMENT CORP.
(1)    .*MAYNARD, MASS. 01754
(1)    .*
(1)    .*PROGRAM BY RAYMOND 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)
(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)
(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)
(1)    .*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000    R0= X0      ;;GENERAL REGISTER
(1)    000001    R1= X1      ;;GENERAL REGISTER
(1)    000002    R2= X2      ;;GENERAL REGISTER
(1)    000003    R3= X3      ;;GENERAL REGISTER
(1)    000004    R4= X4      ;;GENERAL REGISTER
(1)    000005    R5= X5      ;;GENERAL REGISTER
(1)    000006    R6= X6      ;;GENERAL REGISTER
(1)    000007    R7= X7      ;;GENERAL REGISTER
(1)    000006    SP= X6      ;;STACK POINTER
(1)    000007    PC= X7      ;;PROGRAM COUNTER
(1)
(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)      100000      :*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)      040000      SW15= 100000
(1)      020000      SW14= 40000
(1)      010000      SW13= 20000
(1)      004000      SW12= 10000
(1)      002000      SW11= 4000
(1)      001000      SW10= 2000
(1)      000400      SW09= 1000
(1)      000200      SW08= 400
(1)      000100      SW07= 200
(1)      000040      SW06= 100
(1)      000020      SW05= 40
(1)      000010      SW04= 20
(1)      000004      SW03= 10
(1)      000002      SW02= 4
(1)      000001      SW01= 2
(1)      000000      SW00= 1
(1)      .EQUIV      SW09,SW9
(1)      .EQUIV      SW08,SW8
(1)      .EQUIV      SW07,SW7
(1)      .EQUIV      SW06,SW6
(1)      .EQUIV      SW05,SW5
(1)      .EQUIV      SW04,SW4
(1)      .EQUIV      SW03,SW3
(1)      .EQUIV      SW02,SW2
(1)      .EQUIV      SW01,SW1
(1)      .EQUIV      SW00,SW0
(1)
(1)      :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)      100000      BIT15= 100000
(1)      040000      BIT14= 40000
(1)      020000      BIT13= 20000
(1)      010000      BIT12= 10000
(1)      004000      BIT11= 4000
(1)      002000      BIT10= 2000
(1)      001000      BIT09= 1000
(1)      000400      BIT08= 400
(1)      000200      BIT07= 200
(1)      000100      BIT06= 100
(1)      000040      BIT05= 40
(1)      000020      BIT04= 20
(1)      000010      BIT03= 10
(1)      000004      BIT02= 4
(1)      000002      BIT01= 2
(1)      000001      BIT00= 1
(1)      .EQUIV      BIT09,BIT9
(1)      .EQUIV      BIT08,BIT8
(1)      .EQUIV      BIT07,BIT7
(1)      .EQUIV      BIT06,BIT6
(1)      .EQUIV      BIT05,BIT5
(1)      .EQUIV      BIT04,BIT4
(1)      .EQUIV      BIT03,BIT3
(1)      .EQUIV      BIT02,BIT2
(1)      .EQUIV      BIT01,BIT1
(1)      .EQUIV      BIT00,BIT0
```

```
(1)
(1)
(1) 000004
(1) 000010
(1) 000014
(1) 000014
(1) 000014
(1) 000020
(1) 000024
(1) 000030
(1) 000034
(1) 000060
(1) 000064
(1) 000240
14
(1)
(1)
(1) 000250
(1)
(1)
(1) 177572
(1) 177574
(1) 177576
(1) 172516
(1)
(1)
(1) 177600
(1) 177602
(1) 177604
(1) 177606
(1) 177610
(1) 177612
(1) 177614
(1) 177616
(1)
(1)
(1) 177620
(1) 177622
(1) 177624
(1) 177626
(1) 177630
(1) 177632
(1) 177634
(1) 177636
(1)
(1)
(1) 177640
(1) 177642
(1) 177644
(1) 177646
(1) 177650

;*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14        ;;'T' BIT
TRTVEC= 14        ;;TRACE TRAP
BPTVEC= 14        ;;BREAKPOINT TRAP (BPT)
IOTVEC= 20        ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24        ;;POWER FAIL
EMTVEC= 30        ;;EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34        ;;'TRAP' TRAP
TKVEC= 60         ;;TTY KEYBOARD VECTOR
TPVEC= 64         ;;TTY PRINTER VECTOR
PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
.SBTTL MEMORY MANAGEMENT DEFINITIONS

;*KT11 VECTOR ADDRESS
MMVEC= 250

;*KT11 STATUS REGISTER ADDRESSES
SR0= 177572
SR1= 177574
SR2= 177576
SR3= 172516

;*USER 'I' PAGE DESCRIPTOR REGISTERS
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616

;*USER 'D' PAGE DESCRIPTOR REGISTORS
UDFDR0= 177620
UDPDR1= 177622
UDPDR2= 177624
UDPDR3= 177626
UDPDR4= 177630
UDPDR5= 177632
UDPDR6= 177634
UDPDR7= 177636

;*USER 'I' PAGE ADDRESS REGISTERS
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
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)      :      -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)      :      =174
(1) 000174 000000  DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000  SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
(1)      :
(1) 000200 000137 002034 .SBTTL STARTING ADDRESS(ES)
23 000204 000137 002014  JMP @WBEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
24 000210 000137 002022  JMP RTRT   ;RESTART ADDRESS
25      :
26      :      -100
27 000100 000104 000200 000002  JMP TESTER ;STARTING ADDRESS WHEN IN THE OPTION AREA
      :      104,200,2          ;B EVENT SAFE GUARD
  
```

29
(1)
(2)
(1)
(1) 000106
(1) 000046
(1) 000046 030166
(1) 000052 000052
(1) 000052 000000
(1) 000106
30 001000
31
(1)
(2)
(1)
(2)
(1) 001000
(1) 000024 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 001000
(1) 001000
(2)
(1)
(1)
(1)
(1) 001000
(1) 001000 000000
(1) 001002 001174
(1) 001004 000030
(1) 001006 000010
(1) 001010 000030
(1) 001012 000031

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

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

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 30     ;;RUN TIME OF LONGEST TEST
$PASTM: .WORD 10     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 30     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

32
 (1)
 (2)
 (1)
 (1)
 (1)
 (1)
 (1) 001100 001100
 (1) 001100 000000
 (1) 001102 000
 (1) 001103 000
 (1) 001104 000000
 (1) 001106 000000
 (1) 001110 000000
 (1) 001112 000000
 (1) 001114 000
 (1) 001115 001
 (1) 001116 000000
 (1) 001120 000000
 (1) 001122 000000
 (1) 001124 000000
 (1) 001126 000000
 (1) 001130 000000
 (1) 001132 000000
 (1) 001134 000
 (1) 001135 000
 (1) 001136 000000
 (1) 001140 177570
 (1) 001142 177570
 (1) 001144 177560
 (1) 001146 177562
 (1) 001150 177564
 (1) 001152 177566
 (1) 001154 000
 (1) 001155 002
 (1) 001156 012
 (1) 001157 000
 (1) 001160 000000
 (1) 001162 000000
 (1) 001164 177607 000377
 (1) 001170 077
 (1) 001171 015
 (1) 001172 000012

.SBTTL COMMON TAGS

 *THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
 *USED IN THE PROGRAM.
 .=1100
 \$CMTAG: .WORD 0 ;:START OF COMMON TAGS
 \$TSTNM: .BYTE 0 ;:CONTAINS THE TEST NUMBER
 \$ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG
 \$ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT
 \$LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS
 \$LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS
 \$ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED
 \$ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE
 \$ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST
 \$ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION
 \$GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA
 \$BDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA
 \$GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA
 \$BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA
 .WORD 0 ;:RESERVED--NOT TO BE USED
 \$AUTOB: .BYTE 0 ;:AUTOMATIC MODE INDICATOR
 \$INTAG: .BYTE 0 ;:INTERRUPT MODE INDICATOR
 \$SWR: .WORD DSWR ;:ADDRESS OF SWITCH REGISTER
 \$DISPLAY: .WORD DDISP ;:ADDRESS OF DISPLAY REGISTER
 \$TKS: 177560 ;:TTY KBD STATUS
 \$TKB: 177562 ;:TTY KBD BUFFER
 \$TPS: 177564 ;:TTY PRINTER STATUS REG. ADDRESS
 \$TPB: 177566 ;:TTY PRINTER BUFFER REG. ADDRESS
 \$NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS
 \$FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
 \$FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
 \$TPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
 \$TIMES: 0 ;:MAX. NUMBER OF ITERATIONS
 \$ESCAPE: 0 ;:ESCAPE ON ERROR ADDRESS
 \$BELL: .ASCIZ <207><377><377> ;:CODE FOR BELL
 \$QUES: .ASCII /?/ ;:QUESTION MARK
 \$CRLF: .ASCII <15> ;:CARRIAGE RETURN
 \$LF: .ASCIZ <12> ;:LINE FEED

.SBTTL APT MAILBOX-ETABLE

 .EVEN
 \$MAIL: ;:APT MAILBOX
 \$MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN ;:TEST NUMBER
 \$PASS: .WORD APASS ;:PASS COUNT
 \$DEVCT: .WORD ADEVCT ;:DEVICE COUNT
 \$UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
 \$MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS

(2)	001212	000000	\$MSGLG:	.WORD	AMSGLG	::MESSAGE LENGTH
(2)	001214		\$ETABLF:			::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	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
(2)	001254	170420	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:			
(?)			.MEXIT			

(1) .SBTTL ERROR POINTER TABLE
 (1)
 (1) : *THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 (1) : *THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 (1) : *LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 (1) : *NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 (1) : *NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
 (1)
 (1) : * EM ;:POINTS TO THE ERROR MESSAGE
 (1) : * DH ;:POINTS TO THE DATA HEADER
 (1) : * DT ;:POINTS TO THE DATA
 (1) : * DF ;:POINTS TO THE DATA FORMAT

Index	Item	Code	Pointer	Pointer	Pointer	Pointer	Description
33	001256	030222	032502	032732	EM1,DH2,DT2,DF0	:M8026	NCV11 TIMEOUT
34	001266	030262	032502	032732	EM2,DH2,DT2,DF0	:M8026	COMMAND-STATUS REGISTER ERROR
35	001276	030326	032502	032732	EM3,DH2,DT2,DF0	:M8026	SPECIAL FUNCTION REGISTER ERROR
36	001306	030374	032502	032732	EM4,DH2,DT2,DF0	:M8026	WORD COUNT REGISTER ERROR
37	001316	030434	032502	032732	EM5,DH2,DT2,DF0	:M8026	BUS ADDRESS REGISTER ERROR
38	001326	030475	032502	032732	EM6,DH2,DT2,DF0	:M8026	OFFSET REGISTER ERROR
39	001336	030531	032502	032732	EM7,DH2,DT2,DF0	:M8026	DUAL REGISTER SELECTION ERROR
40	001346	030575	032502	032732	EM10,DH2,DT2,DF0	:M8026-M8036	LOW 16 BIT Z COUNT ERROR
41	001356	030642	032502	032732	EM11,DH2,DT2,DF0	:M8026-M8036	HIGH 16 BIT Z COUNT ERROR
42	001366	030710	032502	032732	EM12,DH2,DT2,DF0	:M8026	Z COUNT STATUS ERROR
43	001376	030743	032502	032732	EM13,DH2,DT2,DF0	:M8026	Z COUNT INTERRUPT ERROR
44	001406	031001	032502	032732	EM14,DH2,DT2,DF0	:M8036	JOYSTICK STATUS ERROR
45	001416	031035	032502	032732	EM15,DH2,DT2,DF0	:M8036	JOYSTICK DATA ERROR
46	001426	031067	032502	032732	EM16,DH2,DT2,DF0	:M8036	DATA INCREMENT ERROR
47	001436	031122	032502	032732	EM17,DH2,DT2,DF0	:M8036	DATA DECREMENT ERROR
48	001446	031155	032502	032732	EM20,DH2,DT2,DF0	:M8026-M8036	MATRIX MODE ADDRESS MAKER DATA ERROR
49	001456	031236	032502	032732	EM21,DH2,DT2,DF0	:M8026	LIST MODE ADDRESS MAKER DATA ERROR
50	001466	031307	032502	032732	EM22,DH2,DT2,DF0	:M8026	LIST MODE TRANSFER BUS ADDRESS ERROR
51	001476	031362	032502	032732	EM23,DH2,DT2,DF0	:M8026	LIST MODE TRANSFER WORD COUNT ERROR
52	001506	031434	032502	032732	EM24,DH2,DT2,DF0	:M8026	LIST MODE TRANSFER OFFSET ERROR
53	001516	031502	032502	032732	EM25,DH2,DT2,DF0	:M8026	TIMEOUT STATUS ERROR

54	001524	033004							
	001526	031535	032502	032732	EM26,DH2,DT2,DF0			;M8026	TIMEOUT INTERRUPT ERROR
	001534	033004							
55	001536	031573	032502	032732	EM27,DH2,DT2,DF0			;M8026	SET 'EVENT' OR 'TIME' DATA ERROR
	001544	033004							
56	001546	031642	032502	032732	EM30,DH2,DT2,DF0			;M8026	CELL INCREMENT DATA ERROR
	001554	033004							
57	001556	031702	032502	032732	EM31,DH2,DT2,DF0			;M8026	CELL OVERFLOW STATUS ERROR
	001564	033004							
58	001566	031743	032502	032732	EM32,DH2,DT2,DF0			;M8026	CELL OVERFLOW INTERRUPT ERROR
	001574	033004							
59	001576	032007	032526	032744	EM33,DH3,DT3,DF0			;M8026	MATRIX MODE ADDRESS MUX ERROR
	001604	033004							
60	001606	032053	032502	032732	EM34,DH2,DT2,DF0			;M8026	'TESTX' FUNCTION ERROR
	001614	033004							
61	001616	032110	032561	032760	EM35,DH4,DT4,DF0			;M8026	DATA ERROR WHEN TRANSFERRING TO EXTENDED MEMORY
	001624	033004							
62	001626	032175	032502	032732	EM36,DH2,DT2,DF0			;M8026	LIST MODE TRANSFER STATUS ERROR
	001634	033004							
63	001636	032243	032526	032744	EM37,DH3,DT3,DF0			;M8026	LIST MODE TRANSFER DATA ERROR
	001644	033004							
64	001646	032307	032502	032732	EM40,DH2,DT2,DF0			;JUMPER-M8026-M7952	'EVENT' OR 'TIME' MARK ERROR
	001654	033004							
65	001656	032367	032467	032776	EM41,DH1,DT5,DF0			;M7952	CLOCK TIMEOUT
	001664	033004							
66	001666	032427	032467	032724	EM42,DH1,DT1,DF0			;M8217	INTERRUPT LEVEL ERROR
	001674	033004							
67									;DL11 BLASTER COMM. ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
68	001676	176510			DLICSR: 176510				;INPUT STATUS REG.
69	001700	176512			DLIBD: 176512				;INPUT DATA
70	001702	176514			DLOCSR: 176514				;OUTPUT STATUS
71	001704	176516			DLODB: 176516				;OUTPUT DATA
72									
73									;VSV01 ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
74	001706	172600			VTCHAR: 172600				;CHAR. STATUS
75	001710	172602			VTYPOS: 172602				
76	001712	172603			VTXPOS: 172603				
77	001714	172604			VTCXY: 172604				
78	001716	172620			VTCSR: 172620				;MAP STATUS
79	001720	172622			VTMAP: 172622				
80	001722	172624			VTPX: 172624				
81	001724	172630			VTINT: 172630				
82									
83									;TESTER ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
84	001726	176416			DACSR: 176416				;KNOWN GOOD D/A STATUS
85	001730	176420			DACO: 176420				
86	001732	176422			DAC1: 176422				
87	001734	176424			DAC2: 176424				
88	001736	176426			DAC3: 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 050032 RTRT: INC AGAN ;SET RESTART FLAG
141 002020 000411 BR RTRTA
142 002022 005237 050024 TESTER: INC WFMODE ;SET FLAG INDICATING OPTION AREA MODE
143 002026 005037 050032 CLR AGAN
144 002032 000404 BR RTRTA
145 002034 005037 050032 BEGIN: CLR AGAN
146 002040 005037 050024 CLR WFMODE ;CLEAR FLAG INDIC. OPTION TEST AREA MODE
147 002044 RTRTA:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 002044 012706 001100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 002050 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 002052 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 002056 001374 BNE -6 ;;LOOP BACK IF NO
(1) 002060 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 002064 012737 034110 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002072 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
(1) 002100 012737 035104 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002106 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
(1) 002114 012737 037270 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002122 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
(1) 002130 012737 035706 000024 MOV #SPURDN,@PURVEC ;;POWER FAILURE VECTOR
(1) 002136 012737 000340 000026 MOV #340,@PURVEC+2 ;;LEVEL 7
(1) 002144 013737 030134 030126 MOV SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 002152 005037 001160 CLR STIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002156 005037 001162 CLR SESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002162 012737 000001 001115 MOVB #1,SEPMAX ;;ALLOW ONE ERROR PER TEST
(1) 002170 012737 002170 001106 MOV #.,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002176 012737 002176 001110 MOV #.,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002204 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002210 012737 002244 000004 MOV #64$,@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) BR 65$ ;;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$:

```

```

149      .SBTTL TYPE PROGRAM NAME
(1)      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002314 005227 177777      INC #1      ;;FIRST TIME?
(1) 002320 001044      BNE 68$      ;;BRANCH IF NO
(1) 002322 022737 030166 000042      CMP #SENDAD,@#42      ;;ACT-11?
(1) 002330 001440      BEQ 68$      ;;BRANCH IF YES
(1) 002332 104401 002400      TYPE ,69$      ;;TYPE ASCIZ STRING
(2)      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002336 005737 000042      TST @#42      ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002342 001012      BNE 70$      ;;BRANCH IF YES
(2) 002344 123727 001214 000001      CMPB $ENV,#1      ;;ARE WE RUNNING UNDER APT?
(2) 002352 001406      BEQ 70$      ;;BRANCH IF YES
(2) 002354 023727 001140 000176      CMP SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
(2) 002362 001005      BNE 71$      ;;BRANCH IF NO
(2) 002364 104406      GTSWR      ;;GET SOFT-SWR SETTINGS
(2) 002366 000403      BR 71$
(2) 002370 112737 000001 001134 70$:      MOV# #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
(2) 002376 71$:
(1) 002376 000415      BR 68$      ;;GET OVER THE ASCIZ
(1)      ;;69$: .ASCIZ <CRLF>#CZNCCB NCV11 DIAGNOSTIC#<CRLF>
(1) 002432 68$:
150      .SBTTL PRIME THE NCV11 ADDRESSES FROM THE DEFAULT VALUES
151 002432 112737 000103 052410      MOV# #'C,RUNIT      ;LOAD 'CONTROL' SECTION BUS TRAP FLAG
152 002440 012737 003212 000004      MOV #BUSTRP,@WERRVEC      ;LOAD BUS TRAP RETURN ADDRESS
153 002446 012700 001746      MOV #CSR,R0      ;GET ADDRESS POINTER
154 002452 013701 001250      MOV $BASE,R1      ;GET DEFAULT ADDRESS
155 002456 010120 1$:      MOV R1,(R0)+      ;LOAD AN VALUE
156 002460 062701 000002      ADD #2,R1      ;BUMP THE VALUE
157 002464 022700 001766      CMP #CSR#B,R0      ;TEST IF DONE
158 002470 001372      BNE 1$      ;BR IF NOT
159 002472 013710 001250      MOV $BASE,(R0)      ;LOAD ODD BYTE
160 002476 005220      INC (R0)+      ;ADDRESSES
161 002500 013710 001250      MOV $BASE,(R0)
162 002504 062720 000011      ADD #11,(R0)+
163 002510 013701 001244      MOV $VECT1,R1      ;GET DEFAULT VECTOR
164 002514 010102      MOV R1,R2      ;COPY R1
165 002516 000302      SWAB R2      ;EXCHANGE BYTES
166 002520 042702 177437      BIC #177437,R2      ;STRIP OFF ALL BUT BR LEVEL BITS
167 002524 010237 002002      MOV R2,BRLEV      ;SAVE FOR USE LATER
168 002530 042701 160000      BIC #160000,R1      ;CLEAR OFF BR LEVEL
169 002534 010120 2$:      MOV R1,(R0)+      ;LOAD VECTOR
170 002536 005721      TST (R1)+      ;BUMP THE VALUE
171 002540 022700 002002      CMP #VECTB1+2,R0      ;TEST IF DONE
172 002544 001373      BNE 2$
173 002546 013737 001254 001740      MOV $CDW1,KWCSR      ;GET CLOCK ADDRESS
174 002554 013737 001740 001742      MOV KWCSR,KWCSR1      ;SET CLOCK HIGH BYTE ADDR.
175 002562 013737 001740 001744      MOV KWCSR,KWPSR      ;SET CLOCK PRESET ADDRESS
176 002570 005237 001742      INC KWCSR1
177 002574 062737 000002 001744      ADD #2,KWPSR
  
```

```
179          .SBTTL  DIAGNOSTIC IDENTIFICATION TYPEOUT
180 002602 005737 050032      TST  AGAN          ;TEST IF RESTART
181 002606 001017              BNE  RBEGO         ;BR IF YES
182 002610 105737 001134      TSTB $AUTOB       ;TEST IF ACT/API
183 002614 001405              BEQ  3$           ;BR IF NOT
184 002616 012737 177777 050022  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 050024      3$: TST  WFMODE     ;TEST IF OPTION CHECKOUT MODE
187 002634 001402              BEQ  4$           ;BR IF YES
188 002636 104401              TYPE
189 002640 046172              LISTO            ;TELL OPERATOR ABOUT OPTION AREA TESTS
190 002642 104401      4$:  TYPE
191 002644 046402              LIST            ;TELL OPERATOR ABOUT THE TEST
192          .SBTTL  KEYBOARD COMMAND DECODER
193 002646 104401 047015      RBEGO: TYPE, LIST1 ;PRINT 'DOT'
194 002652 104411      1$:  RDLIN          ;READ THE RESPONSE
195 002654 013637 052410      MOV  @ (SP)+,RUNIT ;GET THE CHARACTER
196 002660 142737 000040 052410  BICB #40,RUNIT    ;ENSURE UPPER CASE
197 002666 112737 000042 052411  MOVB #'',RUNIT+1  ;FIX ASCII MESSAGE
198 002674 122737 000101 052410  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 050024      2$: TST  WFMODE     ;TEST IF ON THE TESTER
202 002714 001406              BEQ  3$           ;BR IF NOT
203 002716 122737 000102 052410  CMPB #'B',RUNIT   ;DETERMINE IF AN B
204 002724 001002              BNE  3$           ;BR IF NOT
205 002726 000137 042272              JMP  BLAST        ;RUN THE BLASTER SECTION
206 002732 122737 000104 052410  3$: CMPB #'D',RUNIT ;DETERMINE IF AN D
207 002740 001004              BNE  4$           ;BR IF NOT
208 002742 005037 050022              CLR  RUNDIF      ;INDICATE DIFLIN IS NOT TO BE RUN
209 002746 000137 043546              JMP  DIFLIN      ;AND RUN IT
210 002752 122737 000106 052410  4$: CMPB #'F',RUNIT ;DETERMINE IF AN F
211 002760 001005              BNE  5$           ;BR IF NOT
212 002762 012737 177777 050022  MOV  #-1,RUNDIF   ;INDICATE RUN DIFLIN AFTER THE LOGIC TEST
213 002770 000137 003360              JMP  LOGIC        ;AND RUN LOGIC TEST
214 002774 005737 050024      5$: TST  WFMODE     ;TEST IF OPTION CHECKOUT MODE
215 003000 001406              BEQ  6$           ;BR IF NOT
216 003002 122737 000111 052410  CMPB #'I',RUNIT   ;DETERMIN IF AN I
217 003010 001002              BNE  6$           ;BR IF NOT
218 003012 000137 047022              JMP  POTIME       ;RUN IN-HOUSE ADJUSTMENT LOOP
219 003016 122737 000114 052410  6$: CMPB #'L',RUNIT ;DETERMINE IF AN L
220 003024 001004              BNE  7$           ;BR IF NOT
221 003026 005037 050022              CLR  RUNDIF      ;INDICATE NO DIFLIN
222 003032 000137 003360              JMP  LOGIC
223 003036 122737 000110 052410  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 052410  10$: CMPB #'3',RUNIT ;DETERMINE IF AN '?'
227 003060 001002              BNE  11$          ;BR IF NOT
228 003062 000137 002034              JMP  BEGIN        ;START OVER
229 003066 005737 050024      11$: TST  WFMODE     ;TEST IF OPTION CHECKOUT MODE
230 003072 001406              BEQ  12$          ;BR IF NOT
231 003074 122737 000124 052410  CMPB #'T',RUNIT   ;DETERMINE IF AN T
232 003102 001002              BNE  12$          ;BR IF NOT
233 003104 000137 042020              JMP  BTALK        ;KEYBOARD LOOP WITH BLASTER
234 003110 005737 050024      12$: TST  WFMODE     ;TEST IF OPTION CHECKOUT MODE
```

235	003114	001406				BEQ	13\$:BR IF NOT
236	003116	122737	000103	052410		CMPB	#'C,RUNIT	:DETERMINE IF AN C
237	003124	001002				BNE	13\$:BR IF NOT
238	003126	000137	042754			JMP	PBLAST	:RUN M8036 PROM BLASTING CODE
239	003132	122737	000117	052410	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	052410	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	052410	15\$:	CMPB	#'M,RUNIT	:DETERMINE IF AN M
247	003170	001005				BNE	FATFNG	:BR IF NOT
248	003172	012737	000377	050022		MOV	#377,RUNDIF	:INDICATE DIFLIN WITH EXPAND REPORT
249	003200	000137	043546			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      $QUES      ;TYPE QUESTION MARK
254 003210 000616      BR      RBEGO
255
256
257      ;RETURN TO HERE UPON UNEXPECTED BUS TRAP
258 003212 104401 052340 BUSTRP: TYPE, FATAL0      ;REPORT FATAL TRAP TO THE OPERATOR
259 003216 011646      MOV      (SP),-(SP)      ;GET TRAP ADDRESS
260 003220 104402      TYPOC
261 003222 005777 175712 1$: 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 052430 2$: TYPE, RUNITA
265 003236 000137 002014      JMP      RTRT      ;RESTART
266
267      ;ROUTINE TO INPUT NEW TTY ADDRESS FROM THE OPERATOR
268 003242 104401 052520 XTTY: TYPE, WARN0      ;TELL THE OPERATOR THE PULES
269 003246 104412      RDOCT      ;LISTEN TO THE OPERATOR
270 003250 012600      MOV      (SP)+,R0      ;READ THEIR INPUT
271 003252 001754      BEQ      FATFNG      ;BR IF JUST 'CR'
272 003254 005710      TST      (R0)      ;ADDRESS THE DEVICE AND CHECK BUS TRAP
273      ;IF NO BUS TRAP - THEY MUST KNOW WHAT THEY ARE DOING
274 003256 012701 001144      MOV      #STKS,R1      ;GET POINTER
275 003262 010021 1$: MOV      R0,(R1)+      ;LOAD THE VALUE
276 003264 005720      TST      (R0)+      ;BUMP VALUE
277 003266 020127 001154      CMP      R1,#STKS+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 051164 TFSITE: TYPE, PRIM6      ;TELL OPER. ABOUT A017
283 003304 104401 041342      TYPE, FIELDI      ;ASK THE OPERATOR FOR INPUT Z MODE
284 003310 005037 040356      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 040356      MOV      #2,INOUTZ      ;SET MAINT Z CONSTANT
291 003346 000137 037356 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      $PASS          ;TEST IF FIRST PASS
300 003366 001124          BNE      TST1           ;:BR IF NOT
301 003370 005737 050024   TST      WFMODE        ;CHECK IF ON TESTER
302 003374 001402          BEQ      5$            ;BR IF NOT
303 003376 104401 052246   TYPE,   PRIM4         ;TELL OPERATOR CABLE MUST BE CONNECTED
304 003402 005037 036122   5$:     CLR      $KT11   ;INDICATE NO KT11
305 003406 004737 036064   JSR     PC,$SIZE      ;SIZE BASIC MEMORY SIZE
306 003412 162737 001000 036402   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 036402   SUB     #3200,$LSTAD ;LEAVE ROOM FOR XXDP MONITOR
310 003434 013737 036402 003354   1$:     MOV     $LSTAD,ADNOKT ;SAVE BASIC MEMORY STAGE
311 003442 012737 000200 036122   MOV     #BIT7,$KT11  ;WITH M.M.
312 003450 004737 036064   JSR     PC,$SIZE      ;DETERMINE THE AMOUNT OF MEMORY
313 003454 005737 036122   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 052461   TYPE,   MSGMEM        ;TELL OPERATOR THE AMOUNT OF MEMORY
325 003546 013700 036404   MOV     $LSTBK,RO     ;GET LAST GOOD BANK
326 003552 006200          ASR     RO
327 003554 006200          ASR     RO
328 003556 006200          ASR     RO
329 003560 006200          ASR     RO
330 003562 006200          ASR     RO
331 003564 005200          INC     RO             ;CONVERT
332 003566 010046          MOV     RO,-(SP)      ;TYPE OUT DEC. VALUE
333 003570 104405          TYPDS
334 003572 104401 052501          TYPE,   MSGK          ;AND THEN TYPE OUT THE REMAINDER OF MEMORY MESSAGE
335 003576 013737 000004 002004   4$:     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 0C0100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS
351 003650 013737 000004 004016 MOV @WERRVEC,10$ ;SAVE BUS TRAP VECTOR
352 003656 012737 003754 000004 MOV #1$,@WERRVEC ;LOAD BUS TRAP VECTOR
353 003664 005777 176056 TST @CSR ;ADDRESS
354 003670 005777 176054 TST @OFF ;
355 003674 005777 176052 TST @WCR ;THE
356 003700 005777 176050 TST @BAR ;
357 003704 005777 176046 TST @SFR ;NCV11
358 003710 005777 176044 TST @ADM ;
359 003714 005777 176044 TST @BAR1 ;ADDRESSES
360 003720 012737 003762 000004 MOV #2$,@WERRVEC ;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$,@WERRVEC ;RESTORE THE BUS TRAP VECTOR
366 003752 000423 BR TST2 ;;BR AND TEST THE NCV11
367 003754 022626 1$: (MP (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$: (MP (SP)+,(SP)+ ;CLEAN THE STACK
371 003764 104041 ERROR 41 ;BUS TRAP WHEN REFERENCING THE NCV11 CLOCK
372 003766 012737 000001 004020 MOV #1,DEADKW ;INDICATE NCV11 CLOCK IS NOT THERE
373 003774 013737 004016 000004 MOV 10$,@WERRVEC ;RESTORE LOC 4
374 004002 000407 BR TST2 ;;
375 004004 013737 004016 000004 3$: MOV 10$,@WERRVEC ;RESTORE BUS TRAP VECTOR
376 004012 000137 030100 JMP SEOP ;EXIT
377 004016 000000 10$: 0
378 004020 000000 DEADKW: 0 ;NON-ZERO SAYS NO NCV11 CLOCK
  
```



```

381      ;*****
(3)      ;*TEST 2      FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.
(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 1S:  MOV  $TEMP,@CSR      ;LOAD CSR REG.
386 004054 017737 175666 001126      MOV  @CSR,$BDDAT      ;READ CSR
387 004062 013737 002004 001124      MOV  $TEMP,$GDDAT     ;LOAD EXPECTED
388 004070 052737 000200 001124      BIS  #BIT7,$GDDAT     ;FUDGE THE 'READY' BIT
389 004076 023737 001124 001126      CMP  $GDDAT,$BDDAT   ;COMPARE THE VALUES
390 004104 001401
391 004106 104002      BEQ  2$              ;;BR IF SAME
392
393 004110 006237 002004 0C2004 2$:  ASR  $TEMP          ;TRY THE NEXT DATA BIT
394 004114 022737 000001 0C2004      CMP  #1,$TEMP        ;TEST IF NOW BIT 0
395 004122 001351      BNE  1$              ;BR IF NOT
396
397      ;*****
(3)      ;*TEST 3      VERIFY THAT 'INIT' CLEARS THE CSR REGISTER
(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)      ;*TEST 4      VERIFY THAT 'CLEAR ALL' CLEARS THE CSR REGISTER
(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

```

```

416      ;:*****
(3)      ;*TEST 5      VERIFY LOW BYTE OPERATION OF THE 'CSR' REGISTER
(3)      ;:*****
(2) 004244 000004      TST5:  SCOPE
(1) 004246 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
417 004254 012777 003636 175464      MOV      #3636,@CSR      ;LOAD CSR REGISTER
418 004262 012737 003600 001124      MOV      #3600,$GDDAT      ;LOAD EXPECTED VALUE
419 004270 105077 175452      CLRB     @CSR      ;CLEAR LOW BYTE
420 004274 017737 175446 001126      MOV      @CSR,$BDDAT      ;READ STATUS REG.
421 004302 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
422 004310 001401      BEQ      1$      ;;BR IF SAME
423 004312 104002      ERROR    2      ;CLEARING LOW BYTE OF THE CSR CHANGED THE HIGH B
424 004314 012777 003636 175424 1$:  MOV      #3636,@CSR      ;LOAD CSR REGISTER
425 004322 012737 000236 001124      MOV      #236,$GDDAT      ;LOAD EXPECTED VALUE
426 004330 105077 175432      CLRB     @CSRHB      ;CLEAR HIGH BYTE OF THE CSR
427 004334 017737 175406 001126      MOV      @CSR,$BDDAT      ;READ STATUS
428 004342 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
429 004350 001401      BEQ      TST6      ;;BR IS SAME
430 004352 104002      ERROR    2      ;CLEARING HIGH BYTE OF CSR CHANGED THE LOW BYTE
431      ;:*****
(3)      ;*TEST 6      FLOAT A 1 ACROSS 4 BITS OF THE SPECIAL FUNCTION REGISTER
(3)      ;:*****
(2) 004354 000004      TST6:  SCOPE
432 004356 012737 004372 001110      MOV      #1$,$LPERR      ;LOAD LOOP ADDRESS
433 004364 012737 000020 002004      MOV      #BIT4,$STEMP      ;LOAD INITIAL REG. VALUE
434
435 004372 013777 002004 175356 1$:  MOV      $STEMP,@SFR      ;LOAD SFR REG.
436 004400 017737 175352 001126      MOV      @SFR,$BDDAT      ;READ SFR
437 004406 013737 002004 001124      MOV      $STEMP,$GDDAT      ;LOAD EXPECTED
438 004414 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE THE VALUES
439 004422 001401      BEQ      2$      ;;BR IF SAME
440 004424 104003      ERROR    3      ;UNEXPECTED VALUE IN THE SFR REGISTER
441 004426 006237 002004 2$:  ASR      $STEMP      ;TRY THE NEXT DATA BIT
442 004432 022737 000001 002004      CMP      #1,$STEMP      ;TEST IF NOW BIT 0
443 004440 001354      BNE      1$      ;BR IF NOT
444
445      ;:*****
(3)      ;*TEST 7      VERIFY THAT CLEARING HIGH BYTE OF SFR DOES NOT CLEAR LOW BYTE
(3)      ;:*****
(2) 004442 000004      TST7:  SCOPE
(1) 004444 012737 000010 001160      MOV      #10,$TIMES      ;;DO 10 ITERATIONS
446 004452 012777 000014 175276      MOV      #14,@SFR      ;LOAD THE S.F. REGISTER
447 004460 012737 000014 001124      MOV      #14,$GDDAT      ;LOAD THE EXPECTED VALUE
448 004466 105077 175276      CLRB     @SFRHB      ;CLEAR HIGH BYTE OF S.F. REG.
449 004472 017737 175260 001126      MOV      @SFR,$BDDAT      ;READ THE REGISTER
450 004500 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE THE VALUES
451 004506 001401      BEQ      TST10      ;;BR IF SAME
452 004510 104003      ERROR    3      ;CLEARING HIGH BYTE OF CSR REG. CHANGED THE LOW
453
    
```

```

455      ;*****
(3)      ;*TEST 10      VERIFY THAT 'INIT' CLEARS THE SFR REGISTER
(3)      ;*****
(2) 004512 000004      TST10: SCOPE
(1) 004514 012737 000010 001160      MOV #10,$TIMES      ;;DO 10 ITERATIONS
456 004522 012777 000016 175226      MOV #16,@SFR      ;LOAD SFR REG.
457 004530 000005      RESET      ;INITILIZE THE REGISTER
458 004532 005037 001124      CLR $GDDAT      ;LOAD EXPECTED
459 004536 017737 175214 001126      MOV @SFR,$BDDAT      ;READ SFR REG.
460 004544 001401      BEQ TST11      ;;BR IF EQUAL
461 004546 104003      ERROR 3      ;'BUS INIT' FAILED TO CLEAR SFR REG.
462
463

```

```

(3)      ;*****
(3)      ;*TEST 11      VERIFY THAT 'CLEAR ALL' CLEARS THE SFR REGISTER
(3)      ;*****
(2) 004550 000004      TST11: SCOPE
(1) 004552 012737 000010 001160      MOV #10,$TIMES      ;;DO 10 ITERATIONS
464 004560 012777 000016 175170      MOV #16,@SFR      ;LOAD SFR REG.
465 004566 052777 004000 175162      BIS #CLRALL,@SFR      ;GENERATE 'CLR ALL L'
466 004574 005037 001124      CLR $GDDAT      ;LOAD EXPECTED VALUE
467 004600 017737 175152 001126      MOV @SFR,$BDDAT      ;READ THE SFR REG.
468 004606 001401      BEQ TST12      ;;BR IF SAME
469 004610 104003      ERROR 3      ;'CLR ALL L' FAILED TO CLEAR SFR REG.
470

```

```
472      ;:*****  
(3)      ;*TEST 12      FLOAT A 1 ACROSS THE WORD COUNT REGISTER  
(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)      ;*TEST 13      FLOAT A 1 ACROSS THE BUS ADDRESS REGISTER  
(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)      ;*TEST 14      FLOAT A 1 ACROSS THE OFFSET REGISTER  
(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  
(1) 005056 012737 000010 001160  
513      :LOAD DIFFERENT NUMBERS INTO THE WCR, BAR AND OFF REGISTERS  
514 005064 012777 004000 174664      MOV #10,$TIMES      ;;DO 10 ITERATIONS  
515 005072 012777 011111 174650      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE  
516 005100 012777 022222 174644      MOV #11111,@OFF      ;LOAD OFFSET REGISTER  
517 005106 012777 033333 174640      MOV #22222,@WCR      ;LOAD W.C. REGISTER  
518 005114 012777 000036 174634      MOV #33333,@BAR      ;LOAD B.A. REGISTER  
519 005122 012777 007636 174616      MOV #36,@SFR      ;LOAD SPECIAL FUNCTION REGISTER  
520      MOV #7636,@CSR      ;LOAD COMMAND/STATUS REGISTER  
521 005130 012737 011111 001124      ;NOW READ EACH REGISTER AND CHECK THE VALUE  
522 005136 017737 174606 001126      MOV #11111,$GDDAT      ;LOAD EXPECTED VALUE  
523 005144 023737 001124 001126      MOV @OFF,$BDDAT      ;READ A OFFSET REG.  
524 005152 001401      CMP $GDDAT,$BDDAT      ;COMPARE  
525 005154 104007      BEQ 1$      ;;BR IF SAME  
526 005156 012737 022222 001124 1$:      MOV #22222,$GDDAT      ;DUAL ADDRESS ERROR  
527 005164 017737 174562 001126      MOV @WCR,$BDDAT      ;LOAD EXPECTED VALUE  
528 005172 023737 001124 001126      CMP $GDDAT,$BDDAT      ;READ W.C. REG.  
529 005200 001401      BEQ 2$      ;COMPARE  
530 005202 104007      ERROR 7      ;;BR IF SAME  
531 005204 012737 033333 001124 2$:      MOV #33333,$GDDAT      ;DUAL ADDRESS ERROR  
532 005212 017737 174536 001126      MOV @BAR,$BDDAT      ;LOAD EXPECTED VLAUE  
533 005220 023737 001124 001126      CMP $GDDAT,$BDDAT      ;READ B.A. REG.  
534 005226 001401      BEQ 3$      ;COMPARE  
535 005230 104007      ERROR 7      ;;BR IF SAME  
536 005232 012737 000036 001124 3$:      MOV #36,$GDDAT      ;DUAL ADDRESS ERROR  
537 005240 017737 174512 001126      MOV @SFR,$BDDAT      ;LOAD EXPECTED VALUE  
538 005246 023737 001124 001126      CMP $GDDAT,$BDDAT      ;READ SPECIAL FUNCTION REG  
539 005254 001401      BEQ 4$      ;COMPARE  
540 005256 104007      ERROR 7      ;;BR IF SAME  
541 005260 012737 007636 001124 4$:      MOV #7636,$GDDAT      ;DUAL ADDRESS ERROR  
542 005266 017737 174454 001126      MOV @CSR,$BDDAT      ;LOAD EXPECTED VALUE  
543 005274 023737 001124 001126      CMP $GDDAT,$BDDAT      ;READ COMMAND/STATUS REGISTER  
544 005302 001401      BEQ 5$      ;COMPARE  
545 005304 104007      ERROR 7      ;;BR IF SAME  
546 005306 012777 004000 174442 5$:      MOV #CLRALL,@SFR      ;DUAL ADDRESS ERROR  
      ;CLEAR THE DEVICE
```

```

548
(3)
(3)
(2) 005314 000004
(1) 005316 012737 000010 001160
549 005324 005037 001124
550 005330 012777 004000 174420
551 005336 012777 000003 174404
552 005344 012777 004000 174404
553 005352 017737 174372 001126
554 005360 001401
555 005362 104006
556
557
(3)
(3)
(2) 005364 000004
558 005366 012777 004000 174362
559 005374 052777 000010 174354
560 005402 000240
561 005404 000240
562 005406 000240
563 005410 052777 000022 174330
564
565 005416 012737 000022 001124
566 005424 052777 000001 174314
567 005432 017737 174310 001126
568 005440 023737 001124 001126
569 005446 001404
570 005450 052777 000400 174300
571 005456 104002
572
573 005460 052777 000400 174270
574 005466 012737 000222 001124
575 005474 017737 174246 001126
576 005502 023737 001124 001126
577 005510 001401
578 005512 104002
579
580
581 005514 012737 000200 001124
582 005522 052777 000001 174216
583 005530 052777 004000 174220
584 005536 017737 174204 001126
585 005544 023737 001124 001126
586 005552 001401
587 005554 104001

```

```

:*****
:*TEST 16 VERIFY 'CLR ALL' CLEARS THE EXTENDED OFFSET BITS
:*****
TST16: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
CLR $GDDAT ;CLEAR EXPECTED VALUE
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #3,@OFF ;LOAD EXTENDED OFFSET REGISTER
MOV #CLRALL,@SFR ;CLEAR THE EXTENDED OFFSET REG.'S
MOV @OFF,$BDDAT ;READ EXTENDED OFFSET REG.
BEQ TST17 ;;BR IF CLEARED
ERROR 6 ;CLEAR ALL FAILED TO CLEAR EXTENDED OFFSET REGIS

:*****
:*TEST 17 TEST THE 'ACTIVE' FLOP CAN SET AND CLEAR
:*****
TST17: SCOPE
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
BIS #TSTDMA,@SFR ;SET THE 'TEST DMA' TO PREVENT DATA TRANSFERS
NOP
NOP
NOP
BIS #BIT4!BIT1,@CSR ;SET 'MATRIX MODE' AND 'BYTE' MODE
:POKE THE 'ACTIVE' FLOP AND VERIFY 'INTERFACE IDLE' GOES LOW
MOV #BIT4!BIT1,$GDDAT ;LOAD EXPECTED VALUE
BIS #BIT0,@CSR ;SET 'ACTIVE' TO A 1
MOV @CSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 1$ ;;BR IF EXPECTED
BIS #ENDDMA,@SFR ;STOP DMA IF POSSIBLE
ERROR 2 ;'ACTIVE' FLOP FAILED TO SET
:POKE THE 'END DMA' SIGNAL AND VERIFY 'ACTIVE' CLEARS
1$: BIS #ENDDMA,@SFR ;SEND 'END DMA' SIGNAL
MOV #BIT7!BIT4!BIT1,$GDDAT ;SET 'INTERFACE IDLE' INTO EXPECTED
MOV @CSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;;BR IF SAME
ERROR 2 ;'END DMA' AGAIN FAILED TO CLEAR 'ACTIVE' FLOP
:POKE 'ENABLE DMA' AND THEN ISSUE 'CLR ALL' SIGNAL TO
ENSURE THE 'ACTIVE' FLOP CLEARS
2$: MOV #BIT7,$GDDAT ;LOAD EXPECTED
BIS #BIT0,@CSR ;POKE 'ENABLE DMA'
BIS #CLRALL,@SFR ;GENERATE 'CLR ALL'
MOV @CSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST20 ;;BR IF SAME
ERROR 1 ;'CLR ALL' FAILED TO CLEAR 'ACTIVE' FLOPS

```

```

592
(3)
(3)
(2) 005556 000004
(1) 005560 012737 000100 001160
593 005566 012777 004000 174162
594 005574 012777 000010 174154
595 005602 012777 000000 174142
596 005610 012777 000000 174136
597 005616 012777 000022 174122
598 005624 052777 000001 174114
599 005632 052777 000002 174116
(1) 005640 042777 000002 174110
600 005646 012737 000001 001124
601 005654 017737 174074 001126
602 005662 001002
603 005664 104010
604 005666 000420
605 005670 017737 174056 001126
606 005676 001401
607 005700 104011
608 005702 012737 000020 001124
609 005710 017737 174032 001126
610 005716 032737 040000 001126
611 005724 001401
612 005726 104012

```

```

*****
*TEST 20 VERIFY Z INPUTS CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN MATRI
*****
TST20: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #0,@WCR ;LOAD W.C. REG
MOV #0,@BAR ;LOAD B.A. REG
MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;GO 'ACTIVE'
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV #1,$GDDAT ;LOAD EXPECTED
MOV @BAR,$BDDAT ;READ LOW 16 BITS
BNE 1$ ;;BR IF NON-ZERO
ERROR 10 ;Z INPUT FAILED TO CAUSE THE LOW BYTE OF 32 BIT
BR TST21 ;;
1$: MOV @WCR,$BDDAT ;READ HIGH 16 BITS
BEQ 2$ ;;BR IF CLEARED
ERROR 11 ;HIGH 16 BIS CHANGED IN ERROR
2$: MOV #BIT4,$GDDAT ;LOAD EXPECTED
MOV @CSR,$BDDAT ;READ STATUS
BIT #BIT14,$BDDAT ;TEST IF 'CELL OVERFLOW'
BEQ TST21 ;;BR IF CORRECT
ERROR 12 ;'CELL OVERFLOW' FLOP SET IN ERROR

```

```

613
(3)
(3)
(2) 005730 000004
(1) 005732 012737 000100 001160
614 005740 012777 004000 174010
615 005746 012777 000010 174002
616 005754 012777 000000 173770
617 005762 012777 000376 173764
618 005770 012777 000022 173750
619 005776 052777 000001 173742
620 006004 052777 000002 173744
(1) 006012 042777 000002 173736
621 006020 012737 000400 001124
622 006026 017737 173722 001126
623 006034 105737 001127
624 006040 001002
625 006042 104010
626 006044 000423
627 006046 012737 000000 001124
628 006054 017737 173672 001126
629 006062 001401
630 006064 104011
631 006066 012737 000020 001124
632 006074 017737 173646 001126
633 006102 032737 040000 001126
634 006110 001401
635 006112 104012

```

```

*****
*TEST 21 VERIFY Z INPUTS CAUSE THE LOW 16 BITS OF 32 BIT COUNTER TO CHANGE
*****
TST21: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #0,@WCR ;LOAD W.C. REG
MOV #376,@BAR ;LOAD B.A. REG
MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;GO 'ACTIVE'
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV #BIT8,$GDDAT ;LOAD EXPECTED
MOV @BAR,$BDDAT ;READ LOW 16 BITS
TSTB $BDDAT+1 ;TEST HIGH BYTE
BNE 1$ ;BR IF NON-ZERO
ERROR 10 ;Z INPUT FAILED TO CAUSE THE HIGH BYTE OF THE LO
BR TST22 ;;
1$: MOV #0,$GDDAT ;LOAD EXPECTED
MOV @WCR,$BDDAT ;READ HIGH 16 BITS
BEQ 2$ ;;BR IF CLEARED
ERROR 11 ;HIGH 16 BITS CHANGED IN ERROR
2$: MOV #BIT4,$GDDAT ;LOAD EXPECTED
MOV @CSR,$BDDAT ;READ STATUS
BIT #BIT14,$BDDAT ;TEST IF 'CELL OVERFLOW'
BEQ TST22 ;;BR IF CORRECT
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

```

```

*****
*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

```

```

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 23 VERIFY Z INPUTS CAUSE THE HIGH 8 BITS OF THE 32 BIT COUNTER TO CHANGE
*****
TST23: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #377,@WCR ;LOAD W.C. REG
MOV #-2,@BAR ;LOAD B.A. REG
MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;GO 'ACTIVE'
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV #BIT8,$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  
BIT #BIT14,@CSR ;TEST FOR Z/WC OVERFLOW  
BNE 2$ ;BR IF SET  
DEC R1 ;DELAY  
BNE 1$  
DEC R0 ;DFLAY  
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
```

C 4

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 TST27: SCOPE  
(1) 007056 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS  
746 007064 012777 004000 172664 MOV #CLRALL,@SFR ;CLEAR DEVICE  
747 007072 012777 000010 172656 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP  
748 007100 012777 177777 172644 MOV #-1,@WCR ;LOAD Z COUNTER  
749 007106 012777 177776 172640 MOV #-2,@BAR ;LOAD Z COUNTER  
750 007114 012777 000022 172624 MOV #BIT4!BIT1,@CSR ;SET MATRIX MODE  
751 007122 052777 000001 172616 BIS #BIT0,@CSR ;SET NCV11 ACTIVE  
752 ;NOW ENABLE 'TEST Z' INPUTS  
753 007130 052777 000002 172620 BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
(1) 007136 042777 000002 172612 BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
754 007144 013700 002006 MOV CPUDLO,R0 ;LOAD GROSS TIMER  
755 007150 005001 CLR R1  
756 007152 012737 040020 001124 MOV #BIT14!BIT4,$GDDAT ;LOAD EXPECTED STATUS  
757  
758 007160 032777 040000 172560 1$: BIT #BIT14,@CSR ;TEST FOR Z/WC OVERFLOW  
759 007166 001014 BNE 2$ ;BR IF SET  
760 007170 005301 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\$:
809 007454 017737 172270 001126
810 007462 001401
811 007464 104012
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,$TIMES ;;DO 40 ITERATIONS  
818 007476 012777 004000 172252 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
819 007504 052777 000001 172244 BIS #REDJOY,@SFR ;REQUEST JOYSTICK DATA  
820 007512 013700 002010 MOV CPUDL1,R0 ;LOAD DELAY COUNTER  
821 007516 105777 172234 1S: TSTB @SFR ;WAIT FOR JOYSTICK READY  
822 007522 100411 BMI 2$ ;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 1S: 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) 007740 000004
(1) 007742 012737 000010 001160
875 007750 012777 004000 172000
(1) 007756 012777 000014 171772
(1) 007764 012777 000000 171754
(1) 007772 052777 000001 171756
(1) 010000 105777 171752
(1) 010004 100375
(1) 010006 017737 171750 001126
(1) 010014 012737 000000 001124
(1) 010022 023737 001124 001126
876 010030 001401
877 010032 104015
878
879
(3)
(3)
(2) 010034 000004
(1) 010036 012737 000010 001160
880 010044 012777 004000 171704
(1) 010052 012777 000014 171676
(1) 010060 012777 002000 171660
(1) 010066 052777 000001 171662
(1) 010074 105777 171656
(1) 010100 100375
(1) 010102 017737 171654 001126
(1) 010110 012737 124250 001124
(1) 010116 023737 001124 001126
881 010124 001401
882 010126 104015
883
884
(3)
(3)
(2) 010130 000004
(1) 010132 012737 000010 001160
885 010140 012777 004000 171610
(1) 010146 012777 000014 171602
(1) 010154 012777 004000 171564
(1) 010162 052777 000001 171566
(1) 010170 105777 171562
(1) 010174 100375
(1) 010176 017737 171560 001126
(1) 010204 012737 050120 001124
(1) 010212 023737 001124 001126
886 010220 001401
887 010222 104015

```
*****  
*TEST 33 JOYSTICK DATA PATH = GAIN = 0 ZB FNABLE 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)
(3)
(2) 010224 000004
(1) 010226 012737 000010 001160
890 010234 012777 004000 171514
(1) 010242 012777 000014 171506
(1) 010250 012777 000023 171470
(1) 010256 052777 000001 171472
(1) 010264 105777 171466
(1) 010270 100375
(1) 010272 017737 171464 001126
(1) 010300 012737 000401 001124
(1) 010306 023737 001124 001126
891 010314 001401
892 010316 104015
893
894
(3)
(3)
(2) 010320 000004
(1) 010322 012737 000010 001160
895 010330 012777 004000 171420
(1) 010336 012777 000014 171412
(1) 010344 012777 000025 171374
(1) 010352 052777 000001 171376
(1) 010360 105777 171372
(1) 010364 100375
(1) 010366 017737 171370 001126
(1) 010374 012737 001002 001124
(1) 010402 023737 001124 001126
896 010410 001401
897 010412 104015
898
899
(3)
(3)
(2) 010414 000004
(1) 010416 012737 000010 001160
900 010424 012777 004000 171324
(1) 010432 012777 000014 171316
(1) 010440 012777 000031 171300
(1) 010446 052777 000001 171302
(1) 010454 105777 171276
(1) 010460 100375
(1) 010462 017737 171274 001126
(1) 010470 012737 002004 001124
(1) 010476 023737 001124 001126
901 010504 001401
902 010506 104015

```
*****  
*TEST 36 JOYSTICK DATA PATH RES. = 001  
*****  
TST36: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #23,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #401,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST37 ;;BR IF SAME  
ERROR 15 ;JOYSTICK DATA PATH ERROR RES. - 001
```

```
*****  
*TEST 37 JOYSTICK DATA PATH RES. = 010  
*****  
TST37: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #25,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #1002,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST40 ;;BR IF SAME  
ERROR 15 ;JOYSTICK DATA PATH ERROR RES. = 010
```

```
*****  
*TEST 40 JOYSTICK DATA PATH RES. = 100  
*****  
TST40: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #31,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #2004,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST41 ;;BR IF SAME  
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)
(2) 010774 000004
(1) 010775 012737 000010 001160
925 011004 012777 004000 170744
(1) 011012 012777 000014 170736
(1) 011020 012777 001023 170720
(1) 011026 052777 000001 170722
(1) 011034 105777 170716
(1) 011040 100375
(1) 011042 017737 170714 001126
(1) 011050 012737 000000 001124
(1) 011056 023737 001124 001126
926 011064 001401
927 011066 104017

```
*****  
*TEST 44 VERIFY THE DATA DECREMENT FUNCTION  
*****  
TST44: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #1023,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #0,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST45 ;;BR IF SAME  
ERROR 17 ;IF DATA WAS 401, THE DATA DECREMENT FUNCTION FAILED  
;OTHERWISE DECREMENTED DATA ERROR
```

928
929
930
(3)
(3)
(2) 011070 000004
(1) 011072 012737 000010 001160
931 011100 012777 004000 170650
(1) 011106 012777 000014 170642
(1) 011114 012777 007037 170624
(1) 011122 052777 000001 170626
(1) 011130 105777 170622
(1) 011134 100375
(1) 011136 017737 170620 001126
(1) 011144 012737 177376 001124
(1) 011152 023737 001124 001126
932 011160 001401
933 011162 104017

```
*****  
*TEST 45 VERIFY THE DATA DECREMENT BORROW  
*****  
TST45: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #7037,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #177376,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST46 ;;BR IF SAME  
ERROR 17 ;DECREMENTED DATA ERROR
```

934
935
(3)
(3)
(2) 011164 000004
(1) 011166 012737 000010 001160
936 011174 012777 004000 170554
(1) 011202 012777 000014 170546
(1) 011210 012777 001021 170530
(1) 011216 052777 000001 170532
(1) 011224 105777 170526
(1) 011230 100375
(1) 011232 017737 170524 001126
(1) 011240 012737 000000 001124
(1) 011246 023737 001124 001126
937 011254 001401
938 011256 104017
939

```
*****  
*TEST 46 VERIFY THE DATA DECREMENT FUNCTION IS INHIBITED  
*****  
TST46: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #1021,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #0,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST47 ;;BR IF SAME  
ERROR 17 ;INHIBIT DECREMENT FUNCTION ERROR  
;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)

```

*****
*TEST 47 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN - 0 ZB ENABLE - 0
*****
TST47: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #36,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$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
    
```

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)

```

*****
*TEST 50 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 1 ZB ENABLE 0
*****
TST50: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #2036,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$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
    
```

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 51 TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN 0 ZB ENABLE 1
*****
TST51: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #4036,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BIT0,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$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 170111
(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)

```
*****  
*TEST 55 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN 0 ZB ENABLE = 0  
*****  
TST55: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #32,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

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)

```
*****  
*TEST 56 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN - 1 ZB ENABLE = 0  
*****  
TST56: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #2032,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

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 57 TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN - 0 ZB ENABLE - 1  
*****  
TST57: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #4032,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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

```
*****  
*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

```
*****  
*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

```
*****  
*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 001124
(3) 013314 001401
(1) 013316 104020

```
*****  
*TEST 66 TEST ADDRESS MAKER - MATRIX MODE - RES - 2 GAIN = 0 ZB ENABLE - 0  
*****  
TST66: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #24,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

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 001124
(3) 013416 001401
(1) 013420 104020

```
*****  
*TEST 67 TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 1 ZB ENABLE 0  
*****  
TST67: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #2024,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

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 001124
(3) 013520 001401
(1) 013522 104020

```
*****  
*TEST 70 TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE 1  
*****  
TST70: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #4024,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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)

```
*****  
*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
```

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)

```
*****  
*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
```

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 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,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST 'DMA AND CONTROL'  
MOV #0,@CSR ;ENSURE LIST MODE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST 'DMA AND CONTROL'  
MOV #4000,@CSR ;ENSURE LIST MODE AND ZB ENABLED  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST 'DMA AND CONTROL'  
MOV #2000,@CSR ;SET GAIN FLOP  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;ENABLE THE NCV11  
BIC #TESTZ,@SFR ;DISABLE THE 'TEST Z' PULSES  
MOV @ADM,$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 001126 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,BUF0 ;PRIME TARGET BUFFER
MOV #11110,@OFF ;LOAD THE OFFSET VALUE WITH A NUMBER
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUF0,@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
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 #BUF0+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 BUF0,\$BDDAT ;GET BUFFER DATA
MOV #3407,\$GDDAT ;LOAD EXPECTED
MOV #BUF0,\$BADDR ;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,$TIMES ;;DO 40 ITERATIONS
1078 014654 012777 004000 165074 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1079 014662 012700 060000 MOV #BUF0,R0 ;LOAD BUFFER POINTER
1080 014666 012720 125252 1$ 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 #BUF0,@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 #TEST2,@SFR ;ENABLE 'TEST Z' PULSES
(1) 014744 042777 000002 165004 BIC #TEST2,@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 A 1 WORD TRANSFER
1109
1110 015070 017737 164656 001126 5$: MOV @WCR,$BDDAT ;READ W.C. REGISTER
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 #BUF0,$BDDADR ;LOAD STARTING ADDRESS
1117 015132 017737 163764 001126 7$: MOV @BDDADR,$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$: ADD #2,$BDDADR ;UPDATE POINTER
1122 015160 022737 062000 001122 CMP #BUF0+1024.,$BDDADR ;TEST IF END OF BUFFER
1123 015166 001361 BNE 7$ ;;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,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #-1,@WCR ;LOAD W.C. REGISTER  
MOV #3,@OFF ;LOAD EXTENDED ADDRESS BITS  
MOV #160000,@BAR ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDR  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA  
MOV #16,@CSR ;LOAD RESOLUTION TO VERIFY 'SFR INIT' CLEARS  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV #BIT15!BIT14!BIT7,$GDDAT ;LOAD EXPECTED  
MOV @CSR,$BDDAT ;READ STATUS REG.  
BMI 1$ ;BR IF 'TIMEOUT' FLOP IS SET  
ERROR 25 ;'TIMEOUT' FLOP FAILED TO SET  
BR TST102 ;;  
1$: CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ 2$ ;;BR IF SAME  
ERROR 25 ;'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 25 ;'CLR ALL' FAILED TO CLEAR TIMEOUT FLOP  
*****  
*TEST 102 VERIFY 'TIMEOUT' FLOP SETS AND 'CLR TIMEOUT' CLEARS IT  
*****  
TST102: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #-1,@WCR ;LOAD W.C. REGISTER  
MOV #3,@OFF ;LOAD EXTENDED ADDRESS BITS  
MOV #160000,@BAR ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDR  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
MOV #BIT15 BIT14.BIT7,$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 036404 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

```
*****  
*TEST 104 VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 16  
*****  
TST104: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CMP $LSTBK,#2140 ;TEST IF ENOUGH MEMORY  
BLO TST105 ;;BR IF NOT ENOUGH MEMORY  
MOV #-1,@WCR ;LOAD 1 WORD XFR  
MOV #177776,@BAR ;LOAD LAST ADDRESS  
CLR @OFF ;ENSURE CLEARED EXTENDED ADDRESS BITS  
MOV #TSTCON!TSTDMA,@SFR ;ENABLE CONTROLER  
MOV #16,@CSR ;ENABLE THE MODE  
BIS #BIT0,@CSR ;ENABLE THE NCV11  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
NOP  
MOV #1,$GDDAT ;LOAD EXPECTED VALUE  
MOV @OFF,$BDDAT ;READ ACUTAL VALUE  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST105 ;;BR IF SAME  
ERROR 5 ;EXTENDED ADDRESS BIT 16 FAILED TO SET
```

1237
1238
(3)
(3)
(2) 016150 000004
(1) 016152 012737 000040 001160
1239 016160 012777 004000 163570
1240 016166 023727 036404 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 105 VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 17  
*****  
TST105: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CMP $LSTBK,#6140 ;TEST MEMORY SPACE >100K  
BLO TST106 ;;BR IF NOT ENOUGH MEMORY  
MOV #-1,@WCR ;LOAD 1 WORD XFR  
MOV #177776,@BAR ;LOAD LAST ADDRESS  
MOV #1,@OFF ;LOAD EXTENDED ADDRESS BIT  
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL  
MOV #16,@CSR ;ENABLE THE MODE  
BIS #BIT0,@CSR ;ENAVLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TFSTZ,@SFR ;DISABLE 'TEST Z' PULSES  
NOP  
NOP  
NOP  
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
NOP  
NOP  
NOP  
MOV #2,$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) *****
1263 (3) *TEST 106 VERIFY 'SET EVENT' DATA GENERATES A 177777 DATA WORD
1264 (2) 016326 000004 TST106: SCOPE
1265 (1) 016330 012737 000040 001160 MOV #40,$TIMES ;;DO 40 ITERATIONS
1266 016336 012777 004000 163412 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1267 016344 012777 177777 163400 MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
1268 016352 012777 060000 163374 MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
1269 016360 012777 000014 163370 MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
1270 016366 012737 007070 060000 MOV #7070,BUFO ;PRESET DATA
1271 016374 052777 000001 163344 BIS #BIT0,@CSR ;ENABLE DEVICE
1272 016402 000240 NOP
1273 016404 000240 NOP
1274 016406 000240 NOP
1275 016410 052777 002000 163340 BIS #BIT10,@SFR ;SET 'EVENT' FLOP
1276 016416 052777 010000 163332 BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
1277 016424 000240 NOP
1278 016426 000240 NOP
1279 016432 013737 060000 001126 MOV BUFO,$BDDAT ;READ DATA
1280 016440 012737 177777 001124 MOV #-1,$GDDAT ;LOAD EXPECTED
1281 016446 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
1282 016454 001401 BEQ TST107 ;;BR IF SAME
1283 016456 104027 ERROR 27 ;INCORRECT DATA VALUE FOR 'EVENT' MARK
;AFTER A 1 WORD TRANSFER
    
```

```

1283 (3) *****
1284 (3) *TEST 107 VERIFY 'SET TIME' DATA GENERATES A 000000 DATA WORD
1285 (2) 016460 000004 TST107: SCOPE
1286 (1) 016462 012737 000040 001160 MOV #40,$TIMES ;;DO 40 ITERATIONS
1287 016470 012777 004000 163260 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
1288 016476 012777 177777 163246 MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
1289 016504 012777 060000 163242 MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
1290 016512 012777 000014 163236 MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
1291 016520 012737 007070 060000 MOV #7070,BUFO ;PRESET THE DATA
1292 016526 052777 000001 163212 BIS #BIT0,@CSR ;ENABLE DEVICE
1293 016534 000240 NOP
1294 016536 000240 NOP
1295 016540 000240 NOP
1296 016542 052777 001000 163206 BIS #BIT9,@SFR ;SET 'TIME' FLOP
1297 016550 052777 010000 163200 BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
1298 016556 000240 NOP
1299 016560 000240 NOP
1300 016562 000240 NOP
1301 016564 013737 060000 001126 MOV BUFO,$BDDAT ;READ DATA
1302 016572 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
1303 016600 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUES
1304 016606 001401 BEQ TST110 ;;BR IF SAME
1305 016610 104027 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
1339 017054 100375
1340 017056 052777 010000 162672
1341 017064 000240
1342 017066 000240
1343 017070 013737 060000 001126
1344 017076 012737 000000 001124
1345 017104 023737 001124 001126
1346 017112 001401
1347 017114 104040
1348

```

```

:*****
:*TEST 110 VERIFY 'CLOCK ST1' GENERATES A EVENT (177777) DATA WORD
:*****
TST110: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
TST DEADKW ;TEST IF NCV11 CLOCK IS PRESENT
BNE TST111 ;;BR IF NOT
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUF0,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUF0 ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE THE DEVICE
NOP
NOP
BIS #BIT8,@KWCSR ;GENERATE CLOCK ST1 TO SET 'EVENT' FLOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUF0,$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,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
TST DEADKW ;TEST IF NCV11 CLOCK IS PRESENT
BNE TST112 ;;BR IF NOT
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUF0,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUF0 ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE THE DEVICE
CLR @KWCSR ;CLEAR STATUS
MOV #-2,@KWPSR ;LOAD COUNTER PRESET
MOV #11,@KWCSR ;ENABLE 1 MHZ. RATE AND CLOCK GO
1$: TSTB @KWCSR ;WAIT FOR CLOCK
BPL 1$
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV BUF0,$BDDAT ;READ DATA
MOV #0,$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,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #-1,@WCR ;SET UP 1 WORD TRANSFER
MOV #BUFO,@BAR ;LOAD BUS ADDRESS FOR RESULT
MOV #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV #7070,BUFO ;PRESET THE DATA
BIS #BIT0,@CSR ;ENABLE DEVICE
NOP
NOP
NOP
BIS #BIT10!BIT9,@SFR ;SET 'TIME AND EVENT' FLOPS
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
NOP
MOV BUFO,$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
(1) 017410 012737 000040 001160
1394      TST114: SCOPE
1395      MOV     #40,$TIMES      ;;DO 40 ITERATIONS
1396      ;CHECK FOR A 'CARRY' INTO EACH BIT
1397      IE:     000002-000001
1398      :       000004-000003
1399      :       000010-000007
1400      :       000020-000017
1401      :       000040-000037
1402      :       000100-000077
1403      :       000200-000177
1404      :       ETC
1405      MOV     #1,$SLPERR      ;LOAD RETURN ADDRESS IF ERROR
1406      MOV     #2,$TEMP       ;LOAD INITIAL VALUE
1407      1$:    MOV     #CLRALL,@SFR      ;CLEAR THE DEVICE
1408      MOV     #BUFO,@OFF      ;LOAD INITIAL OFFSET REGISTER
1409      MOV     #BIT4!BIT2,@CSR   ;ENABLE WORD MATRIX MODE
1410      MOV     #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
1411      BIS     #BIT0,@CSR       ;ENABLE THE DEVICE
1412      MOV     $TEMP,BUFO       ;LOAD PRESET VALUE
1413      DEC     BUFO             ;TO EXPECTED -1
1414      BIS     #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
1415      BIC     #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
1416      BIS     #BIT12,@SFR     ;ALLOW 1 TRANSFER
1417      MOV     $TEMP,$GDDAT     ;LOAD TYPEOUT EXPECTED
1418      MOV     BUFO,$BDDAT      ;READ THE BUFO LOCATION
1419      CMP     $GDDAT,$BDDAT    ;COMPARE VALUES
1420      BEQ     2$              ;;BR IF SAME
1421      ERROR   30              ;INCORRECT DATA IN THE INCREMENT REGISTER
1422      MOV     @CSR,$BDDAT      ;READ STATUS
1423      BIT     #BIT13,$BDDAT    ;TEST FOR UNEXPECTED CELL OVERFLOW
1424      BEQ     3$              ;BR IF NOT
1425      MOV     #BIT4.BIT2,$GDDAT ;LOAD THE EXPECTED TYPEOUT
1426      ERROR   31              ;UNEXPECTED 'CELL OVERFLOW' STATUS
1427      BIS     #BIT13,@SFR      ;TRY TO CLEAR THE FLAG
1428      ASL     $TEMP            ;CHANGE THE EXPCTED
1429      BNE     1$              ;BR IF MORE DATA BITS

```

```
1429      ;*****
(3)      ;*TEST 115      CHECK FOR LOW BYTE 'INC OVFL' TO SET CELL OVERFLOW AND 'CLR CELL' TO CLE
(3)      ;*****
(2) 017614 000004
(1) 017616 012737 000040 001160      TST115: SCOPE
1430 017624 012777 004000 162124      MOV #40,$TIMES      ;;DO 40 ITERATIONS
1431 017632 005077 162114      MOV #CLRALL,@SFR      ;CLEAR THE DEVICE
1432 017636 005077 162112      CLR @WCR      ;CLEAR WC
1433 017642 012777 060000 162100      CLR @BAR      ;CLEAR BAR
1434 017650 012777 000022 162070      MOV #BUFO,@OFF      ;LOAD INITIAL OFFSET REGISTER
1435 017656 012777 000014 162072      MOV #BIT4!BIT1,@CSR      ;ENABLE BYTE MATRIX MODE
1436 017664 052777 000001 162054      MOV #TSTDMA!TSTCON,@SFR      ;SET TEST DMA AND CONTROL
1437 017672 052777 000002 162056      BIS #BIT0,@CSR      ;ENABLE THE DEVICE
(1) 017700 042777 000002 162050      BIS #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
1438 017706 000240      BIC #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
1439 017710 000240      NOP
1440 017712 000240      NOP
1441 017714 112737 000377 060000      MOVB #377,BUFO      ;SET LOW BYTE OF BUFO LOC. TO BYTE -1
1442 017722 112737 000200 060001      MOVB #200,BUFO+1      ;SET HIGH BYTE OF BUFO TO KNOWN VALUE
1443 017730 052777 010000 162020      BIS #BIT12,@SFR      ;ALLOW 1 TRANSFER
1444 017736 000240      NOP
1445 017740 000240      NOP
1446 017742 000240      NOP
1447 017744 017737 161776 001126      MOV @CSR,$BDDAT      ;READ STATUS
1448 017752 012737 020022 001124      MOV #BIT13!BIT4!BIT1,$GDDAT ;LOAD EXPECTED STATUS
1449 017760 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE VALUES
1450 017766 001401      BEQ 1$      ;;BR IF SAME
1451 017770 104031      ERROR 31      ;'CELL OVERFLOW' FLOP FAILED TO SET
1452      ;IN BYTE MODE FROM A LOW BYTE OVERFLOW
1453
1454
1455      ;NOW GENERATE 'CLR CELL' TO CLEAR CELL OVERFLOW FLOP
1456 017772 052777 020000 161756 1$: BIS #BIT13,@SFR      ;GENERATE 'CLR CELL'
1457 020000 017737 161742 001126      MOV @CSR,$BDDAT      ;READ STATUS
1458 020006 012737 000022 001124      MOV #BIT4!BIT1,$GDDAT      ;LOAD EXPECTED
1459 020014 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE VALUES
1460 020022 001401      BEQ 2$      ;;BR IF SAME
1461 020024 104031      ERROR 31      ;'CLR CELL' FAILED TO CLEAR 'CELL OVFL' FLOP
1462
1463      ;NOW VERIFY THE BYTE DATA CELL
1464 020026 013737 060000 001126 2$: MOV BUFO,$BDDAT      ;READ DATA
1465 020034 012737 100377 001124      MOV #100377,$GDDAT      ;LOAD EXPECTED
1466 020042 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE
1467 020050 001401      BEQ TST116      ;;BR IF SAME
1468 020052 104031      ERROR 31      ;OVERFLOW FROM INC. REG. FAILED TO INHIBIT
1469      ;'INC CNT' FROM CHANGING THE DATA OR
1470      ;'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,$TIMES ;;DO 40 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CLR @BAR  
CLR @WCR  
MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
MOV #BIT4!BIT2,@CSR ;ENABLE MATRIX MODE  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
NOP  
NOP  
NOP  
MOV #-1,BUFO ;SET BUFO LOC. TO WORD -1  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
NOP  
MOV @CSR,$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) *****  
      (1) TST117: SCOPE  
1509 020310 012737 000040 001160 MOV #40,$TIMES ;;DO 40 ITERATIONS  
1510 020316 012777 004000 161432 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
1511 020324 005077 161424 CLR @BAR  
1512 020330 005077 161416 CLR @WCR  
1513 020334 012777 060000 161406 MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
1514 020342 012777 000064 161376 MOV #BIT5!BIT4!BIT2,@CSR ;ENABLE INTR. AND MATRIX MODE  
1515 020350 012777 000014 161400 MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
1516 020356 052777 000001 161362 BIS #BIT0,@CSR ;ENABLE THE DEVICE  
1517 020364 052777 000002 161364 BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
      (1) 020372 042777 000002 161356 BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
1518 020400 000240 NOP  
1519 020402 000240 NOP  
1520 020404 000240 NOP  
1521 020406 012737 177777 060000 MOV #-1,BUFO ;SET BUFO LOC. TO WORD -1  
1522 020414 012746 000000 MOV #0,-(SP)  
1523 020420 012746 020426 MOV #1$,-(SP)  
1524 020424 000002 RTI  
1525 020426 012777 020454 161342 1$: MOV #2$,@VECTB0 ;LOAD RETURN VECTOR  
1526 020434 052777 010000 161314 BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
1527 020442 000240 NOP  
1528 020444 000240 NOP  
1529 020446 000240 NOP  
1530 020450 104032 ERROR 32 ;'CELL OVERFLOW' FAILED TO CAUES AN INTERRUPT  
1531 020452 000414 BR 3$ ;;BR TO CLEAN UP  
1532 020454 022626 2$: CMP (SP)+,(SP)+  
1533 020456 017737 161264 001126 MOV @CSR,$BDDAT ;READ STATUS  
1534 020464 012737 020264 001124 MOV #BIT13!BIT7!BIT5!BIT4!BIT2,$GDDAT ;LOAD EXPECTED  
1535 020472 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUE  
1536 020500 001401 BEQ 3$ ;;BR IF 'ACTIVE' CLEARED  
1537 020502 104031 ERROR 31 ;'ACTIVE' FAILED TO CLEAR  
1538 020504 012777 004000 161244 3$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
1539 020512 013777 002000 161256 MOV VECTB1,@VECTB0 ;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 'TFST 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

```
*****  
*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

```
*****  
*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) 21262 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
(1)

```
*****  
*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
(1)

```
*****  
*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
(1)

```
*****  
*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,$GDDAT ;LOAD EXPECTED DATA  
MOV @SBDADR,$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
(1)

```
*****  
*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,$GDDAT ;LOAD EXPECTED DATA  
MOV @SBDADR,$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

```
*****  
*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
(1)

```
*****  
*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 040004 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 040004
(1) 022776 052777 010000 156752
(1) 023004 012737 040004 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,$TIMES ;:DO 100 ITERATIONS  
MOV #CLRALL,@SFR ;:CLEAR THE DEVICE  
MOV #40004,@OFF ;:LOAD THE OFFSET REG.  
MOV #TSTDMA!TSTCON,@SFR ;:SET TEST DMA AND CONTROL  
MOV #BIT4!BIT2,@CSR ;:SELECT MATRIX WORD MODE  
BIS #BIT0,@CSR ;:ENABLE NCV11  
BIS #TESTZ,@SFR ;:ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;:DISABLE 'TEST Z' PULSES  
CLR @#40004 ;:CLEAR THE TARGET LOCATION  
BIS #BIT12,@SFR ;:ALLOW 1 WORD TRANSFER  
MOV #40004,$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 40004
```

```

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: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
:WITH GAIN SET IN TEST CONTROL MODE, THE SELF TEST DATA SHOULD BE BYTE 250
:WHEN PROCESSED THRU THE ADDRESS MAKER LOGIC THE NEW VALUE IS '2552'
:VERIFY THAT THE MATRIX MUX CAN ADD 62550 AND 2552 CORRECTLY
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUF0+2550,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2024,@CSR ;ENABLE GAIN, WORD MATRIX MODE
BIS #BIT0,@CSR ;ENABLE THE NCV11
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
CLR @BUF0+5322 ;CLEAR THE TARGET ADDRESS
CLR @BUF0+2722 ;CLEAR THE TARGET ADDRESS IF 'TESTX' FAILS
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
NOP
NOP
NOP
MOV #BUF0+5322,$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 @BUF0+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 @BUF0+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,$TIMES ;;DO 10 ITERATIONS
:WITH ZB ENABLE SET IN TEST CONTROL MODE, THE SELF TEST DATA SHOULD BE BYTE 224
:WHEN PROCESSED THRU THE ADDRESS MAKER LOGIC THE NEW VALUE IS 12450
:VERIFY THAT THE MATRIX MUX CAN ADD CORRECTLY
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUF0+5224,@OFF ;LOAD THE OFFSET REGISTER
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #4024,@CSR ;ENABLE ZB AND WORD MATRIX MODE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BIT0,@CSR ;ENABLE NCV11
BIC #BIT0,@SFR ;DISABLE 'TEST Z' PULSES
CLR @#BUF0+12450 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
MOV #BUF0+12450,$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 BUF0 + 2450
:*****
:*TEST 137 VERIFY LOW BYTE OPERATION OF THE 'TESTX' FLOP
:*****
TST137: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUF0,@OFF ;LOAD OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;LOAD TEST DMA AND CONTROL
MOV #2024,@CSR ;SET 'GAIN' AND WORD MATRIX MODE
BIS #BIT0,@CSR ;SET NCV11 ACTIVE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
BIS #BIT4,@SFR ;SET 'TESTX' FLOP
CLR @#BUF0+2552 ;CLEAR THE TEST FAILED LOCATION
CLR @#BUF0+0152 ;CLEAR THE TESTX WORKED LOCATION
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
NOP
NOP
NOP
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @#BUF0+0152,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE DATA
BEQ TST140 ;;BR IF SAME
TST @#BUF0+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 @#BUF0+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 000002 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,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #50000,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2030,@CSR ;SELECT GAIN MATRIX BYTE MODE
BIS #BIT0,@CSR ;ENABLE NCV11
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
CLR @BUF0+2726 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER
MOV #BUF0+2726,$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 BUF0+2726
:*****
:*TEST 141 VERIFY BIT 13 ADDER INPUT TO MATRIX MODE MUX
:*****
TST141: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #37774,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2032,@CSR ;SELECT GAIN MATRIX BYTE MODE
BIS #BIT0,@CSR ;ENABLE NCV11
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
CLR @BUF0+5522 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER
MOV #BUF0+5522,$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 BUF0+5522

```

```
1743 :*****  
(3) :*TEST 142 VERIFY BIT 14 ADDER INPUT TO MATRIX MODE MUX  
(3) :*****  
(2) 024024 000004 TST142: SCOPE  
(1) 024026 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS  
1744 :OFFSET = 50000  
1745 :ADM OUTPUT = 53656  
1746 :TARGET = 123656  
1747 024034 012777 004000 155714 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
1748 024042 023727 036404 001340 CMP $LSTBK,#1340 ;TEST IF ENOUGH MEMORY >20K  
1749 024050 103445 BLO TST143 ;;BR IF NO ROOM  
1750 024052 012777 050000 155670 MOV #50000,@OFF ;LOAD THE OFFSET REG.  
1751 024060 012777 000014 155670 MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
1752 024066 012777 002034 155652 MOV #2034,@CSR ;SELECT GAIN MATRIX BYTE MODE  
1753 024074 052777 000001 155644 BIS #BIT0,@CSR ;ENABLE NCV11  
1754 024102 052777 000002 155646 BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
(1) 024110 042777 000002 155640 BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
1755 024116 005037 123656 CLR @BUF0+43656 ;CLEAR THE TARGET LOCATION  
1756 024122 052777 010000 155626 BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER  
1757 024130 012737 123656 001122 MOV #BUF0+43656,$BDADR ;LOAD EXPECTED ADDRESS  
1758 024136 012737 000001 001124 MOV #1,$GDDAT ;LOAD EXPECTED DATA  
1759 024144 017737 154752 001126 MOV @SBDADR,$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 BUF0+43656
```


1765
(3)
(3)
(2) 024164 000004
(1) 024166 012737 000010 001160
1766 024174 012777 004000 155554
1767 024202 005077 155544
1768 024206 005077 155542
1769 024212 012777 060000 155530
1770 024220 012777 002022 155520
1771 024226 012777 000014 155522
1772 024234 052777 000001 155504
1773 024242 052777 000002 155506
(1) 024250 042777 000002 155500
1774 024256 000240
1775 024260 000240
1776 024262 000240
1777 024264 112737 000100 061264
1778 024272 112737 000377 061265
1779 024300 052777 010000 155450
1780 024306 000240
1781 024310 000240
1782 024312 000240
1783 024314 012737 022022 001124
1784 024322 017737 155420 001126
1785 024330 023737 001124 001126
1786 024336 001401
1787 024340 104030
1788
1789 024342 013737 061264 001126 1\$:
1790 024350 012737 177500 001124
1791 024356 023737 001124 001126
1792 024364 001401
1793 024366 104030

```
*****  
*TEST 143 CHECK FOR HIGH BYTE 'INC OVFL' TO SET CELL OVERFLOW  
*****  
TST143: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
CLR @WCR ;CLEAR W.C.  
CLR @BAR ;CLEAR B.A.  
MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER  
MOV #BIT10!BIT4!BIT1,@CSR ;ENABLE MATRIX MODE BYTE AND SET GAIN  
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL  
BIS #BIT0,@CSR ;ENABLE THE DEVICE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
NOP  
NOP  
NOP  
MOVB #100,BUFO+1264 ;LOAD LOW BYTE TO A KNOWN VALUE  
MOVB #377,BUFO+1265 ;SET HIGH BYTE TO 377  
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER  
NOP  
NOP  
NOP  
MOV #BIT13!BIT10!BIT4!BIT1,$GDDAT ;LOAD EXPECTED STATUS  
MOV @CSR,$BDDAT ;READ THE ACTUAL STATUS  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ 1$ ;;BR IF SAME  
ERROR 30 ;CELL OVERFLOW FAILED TO SET  
;WHEN INCREMENTING THE HIGH BYTE  
MOV BUFO+1264,$BDDAT ;READ THE BUFO LOCATION  
MOV #177500,$GDDAT ;LOAD EXPECTED VALUE  
CMP $GDDAT,$BDDAT ;COMPARE VALUES  
BEQ TST144 ;;BR IF SAME  
ERROR 30 ;TARGET LOC. DATA WAS INCORRECT
```

```
1795
(3)
(3)
(2) 024370 000004
(1) 024372 012737 000400 001160
1796 024400 005737 050024
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 $GDDAT,@BAR ;
1805 024452 005377 155276 DEC @BAR ;LOAD INITIAL COUNTER VALUE
1806 024456 012777 000022 155262 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
1807 024464 052777 000001 155254 BIS #BIT0,@CSR ;ENABLE THE DEVICE
1808 024472 052777 000001 155226 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
1809 024500 105777 155222 2$: 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) *TEST 145 VERIFY EACH BIT OF THE HIGHER 16 BIT Z COUNTER (TESTER ONLY)
(3)
(2) 024536 000004
(1) 024540 012737 000400 001160
1819 024546 005737 050024
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 TST146: SCOPE  
(1) 024706 012737 000400 001160 MOV #400,$TIMES ;;DO 400 ITERATIONS  
1843 024714 005737 050024 TST WFMODE ;TEST IF TESTER MODE  
1844 024720 001443 BEQ TST147 ;;BR IF NOT  
1845 024722 012737 000001 001124 MOV #1,$GDDAT ;LOAD EXPECTED VALUE  
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 BPL 2$  
1856 025010 017737 154740 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE  
1857 025016 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
1858 025024 001401 BEQ TST147 ;;BR IF SAME  
1859 025026 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER  
1860 ;IN ERROR WHEN USING CAMERA 01  
1861  
1862
```

```
(3) :*****  
(3) :*TEST 147 VERIFY THAT CAMERA 10 CHANNEL IS OPERATIONAL (TESTER ONLY)  
(3) :*****  
(2) 025030 000004 TST147: SCOPE  
(1) 025032 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS  
1863 025040 005737 050024 TST WFMODE ;TEST IF TESTER MODE  
1864 025044 001443 BEQ TST150 ;;BR IF NOT  
1865 025046 012737 000001 001124 MOV #1,$GDDAT ;LOAD EXPECTED VALUE  
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 BPL 2$  
1876 025134 017737 154614 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE  
1877 025142 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
1878 025150 001401 BEQ TST150 ;;BR IF SAME  
1879 025152 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER  
1880 ;IN ERROR WHEN USING CAMERA 10
```

```
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,$TIMES      ;;DO 100 ITERATIONS  
1883 025164 005737 050024      TST    WFMODE          ;:TEST IF TESTER MODE  
1884 025170 001443      BEQ    TST151          ;;BR IF NOT  
1885 025172 012737 000001 001124      MOV    #1,$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  
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,$TIMES ;DO 2 ITERATIONS  
1903 :CELL INCREMENT OF A FLOATING LOCATION (60000,70000,64000,62000,61000)  
1904 :FILL THE MEMORY BUFFER WITH A KNOWN PATTERN (125252)  
1905 :COLLECT DATA OF 0 AND INCREMENT A TARGET LOCATION  
1906 :VERIFY THAT NO OTHER ADDRESS IS CHANGED  
1907 :AFTER VERIFYING THE WHOLE BUFFER - UPDATE THE OFFSET REGISTER  
1908 : AND INCREMENT ANOTHER LOCATION  
1909 025310 012737 060000 050030 MOV #BUF0,CURRENT ;PRIME THE CURRENT TARGET LOC.  
1910 025316 012737 020000 025664 MOV #BIT13,100$ ;LOAD FORCE BIT  
1911 025324 012737 025332 001110 MOV #2,$SLPERR ;LOAD RETURN ADDRESS  
1912 :PRIME THE BUFFER WITH A 125252 PATTERN  
1913 025332 012700 060000 2$: MOV #BUF0,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$: MOV R1,(R0)+ ;LOAD BUFFER WITH DATA  
1917 025350 020002 CMP R0,R2 ;TEST FOR ENL  
1918 025352 001375 BNE 3$  
1919 025354 012777 177770 022446 MOV #-10,@CURRENT ;PRIME THE TARGET LOCATION  
1920  
1921 025362 012777 004000 154366 MOV #CLRALL,@SFR ;INIT THE DEVICE  
1922 025370 013777 050030 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 000470 BR TST152 ;:  
1942 025506 012700 060000 5$: MOV #BUF0,R0 ;GET BUFFER POINER  
1943 025512 023700 050030 6$: 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 001417 BEQ 10$ ;:BR IF SAME  
1950 025546 104033 FRROR 33 ;CHANGED AN INCORRECT TARGET LOCATION  
1951 025550 000446 BR TST152 ;:  
1952 025552 013737 050030 001122 7$: MOV CURRENT,$BDADR ;LOAD EXPECTED ADDRESS  
1953 025560 017737 022244 001126 MOV @CURRENT,$BDDAT ;LOAD ACTUAL DATA
```

```

1954 025566 012737 003407 001124      MOV    #3407,$GDDAT    ;LOAD EXPECTED DATA
1955 025574 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;COMPARE
1956 025602 001400                      BEQ    10$             ;:BR IF SAME
1957
1958 025604 005720                      10$:  TST    (R0)+       ;BUMP THE POINTER INTO THE BUFFER
1959 025606 023700 003354      CMP    ADNOKT,R0      ;:TEST IF FINISHED THE BUFFER
1960 025612 001337                      BNE    6$             ;:BR AND RETEST THE REST OF THE BUFFER
1961
1962 025614 005737 001202                      TST    $PASS          ;:TEST IF FIRST PASS
1963 025620 001422                      BEQ    TST152         ;:BR IF YES
1964 025622 032777 004000 153310      BIT    #SW11,@SWR     ;:TEST INHIBIT INTER.
1965 025630 001016                      BNE    TST152         ;:BR IF SET
1966 025632 006237 025664                      ASR    100$           ;:CHANGE THE FORCED ADDR. BIT
1967 025636 022737 000002 025664      CMP    #2,100$        ;:TEST IF FINISHED
1968 025644 001410                      BEQ    TST152         ;:BR IF FINISHED
1969 025646 012737 060000 050030      MOV    #BUFO,CURRENT ;MAKE UP NEW ADDRESS
1970 025654 053737 025664 050030      BIS    100$,CURRENT  ;
1971 025662 000623                      BR     2$             ;
1972 025664 020000                      100$: BIT13
1973
1974 (3)
1975 (3)
1976 (2) 025666 000004
1977 (1) 025670 012737 000002 001160
1978 025676 012737 060000 050030      MOV    #BUFO,CURRENT ;PRIME THE CURRENT TARGET LOC.
1979 025704 012737 020000 026244      MOV    #BIT13,100$   ;LOAD FORCE BIT
1980 025712 012737 025720 001110      MOV    #2$,SLPERR    ;LOAD RETURN ADDRESS
1981
1982 025720 012700 060000      2$:  MOV    #BUFO,R0    ;LOAD POINTER TO BUFFER
1983 025724 012701 125252      MOV    #125252,R1    ;LOAD VALUE
1984 025730 013702 003354      MOV    ADNOKT,R2     ;LOAD BUFFER END ADDRESS
1985 025734 010120      3$:  MOV    R1,(R0)+     ;LOAD BUFFER WITH DATA
1986 025736 020002      CMP    R0,R2         ;TEST FOR END
1987 025740 001375      BNE    3$
1988
1989 025742 012777 004000 154006      MOV    #CLRALL,@SFR  ;INIT THE DEVICE
1990 025750 012777 177777 153774      MOV    #-1,@WCR      ;PRIME THE WORD COUNT
1991 025756 013777 050030 153770      MOV    CURRENT,@BAR  ;LOAD TARGET LOCATION POINTER
1992 025764 012777 000004 153764      MOV    #TSTCON,@SFR ;ENABLE MAINT. MODE
1993 025772 052777 000001 153746      BIS    #BIT0,@CSR    ;ENABLE THE NCV11
1994 026000 052777 000002 153750      BIS    #TESTZ,@SFR  ;ENABLE 'TEST Z' PULSES
1995 (1) 026006 042777 000002 153742      BIC    #TESTZ,@SFR  ;DISABLE 'TEST Z' PULSES
1996
1997 026014 013700 002012                      MOV    CPUDL2,R0     ;PRIME THE DELAY
1998 026020 005001
1999 026022 032777 060000 153716 4$:  BIT    #BIT14:BIT13,@CSR ;TEST FOR CELL OR Z OVERFLOW
2000 026030 001014                      BNE    5$             ;:BR IF EITHER IS SET
2001 026032 005301                      DEC    R1             ;:DELAY
2002 026034 001372                      BNE    4$
2003 026036 005300                      DEC    R0             ;:DELAY
2004 026040 001370                      BNE    4$
2004 026042 017737 153700 001126      MOV    @CSR,$BDDAT   ;READ BAD STATUS

```

2005	026050	012737	060224	001124		MOV	#BIT14.BIT13+224,	\$GDDAT ;LOAD EXPECTED STATUS
2006	026056	104036				ERROR	36	;DYNAMIC LIST MODE STATUS ERROR
2007	026060	000472				BR	TST153	::
2008	026062	012700	060000		5\$:	MOV	#BUFO,R0	;GET BUFFER POINER
2009	026066	023700	050030		6\$:	CMP	CURRENT,R0	;TEST IF TARGET ADDRESS
2010	026072	001415				BEQ	7\$::BR IF YES
2011	026074	010037	001122			MOV	R0,\$BDADR	;GET BAD ADDRESS FOR TYPE-OUT
2012	026100	011037	001126			MOV	(R0),\$BDDAT	;GET BAD DATA
2013	026104	012737	125252	001124		MOV	#125252,\$GDDAT	;LOAD EXPECTED DATA
2014	026112	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;COMPARE
2015	026120	001421				BEQ	10\$::BR IF SAME
2016	026122	104021				ERROR	21	;CHANGED AN INCORRECT TARGET LOCATION
2017	026124	000450				BR	TST153	::
2018	026126	013737	050030	001122	7\$:	MOV	CURRENT,\$BDADR	;LOAD EXPECTED ADDRESS
2019	026134	017737	021670	001126		MOV	@CURRENT,\$BDDAT	;GET ACTUAL DATA
2020	026142	012737	003407	001124		MOV	#3407,\$GDDAT	;LOAD EXPECTED DATA
2021	026150	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;COMPARE
2022	026156	001402				BEQ	10\$::BR IF SAME
2023	026160	104037				ERROR	37	;LIST MODE DATA ERROR
2024	026162	000431				BR	TST153	::
2025	026164	005720			10\$:	TST	(R0)+	;BUMP THE POINTER INTO THE BUFFER
2026	026166	023700	003354			CMP	ADNOKT,R0	;TEST IF FINISHED THE BUFFER
2027	026172	001335				BNE	6\$;BR AND RETEST THE REST OF THE BUFFER
2028								
2029	026174	005737	001202			TST	\$PASS	;TEST IF FIRST PASS
2030	026200	001422				BEQ	TST153	::BR IF FIRST PASS
2031	026202	032777	004000	152730		BIT	#SW11,@SWR	;TEST INHIBIT INTER.
2032	026210	001016				BNE	TST153	::BR IF SET
2033	026212	006237	026244			ASR	100\$;CHANGE THE FORCE ADDRESS
2034	026216	022737	000002	026244		CMP	#2,100\$;TEST IF END
2035	026224	001410				BEQ	TST153	::BR IF FINISHED
2036	026226	012737	060000	050030		MOV	#BUFO,CURRENT	;MAKE NEW ADDRESS
2037	026234	053737	026244	050030		BIS	100\$,CURRENT	:
2038	026242	000626				BR	2\$:
2039	026244	020000			100\$:	BIT13		:

```

2041      ;*****
(3)      ;*TEST 153      DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K
(3)      ;*****
(2) 026246 000004      TST153: SCOPE
(1) 026250 012737 000040 001160      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
2042 026256 013700 003354      MOV      ADNOKT,R0      ;GET LAST ADDRESS
2043 026262 162700 060000      SUB      #BUFO,R0      ;DETERMINE THE WORD COUNT VALUE
2044 026266 005400      NEG      R0
2045 026270 006200      ASR      R0
2046      ;PRIME THE BUFFER WITH A 125252 PATTERN
2047 026272 012703 060000      2$: MOV      #BUFO,R3      ;LOAD POINTER TO BUFFER
2048 026276 012701 125252      MOV      #125252,R1      ;LOAD VALUE
2049 026302 013702 003354      MOV      ADNOKT,R2      ;LOAD BUFFER END ADDRESS
2050 026306 010123      3$: MOV      R1,(R3)+      ;LOAD BUFFER WITH DATA
2051 026310 020302      CMP      R3,R2      ;TEST FOR END
2052 026312 001375      BNE      3$
2053      ;NOW COLLET THE LIST MODE DATA
2054 026314 012777 004000 153434      MOV      #CLRALL,@SFR      ;INIT THE DEVICE
2055 026322 010077 153424      MOV      R0,@WCR      ;PRIME THE WORD COUNT
2056 026326 012777 060000 153420      MOV      #BUFO,@BAR      ;LOAD TARGET LOCATION POINTER
2057 026334 012777 000004 153414      MOV      #TSTCON,@SFR      ;ENABLE MAINT. MODE
2058 026342 052777 000001 153376      BIS      #BIT0,@CSR      ;ENABLE THE NCV11
2059 026350 052777 000002 153400      BIS      #TESTZ,@SFR      ;ENABLE MAINT. Z
2060 026356 013700 002012      MOV      CPUDL2,R0      ;PRIME THE DELAY
2061 026362 032777 060000 153356      4$: BIT      #BIT14!BIT13,@CSR      ;TEST FOR CELL OR Z OVERFLOW
2062 026370 001014      BNE      5$      ;;BR IF EITHER IS SET
2063 026372 005301      DEC      R1      ;DELAY
2064 026374 001372      BNE      4$
2065 026376 005300      DEC      R0      ;DELAY
2066 026400 001370      BNE      4$
2067 026402 017737 153340 001126      MOV      @CSR,$BDDAT      ;READ BAD STATUS
2068 026410 012737 060200 001124      MOV      #BIT14!BIT13+200,$GDDAT ;LOAD EXPECTED STATUS
2069 026416 104036      ERROR 36 ;DYNAMIC LIST MODE STATUS ERROR - NO WORD COUNT OVERFLOW
2070 026420 000423      BR      TST154      ;;
2071 026422 012700 060000      5$: MOV      #BUFO,R0      ;GET BUFFER POINER
2072 026426 010037 001122      6$: MOV      R0,$BDADR      ;GET BAD ADDRESS FOR TYPE-OUT
2073 026432 011037 001126      MOV      (R0),$BDDAT      ;GET BAD DATA
2074 026436 012737 003407 001124      MOV      #3407,$GDDAT      ;LOAD EXPECTED DATA
2075 026444 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
2076 026452 001402      BEQ      10$      ;;BR IF SAME
2077 026454 104037      ERROR 37 ;LIST MODE DATA ERROR
2078 026456 000404      BR      TST154      ;;
2079 026460 005720      10$: TST      (R0)+      ;BUMP THE POINTER INTO THE BUFFER
2080 026462 023700 003354      CMP      ADNOKT,R0      ;TEST IF FINISHED THE BUFFER
2081 026466 001357      BNE      6$      ;BR AND RETEST THE REST OF THE BUFFER
  
```


2083
 (3)
 (3)
 (2) 026470 000004
 (1) 026472 012737 000002 001160
 2084 026500 005737 036122
 2085 026504 100127
 2086
 2087
 2088 026506 012737 000600 041742
 2089 026514 013737 041742 172346
 2090 026522 012700 060000
 2091 026526 012710 125252
 2092 026532 052737 001400 177572
 2093 026540 012710 000370
 2094 026544 042737 001400 177572
 2095 026552 013701 041742
 2096 026556 005002
 2099 026560 006301
 (1) 026562 006301
 (1) 026564 006301
 (1) 026566 006301
 2100 026570 006301
 2101 026572 006102
 2102 026574 006301
 2103 026576 006102
 2104 026600 010237 041726
 2105 026604 010137 001122
 2106 026610 060201
 2107
 2108
 2109 026612 012777 004000 153136
 2110 026620 010177 153124
 2111 026624 012777 000022 153114
 2112 026632 012777 000014 153116
 2113 026640 052777 000001 153100
 2114 026646 052777 000002 153102
 (1) 026654 042777 000002 153074
 2115 026662 012737 000371 001124
 2116 026670 052777 010000 153060
 2117 026676 052737 000001 177572
 2118 026704 011037 001126
 2119 026710 042737 000001 177572
 2120 026716 023737 001124 001126
 2121 026724 001401
 2122 026726 104035
 2123 026730 005737 001202
 2124 026734 001413
 2125 026736 032777 004000 152174
 2126 026744 001007
 2127 026746 062737 000200 041742
 2128 026754 023737 036404 041742
 2129 026762 101254

```

:*****
:*TEST 154 ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY
:*****
TST154: SCOPE
MOV #2,$TIMES ;DO 2 ITERATIONS
TST $KT11 ;TEST IF KT-11 INSTALLED
BPL TST155 ;BR IF NONE
;DO A BYTE MATRIX MODE TRANSFER TO A 1 BYTE LOCATION IN EACH 4K EXTENDED
;MEMORY BANK
MOV #600,OUT ;LOAD INITIAL BANK VALUE
1$: MOV OUT,@#KIPAR3 ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
MOV #BUF0,R0 ;LOAD BUFFER POINTER
MOV #125252,(R0) ;PRIME BAD TARGET LOC. IF EXT ADD FAILS
BIS #1400,@#SRO ;ENABLE MAINT. MODE KT-11
MOV #370,(R0) ;LOAD TARGET LOCATION DATA VALUE
BIC #1400,@#SRO ;DISABLE MAINT. MODE KT-11
MOV OUT,R1 ;GET BANK VALUE
CLR R2 ;CLEAR EXT. ADD. TEMPORARY
ASL R1 ;MOVE LEFT
(1) ASL R1 ;MOVE LEFT
(1) ASL R1 ;MOVE LEFT
(1) ASL R1 ;MOVE LEFT
2100 ASL R1
2101 ROL R2 ;SAVE EA BITS
2102 ASL R1
2103 ROL R2 ;SAVE EA BITS
2104 MOV R2,NARROW ;SAVE EA BITS FOR TYPEOUT
2105 MOV R1,$BDADR ;SAVE ADDRESS BITS FOR TYPEOUT
2106 ADD R2,R1 ;MAKE COMPLETE ADDRESS

;NOW GET READY TO DO THE TRANSFER
MOV #CLRALL,@SFR ;INIT THE NCV11
MOV R1,@OFF ;LOAD COMBINED BUFFER ADDRESS
MOV #BIT4!BIT1,@CSR ;ENABLE THE NCV11
MOV #TSTDMA!TSTCON,@SFR ;ENABLE MAINT NCV11 MODE
BIS #BIT0,@CSR ;ENABLE THE NCV11
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
2115 MOV #371,$GDDAT ;LOAD EXPECTED DATA
2116 BIS #BIT12,@SFR ;ENABLE 1 BYTE TRANSFER
2117 BIS #BIT0,@#SRO ;ENABLE KT-11
2118 MOV (R0),$BDDAT ;GET ACTUAL DATA
2119 BIC #BIT0,@#SRO ;DISABLE KT-11
2120 CMP $GDDAT,$BDDAT ;COMPARE DATA
2121 BEQ 2$ ;BR IF SAME
2122 ERROR 3$ ;DATA TRANSFER ERROR TO EXTENDED MEMORY
2123 2$: TST $PASS ;TEST PASS COUNTER
2124 BEQ TST155 ;BR IF FIRST PASS
2125 BIT #SW11,@SWR ;TEST IF INHIBIT INTER.
2126 BNE TST155 ;BR IF SET
2127 ADD #200,OUT ;UPDATE BANK VALUE
2128 CMP $LSTBK,OUT ;TEST IF DONE
2129 BHI 1$ ;BR IF NOT

```

```

2131      ::*****
(3)      :*TEST 155      ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY
(3)      :*****
(2) 026764 000004
(1) 026766 012737 000002 001160      TST155: SCOPE
2132 026774 005737 036122      MOV #2,$TIMES      ;;DO 2 ITERATIONS
2133 027000 100130      TST $KT11      ;:TEST IF KT-11 INSTALLED
2134      BPL TST156      ;:BR IF NONE
2135 027002 012737 000600 041742      ;DO A 1 WORD MODE TRANSFER TO A 1 WORD BUFFER IN EACH 4K EXTENDED MEMORY BANK
2136 027010 013737 041742 172346      MOV #600,OUT      ;LOAD INITIAL BANK VALUE <60000>
2137 027016 012700 060000      1$: MOV OUT,@#K!PAR3      ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
2138 027022 012710 052525      MOV #BUF0,R0      ;LOAD BUFFER POINTER
2139 027026 052737 001400 177572      MOV #52525,(R0)      ;LOAD BAD TARGET LOC. IF EXT ADD FAILS
2140 027034 012710 125252      BIS #1400,@#SRO      ;ENABLE MAINT. MODE KT-11
2141 027040 042737 001400 177572      MOV #125252,(R0)      ;PRESET THE TARGET LOCATION
2142 027046 013701 041742      BIC #1400,@#SRO      ;DISABLE MAINT. MODE KT-11
2143 027052 005002      MOV OUT,R1      ;GET BANK VALUE
2146 027054 006301      CLR R2      ;CLEAR EXT. ADD TEMPORARY
(1) 027056 006301      ASL R1
(1) 027060 006301      ASL R1
(1) 027062 006301      ASL R1
2147 027064 006301      ASL R1
2148 027066 006102      ROL R2      ;SAVE EXT. ADDRESS BITS
2149 027070 006301      ASL R1
2150 027072 006102      ROL R2      ;SAVE EXT. ADDRESS BITS
2151
2152      ;NOW GET READY TO DO THE TRANSFER
2153 027074 012777 004000 152654      MOV #CLRALL,@SFR      ;INIT THE NCV11
2154 027102 010277 152642      MOV R2,@OFF      ;LOAD THE EXTENDED ADDRESS BITS
2155 027106 012777 177777 152636      MOV #-1,@WCR      ;LOAD WORD COUNT
2156 027114 010177 152634      MOV R1,@BAR      ;LOAD BUS ADDRESS
2157 027120 012777 000014 152630      MOV #TSTDMA!TSTCON,@SFR      ;ENABLE MAINT NCV11 MODE
2158 027126 052777 000001 152612      BIS #BIT0,@CSR      ;ENABLE THE NCV11
2159 027134 052777 000002 152614      BIS #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
(1) 027142 042777 000002 152606      BIC #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
2160 027150 010137 001122      MOV R1,$BDADR      ;SAVE TARGET ADDRESS
2161 027154 010237 041726      MOV R2,NARROW      ;SAVE EA BITS FOR ERROR TYPEOUT
2162 027160 052777 010000 152570      BIS #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER
2163 027166 012737 003407 001124      MOV #3407,$GDDAT      ;LOAD EXPECTED DATA
2164 027174 052737 000001 177572      BIS #BIT0,@#SRO      ;ENABLE KT-11
2165 027202 011037 001126      MOV (R0),$BDDAT      ;GET ACTUAL DATA
2166 027206 042737 000001 177572      BIC #BIT0,@#SRO      ;DISABLE KT-11
2167 027214 023737 001124 001126      CMP $GDDAT,$BDDAT      ;COMPARE DATA
2168 027222 001401      BEQ 2$      ;:BR IF SAME
2169 027224 104035      ERROR 35      ;LIST MODE DATA TRANSFER ERROR TO EXTENDED MEMORY
2170 027226 005737 001202      2$: TST $PASS      ;TEST PASS COUNTER
2171 027232 001413      BEQ TST156      ;:BR IF FIRST PASS
2172 027234 032777 004000 151676      BIT #SW11,@SWR      ;TEST INHIBIT INTER.
2173 027242 001007      BNE TST156      ;:BR IF SET
2174 027244 062737 000200 041742      ADD #200,OUT      ;UPDATE BANK VALUE
2175 027252 023737 036404 041742      CMP $LSTBK,OUT      ;TEST IF DONE
2176 027260 101253      BHI 1$      ;BR IF NOT

```

```

2178      (3)
2179      (3)
2180      (2) 027262 000004
2181      (1) 027264 012737 000100 001160
2182      027272 023727 036404 003140
2183      027300 103476
2184      027302 012737 002676 172346
2185      027310 012700 060056
2186      027314 012710 125252
2187      027320 052737 001400 177572
2188      027326 005010
2189      027330 042737 001400 177572
2190
2191
2192      027336 012777 004000 152412
2193      027344 012777 140000 152376
2194      027352 012777 002036 152366
2195      027360 012777 000014 152370
2196      027366 052777 000001 152352
2197      027374 052777 000002 152354
2198      (1) 027402 042777 000002 152346
2199      027410 000240
2200      027412 000240
2201      027414 052777 010000 152334
2202      027422 000240
2203      027424 000240
2204      027426 000240
2205      027430 012737 000001 041726
2206      027436 012737 000400 001124
2207      027444 052737 000001 177572
2208      027452 011037 001126
2209      027456 042737 000001 177572
2210      027464 023737 001124 001126
2211      027472 001401
2212      027474 104020

```

```

:*****
:*TEST 156 VERIFY BIT 15 MATRIX ADDER INPUT
:*****
TST156: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
:ADM OUTPUT = 127657
:OFFSET = 140000
:TARGET = 267657
CMP $LSTBK,#3140 ;:TEST IF >52K IS AVAILABLE
BLO TST157 ;:BR IF NO MORE MEMORY
MOV #2676,@#KIPAR3 ;:LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
MOV #BUF0+56,R0 ;:LOAD BUFFER POINTER
MOV #125252,(R0) ;:LOAD BAD TARGET LOC. IF EXT ADD FAILS
BIS #1400,@#SRO ;:ENABLE MAINT. MODE KT-11
CLR (R0) ;:LOAD TARGET LOCATION DATA VALUE
BIC #1400,@#SRO ;:DISABLE MAINT. MODE KT-11

:NOW GET READY TO DO THE TRANSFER
MOV #CLRALL,@SFR ;:INIT THE NCV11
MOV #140000,@OFF ;:LOAD COMBINED BUFFER ADDRESS
MOV #2036,@CSR ;:ENABLE THE NCV11
MOV #TSTDMA!TSTCON,@SFR ;:ENABLE MAINT NCV11 MODE
BIS #BIT0,@CSR ;:ENABLE THE NCV11
BIS #TESTZ,@SFR ;:ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;:DISABLE 'TEST Z' PULSES
NOP
NOP
BIS #BIT12,@SFR ;:ENABLE 1 BYTE TRANSFER
NOP
NOP
MOV #1,NARROW ;:SAVE EA BITS FOR ERROR TYPEOUT
MOV #400,$GDDAT ;:LOAD EXPECTED DATA
BIS #BIT0,@#SRO ;:ENABLE KT-11
MOV (R0),$BDDAT ;:SAVE ACTUAL DATA
BIC #BIT0,@#SRO ;:DISABLE KT-11
CMP $GDDAT,$BDDAT ;:COMPARE DATA
BEQ TST157 ;:BR IF SAME
ERROR 20 ;:BIT 15 INPUT TO MATRIX MODE ADDER FAILED

```

```

2213      ;:*****
(3)      ;*TEST 157      VERIFY HIGH BYTE OPERATION OF THE 'TEST X'
(3)      ;:*****
(2) 027476 000004      TST157: SCOPE
(1) 027500 012737 000040 001160      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
2214      ;TEST X      =      1
2215      ;OFFSET      =100000
2216      ;ADM OUTPUT      =127657
2217      ;TARGET      =100257
2218 027506 012777 004000 152242      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
2219 027514 023727 036404 001140      CMP      $LSTBK,#1140      ;TEST IF LEAST 20K
2220 027522 103445      BLO      TST160      ;;BR IF NOT
2221 027524 012777 100000 152216      MOV      #100000,@OFF      ;SET BIT 15 OF OFFSET
2222 027532 012777 000034 152216      MOV      #TSTDMA!TSTCON!BIT4,@SFR      ;SET TEST DMA, CONTROL AND TESTX
2223 027540 012777 002036 152200      MOV      #2036,@CSR      ;ENABLE THE NCV11
2224 027546 052777 000001 152172      BIS      #BIT0,@CSR      ;ENABLE THE NCV11
2225 027554 052777 000002 152174      BIS      #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
(1) 027562 042777 000002 152166      BIC      #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
2226 027570 005037 100256      CLR      @#BUF0+20256      ;CLEAR THE TARGET LOC.
2227 027574 052777 010000 152154      BIS      #BIT12,@SFR      ;ALLOW 1 DMA TRANSFER
2228 027602 012737 100256 001122      MOV      #BUF0+20256,$BDADR      ;LOAD EXPECTED ADDRESSES VALUE
2229 027610 012737 000400 001124      MOV      #400,$GDDAT      ;LOAD EXPECTED VALUE READ
2230 027616 017737 151300 001126      MOV      @$BDADR,$BDDAT      ;READ ACTUAL DATA
2231 027624 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
2232 027632 001401      BEQ      TST160      ;;BR IF SAME
2233 027634 104034      ERROR      34      ;MATRIX ADDER INPUT OF ADM15 AND TESTX L
2234      ;FAILED TO BE INHIBITED
    
```

```
2236      ::*****  
(3)      :*TEST 160   VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER  
(3)      :*****  
(2) 027636 000004  
(1) 027640 012737 000100 001160 TST160: SCOPE  
2237      MOV      #100,$TIMES      ;;DO 100 ITERATIONS  
2238      ;ADM OUTPUT =      253527  
2239      ;OFFSET =      210000  
2240      ;TARGET =      463527  
2241 027646 023727 036404 005140      CMP      $LSTBK,#5140      ;TEST IF ENOUGH MEMORY >84K  
2242 027654 103500      BLO      TST161      ;;BR IF NOT  
2243 027656 012737 004635 172346      MOV      #4635,@#KIPAR3      ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>  
2244 027664 012700 060026      MOV      #BUF0+26,R0      ;LOAD BUFFER POINTER  
2245 027670 012710 125252      MOV      #125252,(R0)      ;LOAD BAD TARGET LOC.  
2246 027674 052737 001400 177572      BIS      #1400,@#SRO      ;ENABLE MAINT. MODE KT-11  
2247 027702 005010      CLR      (R0)      ;PRESET THE TARGET LOCATION  
2248 027704 042737 001400 177572      BIC      #1400,@#SRO      ;DISABLE MAINT. MODE KT-11  
2249      ;NOW GET READY TO DO THE TRANSFER  
2250 027712 012777 004000 152036      MOV      #CLRALL,@SFR      ;INIT THE NCV11  
2251 027720 012777 010001 152022      MOV      #10001,@OFF      ;LOAD THE EXTENDED ADDRESS BITS  
2252 027726 012777 000014 152022      MOV      #TSTDMA!TSTCON,@SFR      ;ENABLE MAINT NCV11 MODE  
2253 027734 012777 004036 152004      MOV      #4036,@CSR      ;LOAD MATRIX MODE AND ZB ENABLE  
2254 027742 052777 000001 151776      BIS      #BIT0,@CSR      ;ENABLE THE NCV11  
2255 027750 052777 000002 152000      BIS      #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES  
(1) 027756 042777 000002 151772      BIC      #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES  
2256 027764 000240      NOP  
2257 027766 000240      NOP  
2258 027770 052777 010000 151760      BIS      #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER  
2259 027776 000240      NOP  
2260 030000 000240      NOP  
2261 030002 000240      NOP  
2262 030004 012737 000002 041726      MOV      #2,NARROW      ;SAVE EA BITS FOR ERROR TYPEOUT  
2263 030012 012737 000400 001124      MOV      #400,$GDDAT      ;LOAD EXPECTED DATA  
2264 030020 052737 000001 177572      BIS      #BIT0,@#SRO      ;ENABLE KT-11  
2265 030026 111037 001126      MOV      (R0),$BDDAT      ;SAVE ACTUAL DATA  
2266 030032 042737 000001 177572      BIC      #BIT0,@#SRO      ;DISABLE KT-11  
2267 030040 105037 001126      CLR      $BDDAT      ;MASK OFF LOW BYTE  
2268 030044 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA  
2269 030052 001401      BEQ      TST161      ;;BR IF SAME  
2270 030054 104020      ERROR 20      ;BIT 16 INPUT TO MATRIX MODE ADDER FAILED
```

```
2273          ;:*****  
(3)          ;*TEST 161      DETERMINE IF DIFLIN IS TO BE RUN (F)  
(3)          ;:*****  
(2) 030056 000004          TST161: SCOPE  
(1) 030060 012737 000001 001160      MOV      #1,$TIMES      ;;DO 1 ITERATION  
2274 030066 005737 050022          TST      RUNDIF        ;;TEST IF DIFLIN IS TO BE RUN  
2275 030072 001402          BEQ      $EOP          ;;BR IF NOT TO BE RUN  
2276 030074 000137 043546          JMP      DIFLIN        ;;JUMP AND RUN DIFLIN  
2277          .SBTTL  END OF PASS ROUTINE  
(1)          ;:*****  
(2)          ;*INCREMENT THE PASS NUMBER ($PASS)  
(1)          ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM  
(1)          ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)  
(1)          ;*IF THERES A MONITOR GO TO IT  
(1)          ;*IF THERE ISN'T JUMP TO LOGIC  
(1)          $EOP:  
(1) 030100          SCOPE  
(1) 030100 000004          CLR      $STNM        ;;ZERO THE TEST NUMBER  
(1) 030102 005037 001102          CLR      $TIMES      ;;ZERO THE NUMBER OF ITERATIONS  
(1) 030106 005037 001160          CLR      $PASS        ;;INCREMENT THE PASS NUMBER  
(1) 030112 005237 001202          INC      $PASS        ;;DON'T ALLOW A NEG. NUMBER  
(1) 030116 042737 100000 001202          BIC      #100000,$PASS  
(1) 030124 005327          DEC      (PC)+        ;;LOOP?  
(1) 030126 000001          $EOPCT: .WORD 1  
(1) 030130 003022          BGT      $DOAGN        ;;YES  
(1) 030132 012737          MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER  
(1) 030134 000001          $ENDCT: .WORD 1  
(1) 030136 030126          $EOPCT  
(1) 030140 104401 030205          TYPE    , $ENDMG      ;;TYPE 'END PASS #'  
(2) 030144 013746 001202          MOV      $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT  
(2) 030150 104405          TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN  
(1) 030152 104401 030202          TYPE    , $ENULL     ;;TYPE A NULL CHARACTER  
(1) 030156 013700 000042          $GET42: MOV    @#42,R0 ;;GET MONITOR ADDRESS  
(1) 030162 001405          BEQ      $DOAGN        ;;BRANCH IF NO MONITOR  
(1) 030164 000005          RESET   ;;CLEAR THE WORLD  
(1) 030166 004710          $ENDAD: JSR    PC,(R0) ;;GO TO MONITOR  
(1) 030170 000240          NOP     ;;SAVE ROOM  
(1) 030172 000240          NOP     ;;FOR  
(1) 030174 000240          NOP     ;;ACT11  
(1) 030176          $DOAGN:  
(1) 030176 000137          JMP      @(PC)+        ;;RETURN  
(1) 030200 003360          $RTNAD: .WORD  LOGIC  
(1) 030202 377 377 000          $ENULL: .BYTE  -1,-1,0 ;;NULL CHARACTER STRING  
(1) 030205 015 042412 042116          $ENDMG: .ASCIZ <15><12>/END PASS #/  
(1) 030212 050040 051501 020123  
(1) 030220 000043
```

```
2279 .SBTTL ERROR ASCII MESSAGES
2280
2281 030222 034115 031060 004466 EM1: .ASCIZ /M8026 NCV11 BUS ADDRESS TIMEOUT/
      030230 041516 030526 020061
      030236 052502 020123 042101
      030244 051104 051505 020123
      030252 044524 042515 052517
      030260 000124
2282 030262 034115 031060 004466 EM2: .ASCIZ /M8026 COMMAND-STATUS REGISTER ERROR/
      030270 047503 046515 047101
      030276 026504 052123 052101
      030304 051525 051040 043505
      030312 051511 042524 020122
      030320 051105 047522 000122
2283 030326 034115 031060 004466 EM3: .ASCIZ /M8026 SPECIAL FUNCTION REGISTER ERROR/
      030334 050123 041505 040511
      030342 020114 052506 041516
      030350 044524 047117 051040
      030356 043505 051511 042524
      030364 020122 051105 047522
      030372 000122
2284 030374 034115 031060 004466 EM4: .ASCIZ /M8026 WORD COUNT REGISTER ERROR/
      030402 047527 042122 041440
      030410 052517 052116 051040
      030416 043505 051511 042524
      030424 020122 051105 047522
      030432 000122
2285 030434 034115 031060 004466 EM5: .ASCIZ /M8026 BUS ADDRESS REGISTER ERROR/
      030442 052502 020123 042101
      030450 051104 051505 020123
      030456 042522 044507 052123
      030464 051105 042440 051122
      030472 051117 000
2286 030475 115 030070 033062 EM6: .ASCIZ /M8026 OFFSET REGISTER ERROR/
      030502 047411 043106 042523
      030510 020124 042522 044507
      030516 052123 051105 042440
      030524 051122 051117 000
2287 030531 115 030070 033062 EM7: .ASCIZ /M8026 DUAL REGISTER SELECTION ERROR/
      030536 042011 040525 020114
      030544 042522 044507 052123
      030552 051105 051440 046105
      030560 041505 044524 047117
      030566 042440 051122 051117
      030574 000
2288 030575 115 030070 033062 EM10: .ASCIZ /M8026-M8036 LOW 16 BIT Z COUNT ERROR/
      030602 046455 030070 033063
      030610 046011 053517 030440
      030616 020066 044502 020124
      030624 020132 047503 047125
      030632 020124 051105 047522
      030640 000122
2289 030642 034115 031060 026466 EM11: .ASCIZ /M8026-M8036 HIGH 16 BIT Z COUNT ERROR/
      030650 034115 031460 004466
      030656 044510 044107 030440
      030664 020066 044502 020124
```

	030672	020132	047503	047125		
	030700	020124	051105	047522		
	030706	000122				
2290	030710	034115	031060	004466	EM12:	.ASCIZ /M8026 Z COUNT STATUS ERROR/
	030716	020132	047503	047125		
	030724	020124	052123	052101		
	030732	051525	042440	051122		
	030740	051117	000			
2291	030743	115	030070	033062	EM13:	.ASCIZ /M8026 Z COUNT INTERRUPT ERROR/
	030750	055011	041440	052517		
	030756	052116	044440	052116		
	030764	051105	052522	052120		
	030772	042440	051122	051117		
	031000	000				
2292	031001	115	030070	033063	EM14:	.ASCIZ /M8036 JOYSTICK STATUS ERROR/
	031006	045011	054517	052123		
	031014	041511	020113	052123		
	031022	052101	051525	042440		
	031030	051122	051117	000		
2293	031035	115	030070	033063	EM15:	.ASCIZ /M8036 JOYSTICK DATA ERROR/
	031042	045011	054517	052123		
	031050	041511	020113	040504		
	031056	040524	042440	051122		
	031064	051117	000			
2294	031067	115	030070	033063	EM16:	.ASCIZ /M8036 DATA INCREMENT ERROR/
	031074	042011	052101	020101		
	031102	047111	051103	046505		
	031110	047105	020124	051105		
	031116	047522	000122			
2295	031122	034115	031460	004466	EM17:	.ASCIZ /M8036 DATA DECREMENT ERROR/
	031130	040504	040524	042040		
	031136	041505	042522	042515		
	031144	052116	042440	051122		
	031152	051117	000			
2296	031155	115	030070	033062	EM20:	.ASCIZ /M8026-M8036 MATRIX MODE ADDRESS MAKER DATA ERROR/
	031162	046455	030070	033063		
	031170	046411	052101	044522		
	031176	020130	047515	042504		
	031204	040440	042104	042522		
	031212	051523	046440	045501		
	031220	051105	042040	052101		
	031226	020101	051105	047522		
	031234	000122				
2297	031236	034115	031060	004466	EM21:	.ASCIZ /M8026 LIST MODE ADDRESS MAKER DATA ERROR/
	031244	044514	052123	046440		
	031252	042117	020105	042101		
	031260	051104	051505	020123		
	031266	040515	042513	020122		
	031274	040504	040524	042440		
	031302	051122	051117	000		
2298	031307	115	030070	033062	EM22:	.ASCIZ /M8026 LIST MODE TRANSFER BUS ADDRESS ERROR/
	031314	046011	051511	020124		
	031322	047515	042504	052040		
	031330	040522	051516	042506		
	031336	020122	052502	020123		
	031344	042101	051104	051505		

	031352	020123	051105	047522				
	031360	000122						
2299	031362	034115	031060	004466	EM23:	.ASCIIZ	/M8026	LIST MODE TRANSFER WORD COUNT ERROR/
	031370	044514	052123	046440				
	031376	042117	020105	051124				
	031404	047101	043123	051105				
	031412	053440	051117	020104				
	031420	047503	047125	020124				
	031426	051105	047522	000122				
2300	031434	034115	031060	004466	EM24:	.ASCIIZ	/M8026	LIST MODE TRANSFER OFFSET ERROR/
	031442	044514	052123	046440				
	031450	042117	020105	051124				
	031456	047101	043123	051105				
	031464	047440	043106	042523				
	031472	020124	051105	047522				
	031500	000122						
2301	031502	034115	031060	004466	EM25:	.ASCIIZ	/M8026	TIMEOUT STATUS ERROR/
	031510	044524	042515	052517				
	031516	020124	052123	052101				
	031524	051525	042440	051122				
	031532	051117	000					
2302	031535	115	030070	033062	EM26:	.ASCIIZ	/M8026	TIMEOUT INTERRUPT ERROR/
	031542	052011	046511	047505				
	031550	052125	044440	052116				
	031556	051105	052522	052120				
	031564	042440	051122	051117				
	031572	000						
2303	031573	115	030070	033062	EM27:	.ASCIIZ	/M8026	SET 'EVENT' OR 'TIME' DATA ERROR/
	031600	051411	052105	021040				
	031606	053105	047105	021124				
	031614	047440	020122	052042				
	031622	046511	021105	042040				
	031630	052101	020101	051105				
	031636	047522	000122					
2304	031642	034115	031060	004466	EM30:	.ASCIIZ	/M8026	CELL INCREMENT DATA ERROR/
	031650	042503	046114	044440				
	031656	041516	042522	042515				
	031664	052116	042040	052101				
	031672	020101	051105	047522				
	031700	000122						
2305	031702	034115	031060	004466	EM31:	.ASCIIZ	/M8026	CELL OVERFLOW STATUS ERROR/
	031710	042503	046114	047440				
	031716	042526	043122	047514				
	031724	020127	052123	052101				
	031732	051525	042440	051122				
	031740	051117	000					
2306	031743	115	030070	033062	EM32:	.ASCIIZ	/M8026	CELL OVERFLOW INTERRUPT ERROR/
	031750	041411	046105	020114				
	031756	053117	051105	046106				
	031764	053517	044440	052116				
	031772	051105	052522	052120				
	032000	042440	051122	051117				
	032006	000						
2307	032007	115	030070	033062	EM33:	.ASCIIZ	/M8026	MATRIX MODE ADDRESS MUX ERROR/
	032014	046411	052101	044522				
	032022	020130	047515	042504				

	032030	040440	042104	042522			
	032036	051523	046440	054125			
	032044	042440	051122	051117			
	032052	000					
2308	032053	115	030070	033062	FM34:	.ASCIIZ /M8026	'TESTX' FUNCTION ERROR/
	032060	021011	042524	052123			
	032066	021130	043040	047125			
	032074	052103	047511	020116			
	032102	051105	047522	000122			
2309	032110	034115	031060	004466	EM35:	.ASCIIZ /M8026	DATA ERROR WHEN TRANSFERING TO EXTENDED MEMORY/
	032116	040504	040524	042440			
	032124	051122	051117	053440			
	032132	042510	020116	051124			
	032140	047101	043123	051105			
	032146	047111	020107	047524			
	032154	042440	052130	047105			
	032162	042504	020104	042515			
	032170	047515	054522	000			
2310	032175	115	030070	033062	EM36:	.ASCIIZ /M8026	LIST MODE TRANSFER STATUS ERROR/
	032202	046011	051511	020124			
	032210	047515	042504	052040			
	032216	040522	051516	042506			
	032224	020122	052123	052101			
	032232	051525	042440	051122			
	032240	051117	000				
2311	032243	115	030070	033062	EM37:	.ASCIIZ /M8026	LIST MODE TRANSFER DATA ERROR/
	032250	046011	051511	020124			
	032256	047515	042504	052040			
	032264	040522	051516	042506			
	032272	020122	040504	040524			
	032300	042440	051122	051117			
	032306	000					
2312	032307	112	046525	042520	EM40:	.ASCIIZ /JUMPER-M8026-M7952	'EVENT' OR 'TIME' MARK ERROR/
	032314	026522	034115	031060			
	032322	026466	033515	032471			
	032330	004462	042442	042526			
	032336	052116	020042	051117			
	032344	021040	044524	042515			
	032352	020042	040515	045522			
	032360	042440	051122	051117			
	032366	000					
2313	032367	115	034467	031065	EM41:	.ASCIIZ /M7952	CLOCK BUS ADDRESS TIMEOUT/
	032374	041411	047514	045503			
	032402	041040	051525	040440			
	032410	042104	042522	051523			
	032416	052040	046511	047505			
	032424	052125	000				
2314	032427	115	031070	033461	EM42:	.ASCIIZ /M8217	INCORRECT INTERRUPT LEVEL/
	032434	044411	041516	051117			
	032442	042522	052103	044440			
	032450	052116	051105	052522			
	032456	052120	046040	053105			
	032464	046105	000				
2315							
2316	032467	105	051122	041520	DH1:	.ASCIIZ /ERRPC	ADDR/
	032474	040411	042104	000122			

2317	032502	051105	050122	004503	DH2:	.ASCII	/ERRPC	ADDR	GOOD	BAD/	
	032510	042101	051104	043411							
	032516	047517	004504	040502							
	032524	000104									
2318	032526	051105	050122	004503	DH3:	.ASCII	/ERRPC	ADDR	GOOD	BAD	BADADR/
	032534	042101	051104	043411							
	032542	047517	004504	040502							
	032550	004504	040502	040504							
	032556	051104	000								
2319	032561	105	051122	041520	DH4:	.ASCII	/ERRPC	ADDR	GOOD	BAD	EA ADR BADADR/
	032566	040411	042104	004522							
	032574	047507	042117	041011							
	032602	042101	042411	020101							
	032610	042101	004522	040502							
	032616	040504	051104	000							
2320	032623	015	012	007	OUTRNG:	.BYTE	15,12,7				
2321	032626	047516	021040	021132		.ASCII	/NO "Z" PULSES OR /				
	032634	050040	046125	042523							
	032642	020123	051117	040							
2322	032647	111	050116	052125		.ASCII	/INPUT VOLTAGE OUT OF RANGE ON CHANNEL #/				
	032654	053040	046117	040524							
	032662	042507	047440	052125							
	032670	047440	020106	040522							
	032676	043516	020105	047117							
	032704	041440	040510	047116							
	032712	046105	021440								
2323	032716	060	015	012	OUTCHN:	.BYTE	60,15,12,7,0				
	032721	007	000								
2324		032724				.EVEN					
2325	032724	001116	001250	000000	DT1:	.WORD	\$ERRPC,\$BASE,0				
2326	032732	001116	001250	001124	DT2:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,0				
	032740	001126	000000								
2327	032744	001116	001250	001124	DT3:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,\$BDADR,0				
	032752	001126	001122	000000							
2328	032760	001116	001250	001124	DT4:	.WORD	\$ERRPC,\$BASE,\$GDDAT,\$BDDAT,NARROW,\$BDADR,0				
	032766	001126	041726	001122							
	032774	000000									
2329	032776	001116	001254	000000	DT5:	.WORD	\$ERRPC,\$CDW1,0				
2330	033004	000000			DF0:	.WORD	0				

```

2335 .SBTTL TTY INPUT ROUTINE
(1)
(2)
(1)
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 033006 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
(1) 033014 001074 BNE 15$ ;; BRANCH IF NO
(1) 033016 105777 146122 TSTB @STKS ;; CHAR THERE?
(1) 033022 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
(1) 033024 117746 146116 MOVB @STKB,-(SP) ;; SAVE THE CHAR
(1) 033030 042716 177600 BIC #^C177,(SP) ;; STRIP-OFF THE ASCII
(1) 033034 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
(1) 033040 001062 BNE 15$ ;; NO, RETURN TO USER
(1) 033042 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
(1) 033050 001456 BEQ 15$ ;; BRANCH IF YES
(1) 033052 104401 033533 $GTSWR: TYPE $CNTLG ;; ECHO THE CONTROL-G (^G)
(1) 033056 104401 033540 TYPE $MSWR ;; TYPE CURRENT CONTENTS
(2) 033062 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
(2) 033066 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 033070 104401 033551 TYPE $MNEW ;; PROMPT FOR NEW SWR
(1) 033074 005046 19$: CLR -(SP) ;; CLEAR COUNTER
(1) 033076 005046 CLR -(SP) ;; THE NEW SWR
(1) 033100 105777 146040 7$: TSTB @STKS ;; CHAR THERE?
(1) 033104 100375 BPL 7$ ;; IF NOT TRY AGAIN
(1) 033106 117746 146034 MOVB @STKB,-(SP) ;; PICK UP CHAR
(1) 033112 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 033116 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
(1) 033122 001005 BNE 10$ ;; BRANCH IF NOT
(1) 033124 104401 033526 TYPE $CNTLU ;; YES, ECHO CONTROL-U (^U)
(1) 033130 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
(1) 033134 000757 BR 19$ ;; LET'S TRY IT AGAIN
(1)
(1)
(1) 033136 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
(1) 033142 001022 BNE 16$ ;; BRANCH IF NO
(1) 033144 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
(1) 033150 001403 BEQ 11$ ;; BRANCH IF YES
(1) 033152 016677 000002 145760 MOV 2(SP),@SWR ;; SAVE NEW SWR
(1) 033160 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
(1) 033164 104401 001171 14$: TYPE $CRLF ;; ECHO <CR> AND <LF>
(1) 033170 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
(1) 033176 001003 BNE 15$ ;; BRANCH IF NOT
(1) 033200 012777 000100 145736 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
(1) 033206 000002 15$: RTI ;; RETURN
(1) 033210 004737 034606 16$: JSR PC,$TYPEC ;; ECHO CHAR
(1) 033214 021627 000060 CMP (SP),#60 ;; CHAR < 0?
  
```

```
(1) 033220 002420          BLT      18$          ;;BRANCH IF YES
(1) 033222 021627 000067    CMP      (SP),#67     ;;CHAR > 7?
(1) 033226 003015          BGT      18$          ;;BRANCH IF YES
(1) 033230 042726 000060    BIC      #60,(SP)+    ;;STRIP-OFF ASCII
(1) 033234 005766 000002    TST      2(SP)        ;;IS THIS THE FIRST CHAR
(1) 033240 001403          BEQ      17$          ;;BRANCH IF YES
(1) 033242 006316          ASL      (SP)         ;;NO, SHIFT PRESENT
(1) 033244 006316          ASL      (SP)         ;;CHAR OVER TO MAKE
(1) 033246 006316          ASL      (SP)         ;;ROOM FOR NEW ONE.
(1) 033250 005266 000002    17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
(1) 033254 056616 177776    BIS      -2(SP),(SP) ;;SET IN NEW CHAR
(1) 033260 000707          BR       7$           ;;GET THE NEXT ONE
(1) 033262 104401 001170    18$: TYPE  $QUES      ;;TYPE ?<CR><LF>
(1) 033266 000720          BR       20$          ;;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) $RDCHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC
(1) 033270 011646          MOV      4(SP),2(SP) ;;SAVE THE PS
(1) 033272 016666 000004 000002 1$: TSTB     @STKS      ;;WAIT FOR
(1) 033300 105777 145640          BPL      1$           ;;A CHARACTER
(1) 033304 100375          MOVB     @STKB,4(SP) ;;READ THE TTY
(1) 033306 117766 145634 000004 BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 033314 042766 177600 000004 CMP      4(SP),#23    ;;IS IT A CONTROL-S?
(1) 033322 026627 000004 000023 BNE      3$           ;;BRANCH IF NO
(1) 033330 001013          TSTB     @STKS      ;;WAIT FOR A CHARACTER
(1) 033332 105777 145606          BPL      2$           ;;LOOP UNTIL ITS THERE
(1) 033336 100375          MOVB     @STKB, -(SP) ;;GET CHARACTER
(1) 033340 117746 145602          BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 033344 042716 177600          CMP      (SP)+,#21   ;;IS IT A CONTROL-Q?
(1) 033350 022627 000021          BNE      2$           ;;IF NOT DISCARD IT
(1) 033354 001366          BR       1$           ;;YES, RESUME
(1) 033356 000750          CMP      4(SP),#140  ;;IS IT UPPER CASE?
(1) 033360 026627 000004 000140 3$: BLT      4$           ;;BRANCH IF YES
(1) 033366 002407          CMP      4(SP),#175  ;;IS IT A SPECIAL CHAR?
(1) 033370 026627 000004 000175 BGT      4$           ;;BRANCH IF YES
(1) 033376 003003          BIC      #40,4(SP)   ;;MAKE IT UPPER CASE
(1) 033400 042766 000040 000004 4$: RTI          ;;GO BACK TO USER
(1) 033406 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) 033410 010346          MOV      #TTYIN,R3   ;;GET ADDRESS
(1) 033412 012703 033516          CMP      #TTYIN+8.,R3 ;;BUFFER FULL?
(1) 033416 022703 033526
```

(1)	033422	101405			BLOS	4\$::BR IF YES
(1)	033424	104410			RDCHR			::GO READ ONE CHARACTER FROM THE TTY
(1)	033426	112613			MOVB	(SP)+,(R3)		::GET CHARACTER
(1)	033430	122713	000177	10\$:	CMPB	#177,(R3)		::IS IT A RUBOUT
(1)	033434	001003			BNE	3\$::SKIP IF NOT
(1)	033436	104401	001170	4\$:	TYPE	,SQUES		::TYPE A '?'
(1)	033442	000763			BR	1\$::CLEAR THE BUFFER AND LOOP
(1)	033444	111337	0335^4	3\$:	MOVB	(R3),9\$::ECHO THE CHARACTER
(1)	033450	104401	033514		TYPE	,9\$		
(1)	033454	122723	000015		CMPB	#15,(R3)+		::CHECK FOR RETURN
(1)	033460	001356			BNE	2\$::LOOP IF NOT RETURN
(1)	033462	105063	177777		CLRB	-1(R3)		::CLEAR RETURN (THE 15)
(1)	033466	104401	001172		TYPE	,SLF		::TYPE A LINE FEED
(1)	033472	012603			MOV	(SP)+,R3		::RESTORE R3
(1)	033474	011646			MOV	(SP),-(SP)		::ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	033476	016666	000004	000002	MOV	4(SP),2(SP)		:: FIRST ASCII CHARACTER ON IT
(1)	033504	012766	033516	000004	MOV	#\$TTYIN,4(SP)		
(1)	033512	000002			RTI			::RETURN
(1)	033514	000		9\$:	.BYTE	0		::STORAGE FOR ASCII CHAR. TO TYPE
(1)	033515	000			.BYTE	0		::TERMINATOR
(1)	033516	000010			\$TTYIN:	.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
(1)	033526	052536	005015	000	\$CNTLU:	.ASCIZ	/^U/<15><12>	::CONTROL 'U'
(1)	033533	136	006507	000012	\$CNTLG:	.ASCIZ	/^G/<15><12>	::CONTROL 'G'
(1)	033540	005015	053523	020122	\$MSWR:	.ASCIZ	<15><12>/SWR - /	
(1)	033546	020075	000					
(1)	033551	040	047040	053505	\$MNEW:	.ASCIZ	/ NEW - /	
(1)	033556	036440	000040					

```
2337 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 033562 011646
(1) 033564 016666 000004 000002
(3) 033572 010046
(3) 033574 010146
(3) 033576 010246
(1) 033600 104411
(1) 033602 012600
(1) 033604 005001
(1) 033606 005002
(1) 033610 112046
(1) 033612 001412
(1) 033614 006301
(1) 033616 006102
(1) 033620 006301
(1) 033622 006102
(1) 033624 006301
(1) 033626 006102
(1) 033630 042716 177770
(1) 033634 062601
(1) 033636 000764
(1) 033640 005726
(1) 033642 010166 000012
(1) 033646 010237 033662
(3) 033652 012602
(3) 033654 012601
(3) 033656 012600
(1) 033660 000002
(1) 033662 000000

$RDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
MOV 4(SP),2(SP) ;;INPUT NUMBER
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
1$: RDLIN ;;READ AN ASCII LINE
MOV (SP)+,R0 ;;GET ADDRESS OF 1ST CHARACTER
CLR R1 ;;CLEAR DATA WORD
CLR R2
2$: MOVB (R0)+,-(SP) ;;PICKUP THIS CHARACTER
BEQ 3$ ;;IF ZERO GET OUT
ASL R1 ;;*2
ROL R2
ASL R1 ;;*4
ROL R2
ASL R1 ;;*8
ROL R2
BIC #^C7,(SP) ;;STRIP THE ASCII JUNK
ADD (SP)+,R1 ;;ADD IN THIS DIGIT
BR 2$ ;;LOOP
3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
MOV R1,12(SP) ;;SAVE THE RESULT
MOV R2,$HIOCT
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI ;;RETURN
$HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
```

2339

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
(1) (1)
(2)
(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) :*   MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1) :*   TYPDS   ;;GO TO THE ROUTINE
(1)
(1) $TYPDS:
(3) 033664 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 033666 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 033670 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 033672 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 033674 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(1) 033676 012746 020200 MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
(1) 033702 016605 000020 MOV 20(SP),R5 ;;GET THE INPUT NUMBER
(1) 033706 100004 BPL 1$ ;;BR IF INPUT IS POS.
(1) 033710 005405 NEG R5 ;;MAKE THE BINARY NUMBER POS.
(1) 033712 112766 000055 000001 MOVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
(1) 033720 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
(1) 033722 012703 034100 MOV #SDBLK,R3 ;;SETUP THE OUTPUT POINTER
(1) 033726 112723 000040 MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
(1) 033732 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
(1) 033734 016001 034070 MOV $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 033740 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
(1) 033742 002402 BLT 4$ ;;BR IF DONE
(1) 033744 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
(1) 033746 000774 BR 3$
(1) 033750 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
(1) 033752 005702 TST R2 ;;CHECK IF BCD DIGIT-0
(1) 033754 001002 BNE 5$ ;;FALL THROUGH IF 0
(1) 033756 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
(1) 033760 100407 BMI 7$ ;;BR IF YES
(1) 033762 106316 5$: ASLB (SP) ;;MSD?
(1) 033764 103003 BCC 6$ ;;BR IF NO
(1) 033766 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 033774 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
(1) 034000 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 034004 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 034006 005720 TST (R0)+ ;;JUST INCREMENTING
(1) 034010 020027 000010 CMP R0,#10 ;;CHECK THE TABLE INDEX
(1) 034014 002746 BLT 2$ ;;GO DO THE NEXT DIGIT
(1) 034016 003002 BGT 8$ ;;GO TO EXIT
(1) 034020 010502 MOV R5,R2 ;;GET THE LSD
(1) 034022 000764 BR 6$ ;;GO CHANGE TO ASCII
(1) 034024 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 034026 100003 BPL 9$ ;;BR IF NO
(1) 034030 116663 177777 177776 MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 034036 105013 9$: CLRB (R3) ;;SET THE TERMINATOR
(3) 034040 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 034042 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 034044 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
```



```

(3) 034046 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 034050 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 034052 104401 034100  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
(1) 034056 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
(1) 034064 012616      MOV      (SP)+,(SP)
(1) 034066 000002      RTI                      ;;RETURN TO USER
(1) 034070 023420      $DTBL: 10000.
(1) 034072 001750      1000.
(1) 034074 000144      100.
(1) 034076 000012      10.
(1) 034100 000004      $DBLK: .BLKW 4
2340 .SBTTL SCOPE HANDLER ROUTINE

(1)
(2)
(1)
(1) *****
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW14=1 LOOP ON TEST
(1) *SW11=1 INHIBIT ITERATIONS
(1) *SW09=1 LOOP ON ERROR
(1) *SW08=1 LOOP ON TEST IN SWR<7:0>
(1) *CALL
(1) * SCOPE ;;SCOPE=IOT

(1)
(1) $SCOPE:
(1) 034110 104407      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(2) 034112 004737 045676 145014  JSR      PC,CTRLCG      ;;TEST FOR CTRL C OR G
(1) 034116 032777 040000 145014  1$: BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
(1) 034124 001114      BNE      $OVER      ;;YES IF SW14=1
(1)
(1) :#####START OF CODE FOR THE XOR TESTER#####
(1) 034126 000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)
(1) 034130 013746 000004      MOV      @WERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 034134 012737 034154 000004  MOV      #5$,@WERRVEC  ;;SET FOR TIMEOUT
(1) 034142 005737 177060      TST      @#177060      ;;TIME OUT ON YOR?
(1) 034146 012637 000004      MOV      (SP)+,@WERRVEC ;;RESTORE THE ERROR VECTOR
(1) 034152 000463      BR      $SVLAD      ;;GO TO THE NEXT TEST
(1) 034154 022626      5$: CMP      (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
(1) 034156 012637 000004      MOV      (SP)+,@WERRVEC ;;RESTORE THE ERROR VECTOR
(1) 034162 000423      BR      7$      ;;LOOP ON THE PRESENT TEST
(1) 034164
(1) 034164 032777 000400 144746  6$:;#####END OF CODE FOR THE XOR TESTER#####
(1) 034172 001404      BIT      #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
(1) 034174 127737 144740 001102  BEQ      2$      ;;BR IF NO
(1) 034202 001465      CMPB     @SWR,$TSTNM      ;;ON THE RIGHT TEST? SWR<7:0>
(1) 034204 105737 001103  2$: BEQ      $OVER      ;;BR IF YES
(1) 034210 001421      TSTB     $ERFLG      ;;HAS AN ERROR OCCURRED?
(1) 034212 123737 001115 001103  BEQ      3$      ;;BR IF NO
(1) 034220 101015      CMPB     $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 034222 032777 001000 144710  BHI      3$      ;;BR IF NO
(1) 034230 001404      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
(1) 034232 013737 001110 001106  7$: BEQ      4$      ;;BR IF NO
(1) 034240 000446      MOV      $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 034242 105037 001103  4$: BR      $OVER
(1) 034246 005037 001160      CLRB     $ERFLG      ;;ZERO THE ERROR FLAG
(1) 034246 005037 001160      CLR      $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
  
```

```

(1) 034252 000415          BR      1$          ;;ESCAPE TO THE NEXT TEST
(1) 034254 032777 004000 144656 3$:  BIT      #BIT11,@SWR  ;;INHIBIT ITERATIONS?
(1) 034262 001011          BNE     1$          ;;BR IF YES
(1) 034264 005737 001202          TST     $PASS      ;;IF FIRST PASS OF PROGRAM
(1) 034270 001406          BEQ     1$          ;;INHIBIT ITERATIONS
(1) 034272 005237 001104          INC     $ICNT      ;;INCREMENT ITERATION COUNT
(1) 034276 023737 001160 001104  CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 034304 002024          BGE     $OVER      ;;BR IF MORE ITERATION REQUIRED
(1) 034306 012737 000001 001104 1$:  MOV     #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
(1) 034314 013737 034372 001160  MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 034322 105237 001102          $SVLAD: INCB    $STNM   ;;COUNT TEST NUMBERS
(1) 034326 113737 001102 001200  MOVB   $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 034334 011637 001106          MOV     (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 034340 011637 001110          MOV     (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 034344 005037 001162          CLR     $ESCAPE   ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 034350 112737 000001 001115  MOVB   #1,$ERMAX   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 034356 013777 001102 144556 $OVER:  MOV     $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 034364 013716 001106          MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 034370 000002          RTI                    ;;FIXES PS
(1) 034372 003720          $MXCNT: 2000.         ;;MAX. NUMBER OF ITERATIONS
2341 .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) 034374 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(1) 034400 100002          BPL     1$          ;;BR IF YES
(1) 034402 000000          HALT                    ;;HALT HERE IF NO TERMINAL
(1) 034404 000430          BR      3$          ;;LEAVE
(1) 034406 010046          1$:  MOV     R0,-(SP)   ;;SAVE R0
(1) 034410 017600 000002          MOV     @2(SP),R0    ;;GET ADDRESS OF ASCIZ STRING
(1) 034414 122737 000001 001214  CMPB   #APTENV,$ENV  ;;RUNNING IN APT MODE
(1) 034422 001011          BNE     62$         ;;NO,GO CHECK FOR APT CONSOLE
(1) 034424 132737 000100 001215  BITB   #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 034432 001405          BEQ     62$         ;;NO,GO CHECK FOR CONSOLE
(1) 034434 010037 034444          MOV     R0,61$      ;;SETUP MESSAGE ADDRESS FOR APT
(1) 034440 004737 035446          JSR    PC,$ATY3    ;;SPOOL MESSAGE TO APT
(1) 034444 000000          61$: .WORD 0        ;;MESSAGE ADDRESS
(1) 034446 132737 000040 001215  62$:  BITB   #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 034454 001003          BNE     60$         ;;YES,SKIP TYPE OUT
(1) 034456 112046          2$:  MOVB   (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 034460 001005          BNE     4$          ;;BR IF IT ISN'T THE TERMINATOR
(1) 034462 005726          TST    (SP)+       ;;IF TERMINATOR POP IT OFF THE STACK
(1) 034464 012600          60$:  MOV     (SP)+,R0  ;;RESTORE R0

```

```

(1) 034466 062716 000002      3$:  ADD      #2,(SP)          ;;ADJUST RETURN PC
(1) 034472 000002              RTI                      ;;RETURN
(1) 034474 122716 000011      4$:  CMPB     #HT,(SP)          ;;BRANCH IF <HT>
(1) 034500 001430              BEQ      8$
(1) 034502 122716 000200      CMPB     #CRLF,(SP)        ;;BRANCH IF NOT <CRLF>
(1) 034506 001006              BNE      5$
(1) 034510 005726              TST      (SP)+            ;;POP <CR><LF> EQUIV
(1) 034512 104401              TYPE
(1) 034514 001171              $CRLF                    ;;TYPE A CR AND LF
(1) 034516 105037 034652      CLRB     $CHARCNT         ;;CLEAR CHARACTER COUNT
(1) 034522 000755              BR       2$               ;;GET NEXT CHARACTER
(1) 034524 004737 034606      5$:  JSR      PC,$TYPEPC     ;;GO TYPE THIS CHARACTER
(1) 034530 123726 001156      6$:  CMPB     $FILLC,(SP)+   ;;IS IT TIME FOR FILLER CHARS.?
(1) 034534 001350              BNE      2$               ;;IF NO GO GET NEXT CHAR.
(1) 034536 013746 001154      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
(1)                                ;;AND THE NULL CHAR.
(1) 034542 105366 000001      7$:  DECB     1(SP)          ;;DOES A NULL NEED TO BE TYPED?
(1) 034546 002770              BLT      6$               ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 034550 004737 034606      JSR      PC,$TYPEPC     ;;GO TYPE A NULL
(1) 034554 105337 034652      DECB     $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1) 034560 000770              BR       7$               ;;LOOP
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1) 034562 112716 000040      8$:  MOVB     #' ,(SP)      ;;REPLACE TAB WITH SPACE
(1) 034566 004737 034606      9$:  JSR      PC,$TYPEPC     ;;TYPE A SPACE
(1) 034572 132737 000007 034652 BITB     #7,$CHARCNT      ;;BRANCH IF NOT AT
(1) 034600 001372              BNE      9$               ;;TAB STOP
(1) 034602 005726              TST      (SP)+            ;;POP SPACE OFF STACK
(1) 034604 000724              BR       2$               ;;GET NEXT CHARACTER
(1) 034606 105777 144336      $TYPEPC: TSTB    @STPS     ;;WAIT UNTIL PRINTER IS READY
(1) 034612 100375              BPL     $TYPEPC
(1) 034614 116677 000002 144330 MOVB     2(SP),@STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 034622 122766 000015 000002 CMPB     #CR,2(SP)        ;;IS CHARACTER A CARRIAGE RETURN?
(1) 034630 001003              BNE      1$               ;;BRANCH IF NO
(1) 034632 105037 034652      CLRB     $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1) 034636 000406              BR       $TYPEPC         ;;EXIT
(1) 034640 122766 000012 000002 1$:  CMPB     #LF,2(SP)        ;;IS CHARACTER A LINE FEED?
(1) 034646 001402              BEQ     $TYPEPC          ;;BRANCH IF YES
(1) 034650 105227              INCB    (PC)+            ;;COUNT THE CHARACTER
(1) 034652 000000      $CHARCNT: .WORD 0        ;;CHARACTER COUNT STORAGE
(1) 034654 000207      $TYPEPC: RTS      PC
(1)

```



```

(1) 035020 005204          4$: INC R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 035022 052703 000060  BIS #'0,R3        ;;MAKE THIS DIGIT ASCII
(1) 035026 052703 000040  5$: BIS #' ,R3      ;;MAKE ASCII IF NOT ALREADY
(1) 035032 110337 035076  MOVB R3,8$        ;;SAVE FOR TYPING
(1) 035036 104401 035076  TYPE ,8$         ;;GO TYPE THIS DIGIT
(1) 035042 105337 035100  7$: DECB $OCNT   ;;COUNT BY 1
(1) 035046 003347        BGT 2$           ;;BR IF MORE TO DO
(1) 035050 002402        BLT 6$           ;;BR IF DONE
(1) 035052 005204        INC R4           ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 035054 000744        BR 2$           ;;GO DO THE LAST DIGIT
(1) 035056 012605 6$: MOV (SP)+,R5      ;;RESTORE R5
(1) 035060 012604        MOV (SP)+,R4      ;;RESTORE R4
(1) 035062 012603        MOV (SP)+,R3      ;;RESTORE R3
(1) 035064 016666 000002 000004  MOV 2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
(1) 035072 012616        MOV (SP)+,(SP)
(1) 035074 000002        RTI                ;;RETURN
(1) 035076 000          8$: .BYTE 0          ;;STORAGE FOR ASCII DIGIT
(1) 035077 000          .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
(1) 035100 000          $OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
(1) 035101 000          $OFILL: .BYTE 0     ;;ZERO FILL SWITCH
(1) 035102 000000        $OMODE: .WORD 0    ;;NUMBER OF DIGITS TO TYPE

```

2344
2345

.SBTTL ERROR HANDLER ROUTINE

```

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

```

```

(1) 035104          $ERROR:
(1) 035104 104407        CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(1) 035106 105237 001103  7$: INCB $ERFLG      ;;SET THE ERROR FLAG
(1) 035112 001775        BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
(1) 035114 013777 001102 144020  MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 035122 032777 002000 144010  BIT #BIT10,@SWR     ;;BELL ON ERROR?
(1) 035130 001402        BEQ 1$          ;;NO - SKIP
(1) 035132 104401 001164        TYPE ,SBELL      ;;RING BELL
(1) 035136 005237 001112  1$: INC $ERTTL      ;;COUNT THE NUMBER OF ERRORS
(1) 035142 011637 001116        MOV (SP),$ERRPC   ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 035146 162737 000002 001116  SUB #2,$ERRPC
(1) 035154 117737 143736 001114  MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 035162 032777 020000 143750  BIT #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1) 035170 001004        BNE 20$         ;;SKIP TYPEOUTS
(1) 035172 004737 035304        JSR PC,$ERRTYP   ;;GO TO USER ERROR ROUTINE
(1) 035176 104401 001171        TYPE ,$CRLF
(1) 035202          20$: CMPB #APTENV,$ENV  ;;RUNNING IN APT MODE
(1) 035210 001007        BNE 2$          ;;NO,SKIP APT ERROR REPORT
(1) 035212 113737 001114 035224  MOVB $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER

```

```

(1) 035220 004737 035456      JSR    PC,$ATY4      ;;REPORT FATAL ERROR TO APT
(1) 035224      000          21$:  .BYTE    0
(1) 035225      000          .BYTE    0
(1) 035226 000777          BR      22$          ;;APT ERROR LOOP
(1) 035230 005777 143704    22$:  TST     @SWR      ;;HALT ON ERROR
(1) 035234 100002          BPL     3$           ;;SKIP IF CONTINUE
(1) 035236 000000          HALT                    ;;HALT ON ERROR!
(1) 035240 104407          CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
(1) 035242 032777 001000 143670 3$:  BIT     #BIT09,@SWR   ;;LOOP ON ERROR SWITCH SET?
(1) 035250 001402          BEQ     4$           ;;BR IF NO
(1) 035252 013716 001110          MOV     $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
(1) 035256 005737 001162    4$:  TST     $ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
(1) 035262 001402          BEQ     5$           ;;BR IF NONE
(1) 035264 013716 001162    5$:  MOV     $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 035270          .
(1) 035270 022737 030166 000042 6$:  CMP     #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 035276 001001          BNE                    ;;BRANCH IF NO
(1) 035300 000000          HALT                    ;;YES
(1) 035302          .
(1) 035302 000002          RTI                     ;;RETURN
  
```

2346
2347

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 035304
(1) 035304 104401 001171      TYPE   $CRLF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035310 010046          MOV    R0,-(SP)       ;;SAVE R0
(1) 035312 005000          CLR    R0             ;;PICKUP THE ITEM INDEX
(1) 035314 153700 001114    BISB  @#$ITEMB,R0
(1) 035320 001004          BNE    1$            ;;IF ITEM NUMBER IS ZERO, JUST
(2) 035322 013746 001116    MOV    $ERRPC,-(SP)  ;;TYPE THE PC OF THE ERROR
(2)
(2) 035326 104402          TYPOC                    ;;SAVE $ERRPC FOR TYPEOUT
(1) 035330 000426          BR      6$            ;;ERROR ADDRESS
(1) 035332 005300    1$:  DEC    R0             ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 035334 006300          ASL   R0             ;;GET OUT
(1) 035336 006300          ASL   R0             ;;ADJUST THE INDEX SO THAT IT WILL
(1) 035340 006300          ASL   R0             ;;WORK FOR THE ERROR TABLE
(1) 035342 062700 001256    ADD    #SERRTB,R0    ;;FORM TABLE POINTER
(1) 035346 012037 035356    MOV    (R0)+,2$      ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 035352 001404          BEQ    3$            ;;SKIP TYPEOUT IF NO POINTER
(1) 035354 104401          TYPE                    ;;TYPE THE 'ERROR MESSAGE'
(1) 035356 000000    2$:  .WORD   0           ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 035360 104401 001171          TYPE   $CRLF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035364 012037 035374    3$:  MOV    (R0)+,4$      ;;PICKUP 'DATA HEADER' POINTER
(1) 035370 001404          BEQ    5$            ;;SKIP TYPEOUT IF 0
(1) 035372 104401          TYPE                    ;;TYPE THE 'DATA HEADER'
(1) 035374 000000    4$:  .WORD   0           ;;'DATA HEADER' POINTER GOES HERE
(1) 035376 104401 001171          TYPE   $CRLF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 035402 011000    5$:  MOV    (R0),R0       ;;PICKUP 'DATA TABLE' POINTER
(1) 035404 001004          BNE    7$            ;;GO TYPE THE DATA
  
```

CZNCB NCV11 DIAGNOSTIC
CZNCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 73-3
ERROR MESSAGE TYPEOUT ROUTINE

SEQ 0110

(1)	035406	012600		6\$:	MOV	(SP)+,R0	::RESTORE R0
(1)	035410	104401	001171		TYPE	,SCLF	::'CARRIAGE RETURN' & 'LINE FEED'
(1)	035414	000207			RTS	PC	::RETURN
(1)	035416			7\$:			
(2)	035416	013046			MOV	@(R0)+,-(SP)	::SAVE @(R0)+ FOR TYPEOUT
(2)	035420	104402			TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)	035422	005710			TST	(R0)	::IS THERE ANOTHER NUMBER?
(1)	035424	001770			BEQ	6\$::BR IF NO
(1)	035426	104401	035434		TYPE	8\$::TYPE TWO(2) SPACES
(1)	035432	000771			BR	7\$::LOOP
(1)	035434	020040	000	8\$:	.ASCIZ	/ /	::TWO(2) SPACES
(1)		035440			.EVEN		

2349

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1)
(2)
(1) 035440 112737 000001 035704 $ATY1: MOVB #1,$FFLG ;;TO REPORT FA*AL ERROR
(1) 035446 112737 000001 035702 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) 035454 000403 BR $ATYC
(1) 035456 112737 000001 035704 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 035464 $ATYC:
(3) 035464 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 035466 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 035470 105737 035702 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 035474 001450 BEQ 5$ ;;IF NOT: BR
(1) 035476 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 035504 001031 BNE 3$ ;;IF NOT: BR
(1) 035506 132737 000100 001215 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 035514 001425 BEQ 3$ ;;IF NOT: BR
(1) 035516 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 035522 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 035530 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 035534 001375 BNE 1$ ;;IF NOT: WAIT
(1) 035536 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 035542 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 035544 001376 BNE 2$
(1) 035546 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 035552 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS
(1) 035554 010037 001212 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
(1) 035560 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 035566 000413 BR 5$
(1) 035570 017637 000004 035614 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 035576 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 035604 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 035610 004737 034374 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 035614 000000 4$: .WORD 0
(1) 035616 5$:
(1) 035616 105737 035704 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 035622 001416 BEQ 12$ ;;IF NOT: BR
(1) 035624 005737 001214 TST $ENV ;;RUNNING UNDER APT?
(1) 035630 001413 BEQ 12$ ;;IF NOT: BR
(1) 035632 005737 001174 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) 035636 001375 BNE 11$ ;;IF NOT: WAIT
(1) 035640 017637 000004 001176 MOV @4(SP),$FATAL ;;GET ERROR #
(1) 035646 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 035654 005237 001174 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
(1) 035660 105037 035704 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) 035664 105037 035703 CLRB $LFLG ;;CLEAR LOG FLAG
(1) 035670 105037 035702 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) 035674 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 035676 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 035700 000207 RTS PC ;;RETURN
(1) 035702 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 035703 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 035704 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 035706 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100

```



```

(1)          000040          APTCSUP=040
2350          .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)          ::*****
(1)          :POWER DOWN ROUTINE
(1) 035706 012737 036046 000024 $PWRDN: MOV    $ILLUP,@PWRVEC  ;;SET FOR FAST UP
(1) 035714 012737 000340 000026      MOV    #340,@PWRVEC+2  ;;PRIO:7
(3) 035722 010046          MOV    R0,-(SP)        ;;PUSH R0 ON STACK
(3) 035724 010146          MOV    R1,-(SP)        ;;PUSH R1 ON STACK
(3) 035726 010246          MOV    R2,-(SP)        ;;PUSH R2 ON STACK
(3) 035730 010346          MOV    R3,-(SP)        ;;PUSH R3 ON STACK
(3) 035732 010446          MOV    R4,-(SP)        ;;PUSH R4 ON STACK
(3) 035734 010546          MOV    R5,-(SP)        ;;PUSH R5 ON STACK
(3) 035736 017746 143176      MOV    @SWR,-(SP)     ;;PUSH @SWR ON STACK
(1) 035742 010637 036052      MOV    SP,$SAVR6     ;;SAVE SP
(1) 035746 012737 035760 000024      MOV    $PWRUP,@PWRVEC ;;SET UP VECTOR
(1) 035754 000000          HALT
(1) 035756 000776          BR     .-2           ;;HANG UP
(1)
(2)          ::*****
(1)          :POWER UP ROUTINE
(1) 035760 012737 036046 000024 $PWRUP: MOV    $ILLUP,@PWRVEC  ;;SET FOR FAST DOWN
(1) 035766 013706 036052      MOV    $SAVR6,SP     ;;GET SP
(1) 035772 005037 036052      CLR    $SAVR6        ;;WAIT LOOP FOR THE ITY
(1) 035776 005237 036052      1$:  INC    $SAVR6     ;;WAIT FOR THE INC
(1) 036002 001375          BNE    1$            ;;OF WORD
(3) 036004 012677 143130      MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
(3) 036010 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
(3) 036012 012604          MOV    (SP)+,R4     ;;POP STACK INTO R4
(3) 036014 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
(3) 036016 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
(3) 036020 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
(3) 036022 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
(1) 036024 012737 035706 000024      MOV    $PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 036032 012737 000340 000026      MOV    #340,@PWRVEC+2 ;;PRIO:7
(1) 036040 104401          TYPE                ;;REPORT THE POWER FAILURE
(1) 036042 036054          $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
(1) 036044 000002          RTI
(1) 036046 000000          $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
(1) 036050 000776          BR     .-2           ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 036052 000000          $SAVR6: 0           ;;PUT THE SP HERE
(1) 036054 005015 047520 042527 $POWER: .ASCIIZ <15><12>'POWER'
(1) 036062 000122          .EVEN
  
```



```

(1) 036276 000421 BR $SIZE
(1) 036300 042737 100000 036122 $KTNEX: BIC #100000,$KT11 ;;KT11 NON-EXISTENT
(1) 036306 012737 036336 000004 $SCORE: MOV #SCROUT,@ERRVEC ;;SET FOR TIMEOUT
(1) 036314 005002 CLR R2 ;;SET UP BANK
(1) 036316 062701 004000 1$: ADD #4000,R1 ;;INCREMENT BY 1K
(1) 036322 062702 000040 ADD #40,R2 ;;1K STEP
(1) 036326 005711 TST (R1) ;;TRAP ON TIME OUT
(1) 036330 022701 177776 CMP #177776,R1 ;;LAST ONE
(1) 036334 001370 BNE 1$ ;;NO--TRY AGAIN
(1) 036336 162701 004000 $SCROUT: SUB #4000,R1
(1) 036342 162702 000040 $SIZE: SUB #40,R2 ;;DROP BACK
(1) 036346 010006 MOV R0,SP ;;RESTORE THE STACK
(1) 036350 012637 000006 MOV (SP)+,@ERRVEC+2 ;;RESTORE ERROR VECTOR
(1) 036354 012637 000004 MOV (SP)+,@ERRVEC
(1) 036360 010137 036402 MOV R1,$LSTAD ;;LAST ADDRESS
(1) 036364 010237 036404 MOV R2,$LSTBK ;;LAST BANK
(1) 036370 012603 MOV (SP)+,R3 ;;RESTORE R3
(1) 036372 012602 MOV (SP)+,R2 ;;RESTORE R2
(1) 036374 012601 MOV (SP)+,R1 ;;RESTORE R1
(1) 036376 012600 MOV (SP)+,R0 ;;RESTORE R0
(1) 036400 000207 RTS PC
(1) 036402 000000 $LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
(1) 036404 000000 $LSTBK: .WORD 0 ;;CONTAINS THE LAST BANK
2353 036406 000000 KTERR: HALT ;;KT11 FAILURE
2354 036410 000776 BR KTERR
  
```

2356

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

(1)
(2)
(1)
(1)
(1)
(1)
(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) 036412
(3) 036412 010046
(3) 036414 010146
(3) 036416 010246
(3) 036420 010346
(3) 036422 010446
(3) 036424 010546
(1) 036426 016646 000022
(1) 036432 016646 000022
(1) 036436 016646 000022
(1) 036442 016646 000022
(1) 036446 000002
(1)
(1)
(1)
(1)

```
$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) 036450
(1) 036450 012666 000022
(1) 036454 012666 000022
(1) 036460 012666 000022
(1) 036464 012666 000022
(3) 036470 012605
(3) 036472 012604
(3) 036474 012603
(3) 036476 012602
(3) 036500 012601
(3) 036502 012600
(1) 036504 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
```


2360

.SBTTL INTEGER DIVIDE ROUTINE

```
*****
*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
*SAME SIGN AS THE DIVIDEND.
*CALL:
*   MOV     LOW DIVIDEND,-(SP) ;:THE HIGH DIVIDEND MUST BE < 1/2
*   MOV     HIGH DIVIDEND,-(SP);  AS LARGE AS THE DIVISOR
*   MOV     DIVISOR,-(SP)
*   JSR    PC,$DIV
*   RETURN ;:QUOTIENT & REMAINDER ARE ON THE STACK
*   'V'=-0 IMPLIES NO ERROR
*   'V'=1  IMPLIES ERROR OCCURRED
*   'C'=0  DIVIDE OVERFLOW OCCURRED
*   'C'=1  ATTEMPTED TO DIVIDE BY ZERO
```

```

*      STACK NO ERROR OVERFLOW DIVIDE BY ZERO
*      -----
*      TOP REMAINDER ALL ZEROS ALL ONES
*      +2 QUOTIENT ALL ZEROS ALL ONES
```

\$DIV:

```
(1) 036620 TRAP ;:PUSH OLD PSW AND PC ON STACK
(2) 036620 104400
(1) 036622 042716 000017 BIC #17,(SP) ;:STRIP AWAY CONDITION CODES
(3) 036626 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
(3) 036630 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
(3) 036632 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
(3) 036634 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
(1) 036636 005046 CLR -(SP) ;:SAVE A PLACE FOR SIGNS
(1) 036640 012746 000021 MOV #17,-(SP) ;:SETUP THE ITERATION COUNTER
(1) 036644 016601 000024 MOV 24(SP),R1 ;:PICKUP THE DIVIDEND
(1) 036650 016600 000022 MOV 22(SP),R0
(1) 036654 100005 BPL 1$ ;:CHECK THE SIGN
(1) 036656 105366 000003 DECB 3(SP) ;:KEEP TRACK OF THE SIGN
(1) 036662 005400 NEG R0 ;:AND NEGATE THE ORIGINAL
(1) 036664 005401 NEG R1 ;:NUMBER
(1) 036666 005600 SBC R0
(1) 036670 016602 000020 1$: MOV 20(SP),R2 ;:PICKUP THE DIVISOR
(1) 036674 002407 BLT 2$ ;:CHECK THE SIGN
(1) 036676 003011 BGT 3$ ;:DIVISOR OF 0 IS A NO-NO
(1) 036700 052766 000003 000014 BIS #3,14(SP) ;:SET 'V' & 'C'
(1) 036706 012700 177777 MOV #-1,R0 ;:SET REMAINDER TO ALL ONES
(1) 036712 000424 BR 7$ ;:EXIT
(1) 036714 005266 000002 2$: INC 2(SP) ;:KEEP TRACK OF DIVISORS SIGN
(1) 036720 000401 BR 4$
(1) 036722 005402 3$: NEG R2 ;:NEGATE THE ORIGINAL NUMBER
(1) 036724 000241 4$: CLC ;:CLEAR 'C'
(1) 036726 000405 BR 6$ ;:START FORMING QUOTIENT
(1) 036730 006100 5$: ROL R0 ;:POSITION MSB'S
(1) 036732 010003 MOV R0,R3 ;:COPY
(1) 036734 060203 ADD R2,R3 ;:COMPARE DIVIDEND & DIVISOR
(1) 036736 103001 BCC 6$ ;:BR IF DIVIDEND > DIVISOR
```

```

(1) 036740 010300      MOV      R3,R0      ;;REMAINDER AFTER THIS LOOP
(1) 036742 006101      6$:    ROL      R1      ;;QUOTIENT BIT ENTERS HERE
(1) 036744 005316      DEC      (SP)      ;;DONE?
(1) 036746 001370      BNE     5$          ;;BR IF NO
(1) 036750 005701      TST     R1          ;;OVERFLOW?
(1) 036752 100005      BPL     8$          ;;BR IF NO
(1) 036754 052766 000002 000014  BIS     #2,14(SP)   ;;SET 'V' IN RETURN STATUS WORD
(1) 036762 005000      CLR     R0          ;;SET REMAINDER TO ALL ZEROS
(1) 036764 010001      7$:    MOV     R0,R1   ;;COPY REMAINDER INTO QUOTIENT
(1) 036766 005726      8$:    TST     (SP)+   ;;CLEAR COUNTER FROM STACK
(1) 036770 005716      TST     (SP)        ;;REMAINDER SIGN CORRECTION NEEDED?
(1) 036772 002004      BGE     9$          ;;BR IF NO
(1) 036774 005400      NEG     R0          ;;NEGATE REMAINDER
(1) 036776 105066 000001  CLRB   1(SP)       ;;CLEAR SIGN
(1) 037002 005316      DEC     (SP)        ;;BUT DON'T FORGET QUOTIENT
(1) 037004 005726      9$:    TST     (SP)+   ;;QUOTIENT SIGN CORRECTION NEEDED?
(1) 037006 001401      BEQ    10$         ;;BR IF NO
(1) 037010 005401      NEG     R1          ;;NEGATE QUOTIENT
(1) 037012 010166 000020  10$:  MOV     R1,20(SP)  ;;RETURN QUOTIENT AND
(1) 037016 010066 000016  MOV     R0,16(SP)  ;;REMAINDER TO USER
(3) 037022 012603      MOV     (SP)+,R3   ;;POP STACK INTO R3
(3) 037024 012602      MOV     (SP)+,R2   ;;POP STACK INTO R2
(3) 037026 012601      MOV     (SP)+,R1   ;;POP STACK INTO R1
(3) 037030 012600      MOV     (SP)+,R0   ;;POP STACK INTO R0
(1) 037032 012666 000002  MOV     (SP)+,2(SP) ;;SETUP TO RETURN CONDITION CODES
(1) 037036 000002      RTI

```


(1)	037206	000000		0		
(1)	037210	000012		12		:::1.0E01
(1)	037212	000000		0		
(1)	037214	000001		1		:::1.0E00
(1)	037216	000000		0		
(1)	037220	000014				\$DECVL: .BLKB 12. :::RESERVE STORAGE FOR ASCII STRING
2363						.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)	037234	016637	000002	037264	\$SB2D: MOV 2(SP), 1\$:::SAVE BINARY NUMBER
(1)	037242	012746	037264		MOV #1\$, -(SP)	:::SET POINTER
(1)	037246	004737	037040		JSR PC, @#\$DB2D	:::CALL DOUBLE LENGTH CONVERT
(1)	037252	062716	000005		ADD #5, (SP)	:::ONLY ALLOW FIVE CHARACTERS
(1)	037256	012666	000002		MOV (SP)+, 2(SP)	:::PICKUP POINTER
(1)	037262	000207			RTS PC	:::RETURN
(1)	037264	000000	000000	1\$:	.WORD 0,0	

2365
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
2369

.SBTTL TRAP DECODER

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

```
$TRAP: MOV 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
```

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

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

.SBTTL TRAP TABLE

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

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

CZNCB NCV11 DIAGNOSTIC
CZNCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 81
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0122

```

2371 .SBTTL A TO D FIELD SITE AND ADJUSTMENT LOOP
2372 037356 004737 045676 FSITE: JSR PC,CTRLCG ;TEST FOR CTRL C OR G
2373 ;CAMERA #0
2374 037362 032777 000100 141550 BIT #SW06,@SWR ;TEST IF CAM 0 G-1
2375 037370 001004 BNE 1$
2376 037372 004537 040104 JSR R5,CONVRT ;CONVERT
2377 037376 000000 0 ;CAMERA 0
2378 037400 040442 CAMOG1 ;STORE RESULTS
2379 037402 032777 000001 141530 1$: BIT #SW00,@SWR ;TEST IF CAM 0 G=2
2380 037410 001004 BNE 2$
2381 037412 004537 040104 JSR R5,CONVRT ;CONVERT
2382 037416 002000 BIT10 ;GAIN = 2
2383 037420 040444 CAMOG2
2384 ;CAMERA #1
2385 037422 032777 000200 141510 2$: BIT #SW7,@SWR ;TEST IF CAM 1 G 1
2386 037430 001004 BNE 3$
2387 037432 004537 040104 JSR R5,CONVRT ;CONVERT
2388 037436 000400 BIT8 ;CAMERA 1
2389 037440 040446 CAM1G1 ;STORE RESULTS
2390 037442 032777 000002 141470 3$: BIT #SW01,@SWR ;TEST IF CAM 1 G=2
2391 037450 001004 BNE 4$
2392 037452 004537 040104 JSR R5,CONVRT ;CONVERT
2393 037456 002400 BIT10!BIT8 ;CAMERA 1 GAIN = 2
2394 037460 040450 CAM1G2 ;RESULTS
2395 ;CAMERA #2
2396 037462 032777 000400 141450 4$: BIT #SW8,@SWR ;TEST IF CAM 2 G-1
2397 037470 001004 BNE 5$
2398 037472 004537 040104 JSR R5,CONVRT ;CONVERT
2399 037476 001000 BIT9 ;CAMERA #2
2400 037500 040452 CAM2G1 ;RESULTS
2401 037502 032777 000004 141430 5$: BIT #SW2,@SWR ;TEST IF CAM 2 G=2
2402 037510 001004 BNE 6$
2403 037512 004537 040104 JSR R5,CONVRT ;CONVERT
2404 037516 003000 BI110!BIT9 ;GAIN = 2
2405 037520 040454 CAM2G2 ;RESULTS
2406 ;CAMERA #3
2407 037522 032777 001000 141410 6$: BIT #SW9,@SWR ;TEST IF CAM 3 G-1
2408 037530 001004 BNE 7$
2409 037532 004537 040104 JSR R5,CONVRT ;CONVERT
2410 037536 001400 BIT9!BIT8 ;CAMERA #3
2411 037540 040456 CAM3G1 ;RESULTS
2412 037542 032777 000010 141370 7$: BIT #SW3,@SWR ;TEST IF CAM 3 G=2
2413 037550 001004 BNE 10$
2414 037552 004537 040104 JSR R5,CONVRT ;CONVERT
2415 037556 003400 BIT10!BIT9.BIT8 ;GAIN = 2
2416 037560 040460 CAM3G2 ;RESULTS
2417 ;JOYSTICK
2418 037562 032777 000020 141350 10$: BIT #SW4,@SWR ;TEST IF JOYSTICK
2419 037570 001011 BNE CALRPT
2420 037572 052777 000001 142156 11$: BIS #BIT0,@SFR ;ASK FOR JOYSTICK
2421 037600 105777 142152 TSTB @SFR ;WAIT FOR JOY DONE
2422 037604 100375 BPL 11$
2423 037606 017737 142150 040462 MOV @JOY,JOYG1 ;SAVE THE RESULT

```

```

2425      :NOW TEST IS TYPEOUT IS ENABLED
2426
2427 037614 004537 040062 CALRPT: JSR      R5,CKTSWR      ;TEST BIT 6
2428 037620 000100          BIT6
2429 037622 000405          BR      1$      ;BR IF SET
2430 037624 104401 040464          TYPE,    CAM0TX      ;REPORT CAMERA #0
2431 037630 004537 040360          JSR      R5,CAMUNP      ;REPORT VALUES
2432 037634 040442          CAM0G1
2433 037636 004537 040062 1$: JSR      R5,CKTSWR      ;TEST BIT 0
2434 037642 000001          BIT0
2435 037644 000405          BR      2$
2436 037646 104401 040503          TYPE,    CAM0TW      ;REPORT G-2
2437 037652 004537 040360          JSR      R5,CAMUNP
2438 037656 040444          CAM0G2
2439
2440 037660 004537 040062 2$: JSR      R5,CKTSWR      ;TEST BIT 7
2441 037664 000200          BIT7
2442 037666 000405          BR      3$
2443 037670 104401 040522          TYPE,    CAM1TX      ;BR IF SET
2444 037674 004537 040360          JSR      R5,CAMUNP      ;REPORT CAMERA #1
2445 037700 040446          CAM1G1          ;REPORT VALUES
2446 037702 004537 040062 3$: JSR      R5,CKTSWR      ;TEST BIT 1
2447 037706 000002          BIT1
2448 037710 000405          BR      4$
2449 037712 104401 040541          TYPE,    CAM1TW      ;REPORT CAM 1 G=2
2450 037716 004537 040360          JSR      R5,CAMUNP
2451 037722 040450          CAM1G2
2452
2453 037724 004537 040062 4$: JSR      R5,CKTSWR      ;TEST BIT 8
2454 037730 000400          BIT8
2455 037732 000405          BR      5$
2456 037734 104401 040560          TYPE,    CAM2TX      ;BR IF SET
2457 037740 004537 040360          JSR      R5,CAMUNP      ;REPORT CAMERA #2
2458 037744 040452          CAM2G1          ;REPORT VALUES
2459 037746 004537 040062 5$: JSR      R5,CKTSWR      ;TEST BIT 2
2460 037752 000004          BIT2
2461 037754 000405          BR      6$
2462 037756 104401 040577          TYPE,    CAM2TW      ;REPORT CAMERA #2 G 2
2463 037762 004537 040360          JSR      R5,CAMUNP
2464 037766 040454          CAM2G2
2465
2466 037770 004537 040062 6$: JSR      R5,CKTSWR      ;TEST BIT 9
2467 037774 001000          BIT9
2468 037776 000405          BR      7$
2469 040000 104401 040616          TYPE,    CAM3TX      ;BR IF SET
2470 040004 004537 040360          JSR      R5,CAMUNP      ;REPORT CAMERA #3
2471 040010 040456          CAM3G1          ;REPORT VLAUES
2472 040012 004537 040062 7$: JSR      R5,C:TSWR      ;TEST BIT 3
2473 040016 000010          BIT3
2474 040020 000405          BR      10$
2475 040022 104401 040635          TYPE,    CAM3TW      ;REPORT CAMERA #3 G 2
2476 040026 004537 040360          JSR      R5,CAMUNP
2477 040032 040460          CAM3G2
2478
2479 040034 004537 040062 10$: JSR      R5,CKTSWR      ;TEST BIT 4
2480 040040 000020          BIT4

```

CZNCB NCV11 DIAGNOSTIC
CZNCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 82-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0124

2481	040042	000405			BR	11\$	
2482	040044	104401	040654		TYPE,	JOYTW	
2483	040050	004537	040360		JSR	R5,CAMUNP	;REPORT JOYSTICK
2484	040054	040462			JOYG1		
2485	040056	000137	037356		11\$: JMP	FSITE	
2486							
2487	040062	032577	141052		CKTSWR: BIT	(R5)+,@SWR	;TEST IF INHIBIT THIS CAMERA
2488	040066	001005			BNE	1\$;BR IF YES
2489	040070	032777	020000	141042	BIT	#SW13,@SWR	;TEST INHIBIT TYPE-OUT
2490	040076	001001			BNE	1\$;BR IF YES
2491	040100	005725			TST	(R5)+	;BUMP EXIT POINTER
2492	040102	000205			1\$: RTS	R5	;EXIT

```

2494      ;SUBROUTINE TO TAKE 8 CONVERSIONS AND AVERAGE FOUR OF THEM
2495 040104 012537 040354 CONVRT: MOV (R5)+,11$ ;SAVE CAMERA CHANNEL
2496 040110 012777 004000 141640 4$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
2497 040116 013777 040354 141622 MOV 11$,@CSR ;SELECT CAMERA AND GAIN
2498 040124 012777 060000 141622 MOV #BUFO,@BAR ;LOAD BUS ADDRESS
2499 040132 012777 177770 141612 MOV #-10,@WCR ;LOAD WORD COUNT
2500 040140 013777 040356 141610 MOV INOUTZ,@SFR ;SET Z INPUTS IF ENABLED
2501 040146 012700 000004 MOV #4.,R0 ;LOAD COUNTER
2502 040152 005001 CLR R1 ;FOR OUT OF RANGE VOLTAGE
2503 040154 052777 000001 141564 BIS #BIT0,@CSR ;ENABLE THE DEVICE
2504 040162 105777 141560 1$: TSTB @CSR ;WAIT FOR IDLE
2505 040166 100432 BMI 3$ ;
2506 040170 005301 DEC R1 ;DELAY
2507 040172 001373 BNE 1$ ;
2508 040174 004737 045676 JSR PC,CTRLCG ;TEST FOR CTRL G OR C
2509 040200 005300 DEC R0 ;FINISHED OUT OF RANGE TIMER
2510 040202 001367 BNE 1$ ;
2511 040204 032777 020000 140726 BIT #SW13,@SWR ;TEST IF INHIBIT TYPEOUT
2512 040212 001012 BNE 5$ ;BR IF INHIBIT
2513 040214 113700 040355 MOVB 11$+1,R0 ;GET CAMERA #
2514 040220 042700 177774 BIC #177774,R0 ;MASK OFF EXTRA BITS
2515 040224 062700 000060 ADD #60,R0 ;MAKE INTO ASCII
2516 040230 110037 032716 MOVB R0,OUTCHN ;INSERT INTO ASCII ERROR MESSAGE
2517 040234 104401 032623 TYPE, OUTRNG ;TELL OPERATOR
2518 040240 032777 100000 140672 5$: BIT #SW15,@SWR ;TEST IF HALT ON NO CONVERSION ERROR
2519 040246 001401 BEQ 6$ ;BR IF CLEARED
2520 040250 000000 HALT ;NO CONVERT FLAG - INPUT VOLTAGE WAS OUT OF RANGE OR NO 'Z' PULSES
2521 040252 000716 6$: BR 4$ ;
2522 040254 012700 060000 3$: MOV #BUFO,R0 ;LOAD POINTER
2523 040260 005003 CLR R3 ;CLEAR RESULT
2524 040262 005004 CLR R4 ;
2525 040264 012737 000004 040352 MOV #4,10$ ;LOAD COUNTER
2526 040272 112002 2$: MOVB (R0)+,R2 ;GET LOW BYTE
2527 040274 112001 MOVB (R0)+,R1 ;GET HIGH BYTE
2528 040276 042702 177400 BIC #177400,R2 ;
2529 040302 042701 177400 BIC #177400,R1 ;
2530 040306 060104 ADD R1,R4 ;UPDATE RESULT
2531 040310 060203 ADD R2,R3 ;UPDATE RESULT
2532 040312 005337 040352 DEC 10$ ;FINISHED ?
2533 040316 001365 BNE 2$ ;
2534 040320 006203 ASR R3 ;
2535 040322 006203 ASR R3 ;
2536 040324 006204 ASR R4 ;
2537 040326 006204 ASR R4 ;
2538 040330 010437 040352 MOV R4,10$ ;SAVE IT
2539 040334 105037 040353 CLRB 10$+1 ;CLEAR HIGH BYTE
2540 040340 110337 040353 MOVB R3,10$+1 ;LOAD HIGH BYTE
2541 040344 013735 040352 MOV 10$,@(R5)+ ;SAVE THE RESULT
2542 040350 000205 RTS R5 ;
2543 040352 000000 10$: 0 ;
2544 040354 000000 11$: 0 ;
2545 040356 000002 INOUTZ: TESTZ ;0= USE CUSTOMER Z PULSES, 1= MAINT Z PULSES
2546
2547 ;SUBROUTINE TO TYPE THE CAMERA MESSAGE
2548 040360 013537 040436 CAMUNP: MOV @(R5)+,10$ ;GET RESULT
2549 040364 113737 040437 040440 MOVB 10$+1,11$ ;

```

CZNCB NCV11 DIAGNOSTIC
CZNCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 83-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0126

2550	040372	105037	040437	CLRB	10\$+1	
2551	040376	105037	040441	CLRB	11\$+1	
2552	040402	013746	040436	MOV	10\$,-(SP)	
2553	040406	104403		TYPOS		
2554	040410	003	001	.BYTE	3,1	:TYPE 3 DIGIT NUMBER
2555	040412	104401	040672	TYPE,	G1B	:AND TRALING ASCII
2556	040416	013746	040440	MOV	11\$,-(SP)	
2557	040422	104403		TYPOS		
2558	040424	003	001	.BYTE	3,1	:TYPE 3 DIGIT NUMBER
2559	040426	000240		NOP		
2560	040430	000240		NOP		
2561	040432	000240		NOP		
2562	040434	000205		RTS	R5	:EXIT
2563	040436	000000		10\$:	0	
2564	040440	000000		11\$:	0	
2565	040442	000000		CAM0G1:	0	
2566	040444	000000		CAM0G2:	0	
2567	040446	000000		CAM1G1:	0	
2568	040450	000000		CAM1G2:	0	
2569	040452	000000		CAM2G1:	0	
2570	040454	000000		CAM2G2:	0	
2571	040456	000000		CAM3G1:	0	
2572	040460	000000		CAM3G2:	0	
2573	040462	000000		JOYG1:	0	
2574						
2575	040464	015	012	CAM0TX:	.BYTE 15,12	
2576	040466	040503	030115	.ASCIZ	/CAM00 G-1 X /	020060 036530
	040474	036507	020061			036530
	040502	000				
2577	040503	015	012	CAM0TW:	.BYTE 15,12	
2578	040505	103	046501	.ASCIZ	/CAM00 G-2 X-/	030060 054040
	040512	043440	031075			054040
	040520	000075				
2579	040522	015	012	CAM1TX:	.BYTE 15,12	
2580	040524	040503	030115	.ASCIZ	/CAM01 G=1 X=/	020061 036530
	040532	036507	020061			036530
	040540	000				
2581	040541	015	012	CAM1TW:	.BYTE 15,12	
2582	040543	103	046501	.ASCIZ	/CAM01 G=2 X=/	030460 054040
	040550	043440	031075			054040
	040556	000075				
2583	040560	015	012	CAM2TX:	.BYTE 15,12	
2584	040562	040503	030115	.ASCIZ	/CAM02 G=1 X=/	020062 036530
	040570	036507	020061			036530
	040576	000				
2585	040577	015	012	CAM2TW:	.BYTE 15,12	
2586	040601	103	046501	.ASCIZ	/CAM02 G=2 X-/	031060 054040
	040606	043440	031075			054040
	040614	000075				
2587	040616	015	012	CAM3TX:	.BYTE 15,12	
2588	040620	040503	030115	.ASCIZ	/CAM03 G=1 X-/	020063 036530
	040626	036507	020061			036530
	040634	000				
2589	040635	015	012	CAM3TW:	.BYTE 15,12	
2590	040637	103	046501	.ASCIZ	/CAM03 G=2 X-/	031460 054040
	040644	043440	031075			054040

CZNCB NCV11 DIAGNOSTIC
CZNCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 83-2
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0127

2591	040652	000075							
2592	040654	015	012			JOYTX:			
2593	040656	047512	051531	044524		JOYTW:	.BYTE 15,12		
	040664	045503	054040	000075			.ASCIZ /JOYSTICK X=/		
2594	040672	054440	000075			G1B:	.ASCIZ / Y=/		
2595	040676	005015	044504	043106		GAIN:	.ASCII <15><12>/DIFFERENTIAL LINEARITY:/		
	040704	051105	047105	044524					
	040712	046101	046040	047111					
	040720	040505	044522	054524					
	040726	072							
2596	040727	040	020040	040507		GAINX:	.ASCII / GAIN = /		
	040734	047111	036440	040					
2597	040741	061	040	040		GAIN1:	.BYTE 61,40,40,0		
	040744	000							
2598	040745	040	026455	000040		DASH:	.ASCIZ / -- /		
2599	040752	046040	041123	005015		LSBMSG:	.ASCIZ / LSB/<15><12>		
	040760	000							
2600	040761	040	045523	050111		SKPMSG:	.ASCIZ / SKIPPED STATE(S)/<15><12>		
	040766	042520	020104	052123					
	040774	052101	024105	024523					
	041002	005015	000						
2601	041005	040	025052	051105		ERMSG:	.ASCIZ / **ERROR**/<15><12>		
	041012	047522	025122	006452					
	041020	000012							
2602	041022	020040	020040	047440		OKMSG:	.ASCIZ / OK/<15><12>		
	041030	006513	000012						
2603	041034	047040	051101	047522		NARMSG:	.ASCIZ # NARROW STATE(S)#<15><12>		
	041042	020127	052123	052101					
	041050	024105	024523	005015					
	041056	000							
2604	041057	040	044527	042504		WIDMSG:	.ASCIZ # WIDE STATE(S)#<15><12>		
	041064	051440	040524	042524					
	041072	051450	006451	000012					
2605	041100	052123	052101	026505		STATE:	.ASCIZ /STATE-- WIDTH/<15><12>		
	041106	020055	044527	052104					
	041114	006510	000012						
2606	041120	046040	041123	046440		LINEA:	.ASCIZ / LSB MAXIMUS AT /		
	041126	054101	046511	051525					
	041134	040440	020124	000					
2607	041141	057	000			SLASH:	.ASCIZ #/#		
2608	041143	122	046105	052101		MSG21:	.ASCIZ /RELATIVE ACCURACY:/<15><12>		
	041150	053111	020105	041501					
	041156	052503	040522	054503					
	041164	006472	000012						
2609	041170	005015	047522	020115		COMP:	.ASCIZ <15><12>/ROM BLASTING COMPLETED/		
	041176	046102	051501	044524					
	041204	043516	041440	046517					
	041212	046120	052105	042105					
	041220	000							
2610	041221	015	052012	046511		TIMEO:	.ASCII <15><12>/TIMEOUT FROM BLASTER CHECK SWITCHES ON BLASTER/		
	041226	047505	052125	043040					
	041234	047522	020115	046102					
	041242	051501	042524	020122					
	041250	020040	044103	041505					
	041256	020113	053523	052111					

2611	041264	044103	051505	047440	
	041272	020116	046102	051501	
	041300	042524	122		
	041303	015	020012	051117	.ASCIZ <15><12>/ OR REPLACE ROM IN BLASTER/<15><12>
	041310	051040	050105	040514	
	041316	042503	051040	046517	
	041324	044440	020116	046102	
	041332	051501	042524	006522	
	041340	000012			
2612	041342	015	012		FIELDI: .BYTE 15,12
2613	041344	047111	042524	047122	.ASCIZ /INTERNAL MAINT. Z PULSES <Y FOR YES> ? /
	041352	046101	046440	044501	
	041360	052116	020056	020132	
	041366	052520	051514	051505	
	041374	036040	020131	047506	
	041402	020122	042531	037123	
2614	041410	037440	000040		
	041414	005015	053101	051105	AVRGO: .ASCIZ <15><12>/AVERAGE OF 128 STATES - /
	041422	043501	020105	043117	
	041430	030440	034062	051440	
	041436	040524	042524	020123	
	041444	020075	000		
2615	041447	015	012	007	UNFIX: .BYTE 15,12,7
2616	041452	054105	042503	042105	.ASCIZ /EXCEEDED CORRECTION COUNTER FOR /
	041460	042105	041440	051117	
	041466	042522	052103	047511	
	041474	020116	047503	047125	
	041502	042524	020122	047506	
	041510	020122	000		
2617	041513	015	012	007	ABORTB: .BYTE 15,12,7
2618	041516	047522	020115	046102	.ASCII /ROM BLASTING ABORTED/
	041524	051501	044524	043516	
	041532	040440	047502	052122	
	041540	042105			
2619	041542	015	012	007	.BYTE 15,12,7,0
	041545	000			
2620	041546	020040	026455	020040	SGNVAL: .ASCII / -- /
2621	041554	053	040		SGNVL1: .BYTE 53,40
2622	041556	060	060	056	PERTXT: .BYTE 60,60,56,60,45,15,12,0
	041561	060	045	015	
	041564	012	000		
2623	041566	015	012	000	RTN: .BYTE 15,12,0,0
	041571	000			
2624	041572	041040	046105	053517	BELMSG: .ASCIZ / BELOW LIMIT/<15><12>
	041600	046040	046511	052111	
	041606	005015	000		
2625	041611	040	047516	046522	NORMSG: .ASCIZ / NORMAL STATES/<15><12>
	041616	046101	051440	040524	
	041624	042524	006523	000012	
2626	041632	040440	047502	042526	ABOMSG: .ASCIZ / ABOVE LIMIT/<15><12>
	041640	046040	046511	052111	
	041646	005015	000		
2627	041651	040	040520	051523	PASMSG: .ASCIZ / PASSED/<15><12>
	041656	042105	005015	000	
2628	041663	040	040506	046111	FAIMSG: .ASCIZ / FAILED/<15><12>
	041670	042105	005015	000	

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 83-4
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0129

2629		041676			
2630	041676	000000		LSBAVG: 0	
2631	041700	000000		LSBSVQ: 0	
2632	041702	000000		LSBSVR: 0	
2633	041704	000000		LSBSVW: 0	
2634	041706	000000		DIFEX1: 0	
2635	041710	042132		ROMPNT: ROMVAL	
2636	041712	027340		NOMIAL: 12000.	: AVERAGE COUNT
2637					
2638					: THESE LOC.'S CLEARED BY DIFLIN
2639	041714	000000		AVGVAL: 0	
2640	041716	000000		DIFERR: 0	
2641	041720	000000		BELOW: 0	
2642	041722	000000		ABOVE: 0	
2643	041724	000000		NORMAL: 0	
2644	041726	000000		NARROW: 0	
2645	041730	000000		NARA: 0	
2646	041732	000000		NARB: 0	
2647	041734	000000		WIDA: 0	
2648	041736	000000		WIDB: 0	
2649	041740	000000		DIF: 0	
2650	041742	000000		OUT: 0	
2651	041744	000000		FIRST: 0	

```

2653
2654 ;KEYBOARD CONVERSATION TO THE BLASTER SELECT TEST 'T'
2655 ;APPLY POWER TO BLASTER AND DEPRESS I/O AND EXECUTE BUTTONS
2656 ;ALL INDICATORS SHOULD LIGHT THE BLASTER SHOULD SEND A '>' CHARACTER
2657 ;TYPE THE COMMANDS FOLLOWED BY A 'CR', SOME WILL ECHO THE CR AND OTHERS
2658 ;WILL NOT TYPE 'ESC'/'ALT' KEYS TO ABORT CURRNET COMMAND
2659 ;TYPE KF2/2 ;THIS SETS OCTAL COMMAND MODE
2660 ;TYPE FM30 ;THIS SET OCTAL INPUT/OUTPUT
2661 ;TYPE SPE[0037/8] ;THIS SIZES THE PROM -- 0037/8 IS REPORTED BY THE BLASTER
2662 ;TYPE LD ;THIS READS THE PROM INTO THE RAM
2663 ;TYPE D00/37 ;THIS OUTPUTS THE RAM CONTENTS TO THE TTY
2664 ;TYPE ZP ;THIS EXITS COMPUTER CONTROL TO THE BUTTON MODE
2665
2666 041746 105777 137724 LOOPC: TSTB @DLICSR ;BLASTER INPUT ?
2667 041752 100007 BPL LOOPD ;NO
2668 041754 000240 NOP
2669 041756 000240 NOP
2670 041760 000240 NOP
2671 041762 017720 137712 MOV @DLIBD,(R0)+ ;SAVE THE CHARACTER
2672 041766 005237 042130 INC BLICNT ;UPDATE COUNTER
2673
2674 041772 105777 137152 LOOPD: TSTB @STPS ;PRINTER READY ?
2675 041776 100363 BPL LOOPC ;BR IF NOT
2676 042000 005737 042130 TST BLICNT ;ANY CHARACTERS
2677 042004 001413 BEQ LOOPE ;BR IF NOT
2678 042006 012177 137140 MOV (R1)+,@STPB ;PRINT THE CHAR.
2679 042012 005337 042130 DEC BLICNT
2680 042016 001353 BNE LOOPC ;BR IF MORE DATA
2681 042020 012700 060000 BTALK: MOV #BUF0,R0
2682 042024 010001 MOV R0,R1
2683 042026 005037 042130 CLR BLICNT
2684 042032 000745 BR LOOPC
2685 ;COME HERE IS NO BLASTER DATA TO BE TYPED
2686 042034 105777 137104 LOOPE: TSTB @STKS ;KEYBOARD INPUT
2687 042040 100342 BPL LOOPC ;BR IF NOT
2688 042042 017702 137100 MOV @STKB,R2 ;READ CHAR
2689 042046 042702 177600 BIC #177600,R2 ;MASK THE DATA
2690 042052 022702 000003 CMP #3,R2 ;TEST FOR CTRL C
2691 042056 001413 BEQ 2$ ;BR IF CTRL C
2692 042060 022702 000177 CMP #177,R2 ;TEST FOR RUBOUT
2693 042064 001412 BEQ LOOPF ;BR IF ESC
2694 042066 020227 000140 CMP R2,#140 ;TEST IF LOWER CASE
2695 042072 103402 BLO 1$ ;BR IF NOT
2696 042074 042702 000040 BIC #40,R2 ;MAKE UPPER CASE
2697 042100 010277 137600 1$: MOV R2,@DL0DB ;LOAD BLASTER DATA OUT
2698 042104 000720 BR LOOPC
2699 042106 000137 002646 2$: JMP RBEGO ;JUMP TO RESTART
2700 042112 012777 000033 137564 LOOPF: MOV #33,@DL0DB ;LOAD ESC <CANCLE>
2701 042120 012777 000134 137024 MOV #' \,@STPB
2702 042126 000707 BR LOOPC
2703 042130 000000 BLICNT: 0

```

```

2705
2706      ;A017 CORRECTION PROM VALUES  <MEMORY RAM>
2707
2708 042132      200      200      200  ROMVAL: .BYTE  200,200,200,200,200,200,200,200 ;GAIN = 1 VALUE
      042135      200      200      200
      042140      200      200
2709 042142      200      200      200      .BYTE  200,200,200,200,200,200,200,200
      042145      200      200      200
2710 042150      200      200
      042152      200      200      200      .BYTE  200,200,200,200,200,200,200,200 ;GAIN = 2 VALUE
      042155      200      200      200
2711 042160      200      200
      042162      200      200      200      .BYTE  200,200,200,200,200,200,200,200
      042165      200      200      200
      042170      200      200

2712
2713      ;A017 CORRECTION PRAM VALUES  <RAM MEMORY>
2714      ;ROMVAL MUST BE IN COMPLEMENTED AND REVERSED ORDFR <LSB=MSB>
2715
2716 042172      376      376      376  RAMVAL: .BYTE  376,376,376,376,376,376,376,376 ;GAIN = 1 VALUE
      042175      376      376      376
      042200      376      376
2717 042202      376      376      376      .BYTE  376,376,376,376,376,376,376,376
      042205      376      376      376
      042210      376      376
2718 042212      376      376      376      .BYTE  376,376,376,376,376,376,376,376 ;GAIN = 2 VALUE
      042215      376      376      376
      042220      376      376
2719 042222      376      376      376      .BYTE  376,376,376,376,376,376,376,376
      042225      376      376      376
      042230      376      376

2720
2721      ;M8036 PROGRAM PROM VALUES
2722
2723
2724 042232      215      244      116  PROROM: .BYTE  215,244,116,144,215,244,116,144 ;M8036 PROGRAM PROM
      042235      144      215      244
      042240      116      144
2725 042242      215      244      116      .BYTE  215,244,116,144,215,244,116,144
      042245      144      215      244
      042250      116      144
2726 042252      106      144      314      .BYTE  106,144,314,304,377,377,377,377
      042255      304      377      377
      042260      377      377
2727 042262      377      377      377      .BYTE  377,377,377,377,377,377,377,377
      042265      377      377      377
      042270      377      377
  
```

```

2729 ;BLASTER SECTION
2730 042272 104401 050762 BLAST: TYPE, PRIMO ;TELL OPERATOR ABOUT THE CABLE
2731 042276 104401 051374 TYPE, PRIM5 ;AND SWITCHES
2732 042302 042777 000001 137366 BIC #BIT0,@DLICSR ;ENSURE INIT. ROM CONTROL LINE
2733 042310 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO INITILIZE
2734 042314 052560 ESCP ;REALLY WE ARE WAITING FOR THE OPERATOR
2735 042316 004537 043062 JSR R5,RDBLST ;TELI THE BLASTER THE KEYBOARD INPUT MODE
2736 042322 052562 KFO
2737 042324 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
2738 042330 052571 FMO
2739 ;PRIME THE MEMORY RAM BUFFER
2740 042332 012700 042132 MOV #ROMVAL,R0 ;GET POINTER
2741 042336 112720 000200 1$: MOVB #200,(R0)+ ;LOAD THE BUFFER
2742 042342 022700 042172 CMP #ROMVAL+40,R0 ;TEST IF DONE
2743 042346 001373 BNE 1$ ;BR IF BUFFER NOT PRIMED
2744 042350 012737 000036 046002 MOV #36,PRIME1 ;LOAD GAIN AND RESOL. VALUES
2745 042356 004537 046006 JSR R5,LIMITS ;LOAD DIFLIN ERROR LIMIT VALUES
2746 042362 046072 G1LIMO ;G1 LIMIT - USERS
2747 042364 046112 G1LIM1 ;G1 LIMIT - OPTION CHECKOUT
2748 042366 112737 000061 040741 MOVB #'1,GAIN1 ;LOAD GAIN TYPEOUT VALUE
2749 042374 012737 042133 041710 MOV #ROMVAL+1,ROMPNT ;LOAD RAM POINTER
2750 042402 004737 042446 JSR PC,4$ ;TEST THAT GAIN
2751 042406 012737 002036 046002 MOV #2036,PRIME1 ;LOAD GAIN AND RESOL. VALUES
2752 042414 004537 046006 JSR R5,LIMITS ;LOAD DIFLIN ERROR LIMIT VALUES
2753 042420 046132 G2LIMO ;G2 LIMIT - USERS
2754 042422 046152 G2LIM1 ;G2 LIMIT - OPTION CHECKOUT
2755 042424 112737 000062 040741 MOVB #'2,GAIN1 ;LOAD GAIN TYPEOUT VALUE
2756 042432 012737 042153 041710 MOV #ROMVAL+21,ROMPNT ;LOAD RAM POINTER
2757 042440 004737 042446 JSR PC,4$ ;TEST GAIN OF 2
2758 042444 000436 BR 10$
2759 042446 012737 000003 042616 4$: MOV #3,100$ ;LOAD LOOP COUNTER
2760 042454 012737 000001 041706 5$: MOV #1,DIFEX1 ;LOAD DIFLIN EXIT FLAG
2761 042462 012737 042500 046004 MOV #6$,DIFEXO ;LOAD DIFLIN EXIT RETURN
2762 042470 004737 042620 JSR PC,ZAPRAM ;ENSURE BLASTER RAM HAS BEEN LOADED WITH
2763 ;THE 'RAMVAL' VALUE
2764 042474 000137 043720 JMP DIFLNO ;RUN 'DIFLIN'
2765 042500 004737 043360 6$: JSR PC,ADJFIX ;ADJUST AND FIX THE CORRECTION WEIGHTS
2766 042504 005337 042616 DEC 100$ ;HAS THIS SECTION BEEN EXECITED N TIMES
2767 042510 001361 BNE 5$ ;BR IF NOT
2768 042512 005737 041716 TST DIFERR ;TEST IF ERROR OCCURRED?
2769 042516 001410 BEQ 8$ ;BR IF NOT
2770 042520 104401 041447 7$: TYPE, UNFIX ;TELL THE OPERATOR
2771 042524 104401 040727 TYPE, GAINX ;ABOUT IT
2772 042530 104401 041513 TYPE, ABORTB ;TELL OPER. THE BLASTING IS ABORTED
2773 042534 000137 002646 JMP RBEGO ;RESTART
2774 042540 000207 8$: RTS PC ;EXIT

```

```

2776 ;THE RAM DATA IS NOW CORRECT - BLAST THE BLANK ROM
2777 042542 004537 043062 10$: JSR R5,RDBLST ;TELL THE BLASTER TO COMP. THE DATA
2778 042546 053011 CMO
2779 042550 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO CHECK-SUM DATA
2780 042554 053021 CCO
2781 042556 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO BLAST THE ROM
2782 042562 053024 PGO
2783 042564 005237 043242 INC NOEXIT ;SET FLAG TO EXIT WITHOUT RECPT OF '>' CHAR FROM BLASTER
2784 042570 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
2785 042574 053034 ZPO
2786 ;INFORM THE OPERATOR ABOUT THE BLASTER ROM
2787 042576 104401 051765 TYPE, PRIM1 ;TELL OPERATOR TO REMOVE RAM CABLE AND INSTERT ROM INTO A017
2788 042602 104411 RDLIN
2789 042604 012600 MOV (SP)+,R0 ;CLEAN STACK
2790 042606 104401 041170 TYPE, COMP ;COMPLETED
2791 042612 000137 002646 JMP RBEGO
2792 042616 000000 100$: 0
2793 ;SUBROUTINE TO CONVERT THE MEMORY RAM VALUE INTO RAM MEMORY VALUE
2794 042620 012703 042172 ZAPRAM: MOV #RAMVAL,R3 ;LOAD OUTPUT POINTER
2795 042624 012700 042132 MOV #ROMVAL,R0 ;LOAD INPUT POINTER
2796 042630 112001 1$: MOVB (R0)+,R1 ;GET A BYTE
2797 042632 105101 COMB R1 ;INVERT VALUE
2798 042634 005002 CLR R2 ;CLEAR RESULT
2802 042636 006001 POR R1 ;MOVE INTO CARRY BIT
(1) 042640 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042642 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042644 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042646 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042650 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042652 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042654 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042656 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042660 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042662 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042664 006102 ROI R2 ;MOVE CARRY INTO BIT 0
(1) 042666 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042670 006102 ROL R2 ;MOVE CARRY INTO BIT 0
(1) 042672 006001 ROR R1 ;MOVE INTO CARRY BIT
(1) 042674 006102 ROL R2 ;MOVE CARRY INTO BIT 0
2803 042676 110223 MOVB R2,(R3)+ ;SAVE OUTPUT BYTE
2804 042700 020027 042172 CMP R0,#ROMVAL+40 ;TEST IF END
2805 042704 001351 BNE 1$ ;BR IF NOT
2806 ;SUBROUTINE TO LOAD MEMORY RAM INTO RAM MEMORY
2807 042706 004537 043244 JSR R5,CONROM ;CONVERT 'ROMVAL' TABLE INTO BLASTER FORMAT
2808 042712 042172 RAMVAL
2809 042714 052777 000001 136754 BIS #BIT0,@DLICSR ;ENABLE RAM MEMORY TO BE LOADED
2810 042722 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER THE NEW RAM DATA
2811 042726 052577 DIO
2812 042730 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO CHECKSUM THE DATA
2813 042734 053021 CCO
2814 042736 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO PROGRAM THE RAM-PAK
2815 042742 053024 PGO
2816 042744 042777 000001 136724 BIC #BIT0,@DLICSR ;RE-ENTER EMMULATE MODE
2817 042752 000207 RTS ;EXIT

```

```

2819          ;BLASTING THE PROGRAM PROM (M8036)
2820
2821 042754 104401 051536 PBLAST: TYPE, PRIM3      ;INFORM THE OPERATOR
2822 042760 042777 000001 136710 BIC #BIT0,ADLICSR ;ENSURE ROM MODE
2823 042766 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO INITILIZE
2824 042772 052560 ESCP
2825 042774 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER THE KEYBOARD INPUT MODE
2826 043000 052562 KFO
2827 043002 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
2828 043006 052571 FMO
2829 043010 004537 043244 JSR R5,CONROM ;CONVERT THE 'PROROM' TABLE INTO BLASTER FORMAT
2830 043014 042232 PROROM
2831 043016 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER THE PROM DATA
2832 043022 052577 DIO
2833 043024 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO CHECKSUM THE DATA
2834 043030 053021 CCO
2835 043032 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO BLAST THE PROM
2836 043036 053024 PGO
2837 043040 005237 043242 INC NOEXIT ;SET FLAG TO EXIT WITHOUT RECPT OF '>' FROM BLASTER
2838 043044 004537 043062 JSR R5,RDBLST ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
2839 043050 053034 ZPO
2840 043052 104401 041170 TYPE, COMP ;INFORM THE OPERATOR IT'S DONE
2841 043056 000137 002646 JMP RBEGO

```

```

2843
2844 ;COME THERE TO OUTPUT A MESSAGE TO THE BLASTER
2845 ;EXIT ONLY WHEN THE BLASTER HAS ECHOED AN '>'
2846 ;REPORT AN ERROR UPON TIMEOUT UNLESS 'NOEXIT' IS NON-ZERO
2847 043062 010046 RDBLST: MOV R0, -(SP) ;SAVE R0
2848 043064 012500 MOV (R5)+, R0 ;GET ARG.
2849 043066 012737 000012 043240 MOV #10, 13$ ;LOAD DELAY FOR ERROR COUNTER
2850 043074 111037 043232 MOV (R0), 10$ ;LOAD A CHAR
2851 043100 017737 136574 043234 MOV @DLIBD, 11$ ;FALSE READ TO CLEAR READY
2852 043106 105777 136570 1$: TST @DLOCSR ;WAIT FOR OUTPUT READY
2853 043112 100375 BPL 1$
2854 043114 105737 043232 TST 10$ ;TEST IF FINISHED ?
2855 043120 001406 BEQ 2$
2856 043122 112037 043232 MOV (R0)+, 10$ ;GET A CHAR.
2857 043126 001403 BEQ 2$ ;BR IF TERM
2858 043130 113777 043232 136546 MOVB 10$, @DLODB ;OUTPUT THE CHARACTER
2859 043136 105777 136534 2$: TST @DLICSR ;WAIT FOR INPUT
2860 043142 100415 BMI 4$ ;BR IF INPUT
2861 043144 005337 043236 DEC 12$ ;DELAY
2862 043150 001372 BNE 2$
2863 043152 005337 043240 DEC 13$ ;DELAY
2864 043156 001367 BNE 2$
2865 043160 005737 043242 TST NOEXIT ;TEST IF EXIT IS ALLOWED WITHOUT '>'
2866 043164 001016 BNE 5$ ;BR IF YES
2867 043166 104401 041221 TYPE, TIMEO
2868 043172 000137 002646 JMP RBEGO
2869 043176 017737 136476 043234 4$: MOV @DLIBD, 11$ ;READ CHAR
2870 043204 042737 177600 043234 BIC #177600, 11$ ;MASK OFF BITS
2871 043212 022737 000076 043234 CMP #'>, 11$ ;TEST FOR >
2872 043220 001332 BNE 1$ ;TRY NEXT CHAR.
2873 043222 012600 5$: MOV (SP)+, R0 ;RESTORE R0
2874 043224 005037 043242 CLR NOEXIT
2875 043230 000205 RTS R5 ;EXIT
2876
2877 043232 000000 10$: 0
2878 043234 000000 11$: 0
2879 043236 000000 12$: 0
2880 043240 000000 13$: 0
2881 043242 000000 NOEXIT: 0

```



```
2883  
2884 ;CONVERT ROM DATA INTO PROM BLASTER FORMAT  
2885 043244 012700 052612 CONROM: MOV #DIDATA+3,R0 ;LOAD RESULT POINTER  
2886 043250 012501 MOV (R5)+,R1 ;LOAD INITIAL VALUE POINTER  
2887 043252 010137 043322 MOV R1,10$ ;COPY THE START  
2888 043256 062737 000040 043322 1$: ADD #40,10$ ;MAKE THE END ADD SS  
2889 043264 111102 MOV (R1),R2 ;GET A BYTE OF DATA  
2890 043266 004737 043344 JSR PC,SHUF0 ;CONVERT IT TO ASCII  
2891 043272 111102 MOV (R1),R2 ;GET SAME BYTE AGAIN  
2892 043274 004737 043336 JSR PC,SHUF3 ;CONVERT 2ND DIGIT  
2893 043300 112102 MOV (R1)+,R2 ;GET MSD  
2894 043302 004737 043324 JSR PC,SHUF6 ;CONVERT IT  
2895 043306 062700 000007 ADD #7,R0 ;UPDATE RESULT POINTER  
2896 043312 020137 043322 CMP R1,10$ ;TEST IF DONE  
2897 043316 001362 BNE 1$  
2898 043320 000205 RTS R5 ;EXIT  
2899 043322 000000 10$: 0  
2900  
2901 ;ROTATE THE DATA RIGHT AND MAKE ASCII CHARACTER  
2902  
2903 043324 042702 177477 SHUF6: BIC #177477,R2 ;MASK OFF BITS  
2904 043330 006202 ASR R2  
2905 043332 006202 ASR R2  
2906 043334 006202 ASR R2  
2907 043336 006202 SHUF3: ASR R2  
2908 043340 006202 ASR R2  
2909 043342 006202 ASR R2  
2910 043344 042702 177770 SHUF0: BIC #177770,R2 ;MASK OFF BITS  
2911 043350 062702 000060 ADD #60,R2 ;MAKE ASCII  
2912 043354 110240 MOV R2,-(R0) ;LOAD RESULT BYTE  
2913 043356 000207 RTS PC ;EXIT  
2914
```

```

2916      :SUBROUTINE TO ADJUST THE LSB VALUES
2917 043360 012700 062036  ADJFIX: MOV    #BUF1+36,R0    ;GET BUFFER POINTER
2918 043364 013702 041710      MOV    ROMPNT,R2      ;GET ROM POINTER
2919 043370 010204      MOV    R2,R4
2920 043372 062704 000017      ADD    #17,R4        ;MAKE END VALUE
2921 043376 012001 10$: MOV    (R0)+,R1      ;GET LSB (17) M-1
2922 043400 061001      ADD    (R0),R1      ;ADD LSB (00) M
2923 043402 163701 041714      SUB    AVGVAL,R1    ;SUB 2 LSB AVG.
2924 043406 163701 041714      SUB    AVGVAL,R1
2925 043412 010137 041704      MOV    R1,LSBSVW    ;SAVE FOR LATER USAGE
2926 043416 010146      MOV    R1,-(SP)     ;SAVE ON STACK
2927 043420 012746 000316      MOV    #316,-(SP)  ;MULTI BY 316 (8)
2928 043424 004737 036506      JSR    PC,$MULT
2929 043430 012637 043542      MOV    (SP)+,100$   ;SAVE LSW
2930 043434 012637 043544      MOV    (SP)+,101$   ;SAVE MSW
2931
2932 043440 013746 043542      MOV    100$,-(SP)   ;
2933 043444 013746 043544      MOV    101$,-(SP)   ;
2934 043450 013746 041714      MOV    AVGVAL,-(SP)
2935 043454 004737 036620      JSR    PC,$DIV
2936 043460 102004      BVC    1$          ;BR IF NO ERROR
2937 043462 000000      HALT                ;MATH ERROR
2938 043464 000000      HALT                ;VALUE OUT OF RANGE
2939 043466 000240      NOP
2940 043470 000240      NOP
2941 043472 012637 041702 1$: MOV    (SP)+,LSBSVR  ;SAVE INTGR. REMAINDER
2942 043476 012637 041700      MOV    (SP)+,LSBSVQ ;SAVE QUOTIENT
2943
2944      ;NOW ADD THE QUOTIENT TO THE MEMORY RAM VALUE
2945
2946 043502 111203      MOVB   (R2),R3      ;GET CURRENT MEMORY RAM VALUE
2947 043504 000240      NOP
2948 043506 042703 177400      BIC    #177400,R3   ;CLEAR OFF UPPER BITS
2949 043512 163703 041700      SUB    LSBSVQ,R3    ;ADD THE QUOTIENT
2950 043516 110322      MOVB   R3,(R2)+     ;RELOAD MEMORY RAM VALUE
2951
2952      ;NOW UPDATE 'M' VALUE POINTER
2953
2954 043520 062700 000036      ADD    #36,R0       ;UPDATE POINTER
2955 043524 011001      MOV    (R0),R1      ;GET VALUE
2956 043526 063701 041704      ADD    LSBSVW,R1    ;CORRECT THE VALUE (#10)
2957 043532 010110      MOV    R1,(R0)     ;RESTORE FOR NEXT USAGE
2958 043534 020204      CMP    R2,R4        ;TEST WHEN FINISHED
2959 043536 001317      BNE    10$         ;BR UNTIL DONE
2960
2961 043540 000207      RTS    PC           ;EXIT
2962
2963 043542 000000 100$: 0
2964 043544 000000 101$: 0

```

```

2966
2967
2968 043546 005037 041706      ;:DIFFERENTIAL LINEARITY
2969 043552 112737 000061 040741 DIFLIN: CLR DIFEX1      ;CLEAR EXIT FLAG
2970 043560 105737 001134      MOV#B #1,GAIN1      ;LOAD GAIN MESSAGE VALUE
2971 043564 001007      TSTB $AUTOB        ;TEST IF UNDE? MONITOR
2972 043566 104401 040676      BNE 10$            ;BR IF YES
2973 043572 005737 043546      TYPE, GAIN
2974 043576 100402      TST DIFLIN        ;TEST BIT 15
2975 043600 104401 050762      BMI 10$           ;DONT TELL OPER. ABOUT A017
2976 043604 012737 000036 046002 10$: MOV #36,PRIME1    ;TELL OPERATOR ABOUT A017
2977 043612 004537 046006      JSR R5,LIMITS     ;LOAD RESOLUTION AND GAIN
2978 043616 046072      G1LIMO            ;LOAD DIFLIN TOLERANCE
2979 043620 046112      G1LIM1           ;G1 USER LIMIT
2980 043622 012737 043632 046004 MOV #1$,DIFEXO    ;G1 OPTION AREA
2981 043630 000433      BR DIFLNO        ;LOAD RETURN ADDRESS
2982 043632 112737 000062 040741 1$: MOV#B #2,GAIN1    ;LOAD GAIN MESSAGE VALUE
2983 043640 012737 002036 046002 MOV #2036,PRIME1  ;LOAD RESOLUTION AND GAIN
2984 043646 004537 046006      JSR R5,LIMITS     ;LOAD DIFLIN TOLERANCE
2985 043652 046132      G2LIMO            ;G2 USER LIMIT
2986 043654 046152      G2LIM1           ;G2 OPTION AREA
2987 043656 012737 043700 046004 MOV #2$,DIFEXO    ;LOAD RETURN ADDRESS
2988 043664 105737 001134      TSTB $AUTOB        ;RUNNING UNDER MONITOR
2989 043670 001002      BNE 11$          ;BR IF YES
2990 043672 104401 040676      TYPE, GAIN
2991 043676 000410      BR DIFLNO
2992 043700 022737 177777 050022 2$: CMP #-1,RUNDIF  ;ENTER VIA 'F' SELECTION ?
2993 043706 001002      BNE 3$           ;BR IF NOT
2994 043710 000137 030100      JMP $EOP         ;YES REPORT END OF PASS
2995 043714 000137 002646      3$: JMP RBEGO
2996
2997
2998 ;DIFLIN ROUTINE
2999 043720 012700 041714 DIFLNO: MOV #AVGVAL,R0 ;LOAD CLEARING POINTER
3000 043724 005020 DIFLNx: CLR (R0)+
3001 043726 022700 041746      CMP #FIRST+2,R0  ;FINISHED ?
3002 043732 001374      BNE DIFLNx       ;BR IF NOT DONE
3003 043734 012700 060000      MOV #BUF0,R0     ;LOAD BUFFER POINTER
3004 043740 005020 1$: CLR (R0)+       ;CLEAR THE DATA BUFFER
3005 043742 022700 064000      CMP #BUF2,R0     ;TEST IF FINISHED
3006 043746 001374      BNE 1$           ;BR IF NOT
3007 ;LOOK FOR THE DATA REGION FOR TO
3008 043750 004737 045630      JSR PC,STDATO    ;FIND THE NON-ZERO TO ZERO TRANS. DATA
3009 ;DATA HAS NOW GONE INTO THE GOOD 0 DATA REGION
3010 043754 012777 000000 135762 MOV #0,@KWPSR    ;INIT THE CLOCK PRESET
3011 043762 012777 000025 135750 MOV #25,@KWCSR   ;ENABLE THE CLOCK
3012 ;NOW TIME 'T1' LENGTH
3013 043770 004737 045474 2$: JSR PC,LISTDT  ;GET CURRENT DATA VALUE
3014 043774 022702 000000      CMP #0,R2        ;TEST FOR EXIT OF 0 DATA REGION
3015 044000 001773      BEQ 2$          ;BR UNTILL DONE
3016 044002 105077 135734      CLRB @KWCSR1
3017 044006 152777 000002 135726 BISB #BIT1,@KWCSR1 ;FIRE ST2
3018 044014 017737 135724 045666 MOV @KWPSR,DIFT1 ;SAVE COUNTER VALUE
3019 ;NOW TIME 'T2' LENGTH
3020 044022 004737 045474 3$: JSR PC,LISTDT  ;GET CURRENT DATA VALUE
3021 044026 022702 000377      CMP #377,R2     ;TEST FOR ENTRY INTO MAX DATA REGION

```

```

3022 044032 001373          BNE      3$          ;BR IF NOT
3023 044034 105077 135702    CLR      @KWCSR1
3024 044040 152777 000002    BIS      #BIT1,@KWCSR1 ;FIRE ST2
3025 044046 017737 135672    MOV      @KWPSR,DIFT2 ;SAVE COUNTER VALUE
3026                                ;NOW TIME 'T3' LENGTH
3027 044054 004737 045474    4$: JSR      PC,LISTDT ;GET CURRENT DATA VALUE
3028 044060 022702 000377    CMP      #377,R2      ;TEST FOR EXIT OF MAX DATA REGION
3029 044064 001773          BEQ      4$          ;BR IF NOT
3030 044066 105077 135650    CLR      @KWCSR1
3031 044072 152777 000002    BIS      #BIT1,@KWCSR1 ;FIRE ST2
3032 044100 017737 135640    MOV      @KWPSR,DIFT3 ;SAVE COUNTER VALUE
3033 044106 005077 135626    CLR      @KWCSR      ;STOP CLOCK
3034                                ;NOW DETERMINE THE AVG TIME (T3+T2)/2 AND SAVE IN 'HT2T3'
3035 044112 013700 045670    MOV      DIFT2,R0     ;GET T2 VALUE
3036 044116 063700 045672    ADD      DIFT3,R0     ;ADD T3 VALUE
3037 044122 006000          ROR      R0          ;/2
3038 044124 010037 045674    MOV      R0,HT2T3    ;SAVE FOR LATER USE
3039                                ;NOW RETURN TO LIST MODE AND LOOK FOR THE NON-ZERO TO ZERO EDGE AGAIN
3040 044130 004737 045630    DIFLN1: JSR      PC,STDATO ;FIND THE EDGE
3041                                ;NOW START THE CLOCK USING THE AVG TIME
3042                                ;AND DO AN MATRIX MODE COLLECTION
3043 044134 013700 045674    MOV      HT2T3,R0    ;GET AVG CLOCK
3044 044140 005400          NEG      R0          ;FIX FOR CLOCK PRESET REG
3045 044142 010077 135576    MOV      R0,@KWPSR   ;LOAD CLOCK PRESET
3046 044146 012777 000020    MOV      #20,@KWCSR ;PRIME CLOCK
3047 044154 012777 004000    MOV      #CLRALL,@SFR ;INIT THE NCV11
3048 044162 012777 060000    MOV      #BUFO,@OFF  ;LOAD OFFSET
3049 044170 005077 135556    CLR      @WCR       ;CLEAR Z COUNTER
3050 044174 005077 135554    CLR      @BAR
3051 044200 013777 046002    MOV      PRIME1,@CSR ;SET MODE AND RES.
3052 044206 052777 000022    BIS      #TESTZ!BIT4,@SFR ;SET TESTZ AND TESTX
3053 044214 052777 000001    BIS      #BIT0,@CSR  ;ENABLE NCV11
3054 044222 052777 000001    BIS      #BIT0,@KWCSR ;ENABLE CLOCK
3055 044230 105777 135504    1$: TSTB     @KWCSR   ;WAIT FOR CLOCK FLAG
3056 044234 100375          BPL      1$
3057 044236 052777 000400    BIS      #ENDDMA,@SFR ;STOP DMA XFR

```

```

3059 ;THE MATRIX MODE XFR IF NOW COMPLETED
3060 ;NOW ADD THE CURRENT BYTE DATA COLLECTION INTO A TOTAL INCREMENT BUFFER
3061 044244 005037 050014 CLR $TEMP1
3062 044250 012700 060000 MOV #BUF0,R0
3063 044254 012701 062000 MOV #BUF1,R1
3064
3065 044260 111037 050014 2$: MOVB (R0),$TEMP1 ;GET THE WORD
3066 044264 105020 CLR (R0)+ ;CLEAR THE BYTE TABLE ENTRY
3067 044266 063721 050014 ADD $TEMP1,(R1)+ ;UPDATE THE LIST
3068 044272 103003 BCC 3$ ;BR IF NO OVERFLOW
3069 044274 005741 TST -(R1) ;POSITON POINTER
3070 044276 012721 177777 MOV #-1,(R1)+ ;SET TO 177777
3071 044302 022701 064000 3$: CMP #BUF2,R1 ;FINISHED?
3072 044306 001364 BNE 2$ ;BR IF NOT
3073 ;GET THE AVERAGE OF THE 128 CENTER STATES
3074 044310 012700 062100 MOV #BUF1+64.,R0 ;GET INITIAL POINTER
3075 044314 005001 CLR R1
3076 044316 005002 CLR R2
3077 044320 062001 4$: ADD (R0)+,R1 ;GET A VALUE
3078 044322 103001 BCC 5$ ;BR IF NO CARRY OVERFLOW
3079 044324 005202 INC R2 ;UPDATE MSW
3080 044326 022700 062300 5$: CMP #BUF1+192.,R0 ;FINISHED THE BUFFER
3081 044332 001372 BNE 4$ ;BR IF NOT DONE AVERAGE
3086 044334 000241 CLC
(1) 044336 006002 ROR R2 ;ROTATE MSW
(1) 044340 006001 ROR R1 ;INTO LSW
(1) 044342 000241 CLC
(1) 044344 006002 ROR R2 ;ROTATE MSW
(1) 044346 006001 ROR R1 ;INTO LSW
(1) 044350 000241 CLC
(1) 044352 006002 ROR R2 ;ROTATE MSW
(1) 044354 006001 ROR R1 ;INTO LSW
(1) 044356 000241 CLC
(1) 044360 006002 ROR R2 ;ROTATE MSW
(1) 044362 006001 ROR R1 ;INTO LSW
(1) 044364 000241 CLC
(1) 044366 006002 ROR R2 ;ROTATE MSW
(1) 044370 006001 ROR R1 ;INTO LSW
(1) 044372 000241 CLC
(1) 044374 006002 ROR R2 ;ROTATE MSW
(1) 044376 006001 ROR R1 ;INTO LSW
3087 044400 010137 041714 MOV R1,AVGVAL ;SAVE THE AVERAGE
3088 044404 004737 045676 JSR PC,CTRLCG ;TEST FOR ^C OR ^G
3089 044410 023737 041714 041712 CMP AVGVAL,NOMIAL ;TEST IF AVG. IF >
3090 044416 103644 BLO DIFLN1 ;BR IF NOT
3091

```

```

3093 ;READ THE TOTAL INCREMENT BUFFER AND DETERMINE IF ANY VALUES OUT OF RANGE
3094
3095 044420 013737 041714 045114 READ1: MOV AVGLVAL,101$ ;GET AVERAGE
3096 044426 012700 000001 MOV #1,R0
3097 044432 012701 062002 MOV #BUF1+2,R1
3098 044436 012137 045112 1$: MOV (R1)+,100$ ;GET A WORD
3099 044442 163737 045114 045112 SUB 101$,100$ ;REMOVE THE AVERAGE
3100 044450 100006 BPL 2$
3101 044452 005137 045112 COM 100$
3102 044456 112737 000055 041554 MOVB #'-,SGNVL1 ;INSERT '-' SIGN
3103 044464 000403 BR 3$
3104 044466 112737 000053 041554 2$: MOVB #'+,SGNVL1 ;INSERT '+' SIGN
3105 044474 013746 045112 3$: MOV 100$,-(SP)
3106 044500 012746 001750 MOV #1000,-(SP)
3107 044504 004737 036506 JSR PC,$MULT ;MULTIPLY NUM-AVG BY 1000.
3108 044510 012637 045130 MOV (SP)+,107$ ;GET RESULT
3109 044514 012637 045132 MOV (SP)+,110$
3110 044520 013746 045130 MOV 107$,-(SP) ;DIVIDE RESULT
3111 044524 013746 045132 MOV 110$,-(SP)
3112 044530 013746 045114 MOV 101$,-(SP) ;DIVIDE AGAIN
3113 044534 004737 036620 JSR PC,$DIV
3114 044540 102003 BVC 4$
3115 044542 000000 HALT ;FATAL ERROR DURING CAL. OF ERROR TOLERANCES
3116 044544 000240 NOP
3117 044546 000240 NOP
3118 044550 012637 045122 4$: MOV (SP)+,104$ ;GET REMAINDER
3119 044554 012637 045124 MOV (SP)+,105$ ;GET QOUT
3120 044560 013702 045114 MOV 101$,R2 ;ROUND UP IF NEEDED
3121 044564 006202 ASR R2
3122 044566 023702 045122 CMP 104$,R2 ;COMPARE RESULT
3123 044572 002402 BLT 5$
3124 044574 005237 045124 INC 105$ ;ROUND UP
3125 044600 000240 5$: NOP
3126 ;DIFLIN ERROR CHECK
3127
3128 044602 023737 045124 046054 6$: CMP 105$,NORTOL ;TEST AGAINST NORMAL
3129 044610 103446 BLO 50$ ;BR IF OK
3130 044612 023737 045124 046060 CMP 105$,NARTOL ;TEST AGAINST WIDE/NARROW TOLERANCE
3131 044620 103416 BLO 20$ ;BR IF OK
3132 044622 023737 045124 046064 CMP 105$,NURTOL ;TEST AGAINST LARGE/SMALL TOLERANCE
3133 044630 103424 BLO 21$ ;BR IF OK
3134 044632 122737 000053 041554 CMPB #'+,SGNVL1 ;TEST IF +
3135 044640 001403 BEQ 7$
3136 044642 005237 041720 INC BELOW ;COUNT THE LOWER VALUE
3137 044646 000435 BR 51$
3138 044650 005237 041722 7$: INC ABOVE ;COUNT THE HIGHER VALUE
3139 044654 000432 BR 51$
3140 044656 122737 000053 041554 20$: CMPB #'+,SGNVL1 ;TEST IF +
3141 044664 001403 BEQ 22$
3142 044666 005237 041732 INC NARB ;COUNT THE LOWER
3143 044672 000423 BR 51$
3144 044674 005237 041736 22$: INC WIDB ;COUNT THE HIGHER
3145 044700 000420 BR 51$
3146 044702 122737 000053 041554 21$: CMPB #'+,SGNVL1 ;TEST IF +
3147 044710 001403 BEQ 23$
3148 044712 005237 041730 INC NARA ;COUNT THE LOWER

```

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 94-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0142

```

3149 044716 000411
3150 044720 005237 041734
3151 044724 000406
3152 044726 005237 041724
3153 044732 122737 000115 052410
3154 044740 001054
3155 044742 105737 001134
3156 044746 001051
3157
3158 044750 005737 041744
3159 044754 001013
3160 044756 005237 041744
3161 044762 104401 041414
3162 044766 013746 041714
3163 044772 104405
3164 044774 104401 041566
3165 045000 104401 041100
3166 045004 010046
3167 045006 104403
3168 045010 003 001
3169 045012 122737 000115 052410
3170 045020 001005
3171 045022 104401 040745
3172 045026 013746 045112
3173 045032 104405
3174 045034 013746 045124
3175 045040 004737 037234
3176 045044 012602
3177 045046 062702 000002
3178 045052 112237 041556
3179 045056 112237 041557
3180 045062 112237 041561
3181 045066 104401 041546
3182
3183 045072 004737 045676
3184 045076 005200
3185 045100 022700 000377
3186 045104 001414
3187 045106 000137 044436
3188
3189 045112 000000
3190 045114 000000
3191 045116 000000
3192 045120 000000
3193 045122 000000
3194 045124 000000
3195 045126 000000
3196 045130 000000
3197 045132 000000
3198 045134 000000

23$: BR 51$
INC WIDA ;COUNT THE HIGHER
BR 51$
50$: INC NORMAL ;COUNT THE NORMAL
CMPB #'M,RUNIT ;TEST IF FORCE REPORT
BNE 77$
51$: TSTB $AUTOB ;TEST IF MONITOR ?
BNE 77$ ;BR IF YES
;REPORT THE STATE INFORMATION
TST FIRST ;FIRST TYPEOUT ?
BNE 10$ ;BR IF YES
INC FIRST ;SET FLAG
TYPE ,AVRGO ;TELL THE OPFRATOR THE AVERG.
MOV AVGVAL,-(SP)
TYPE, RTN
TYPE, STATE ;ADD HEADER INFO
10$: MOV RO,-(SP) ;LOAD STATE #
TYPOS
.BYTE 3,1
CMPB #'M,RUNIT ;TEST IF EXPAND OUTPUT SELECTED
BNE 11$
TYPE, DASH ;INSERT SPACING
MOV 100$,-(SP) ;REPORT DIFFERENCE
TYPDS
11$: MOV 105$,-(SP)
JSR PC,$$B2D ;CONVERT TO ASCII
MOV (SP)+,R2 ;GET RESULT POINTER TO MESSAGE
ADD #2,R2 ;ADJUST POINTER
MOVB (R2)+,PERTXT
MOVB (R2)+,PERTXT+1 ;LOAD THE PERCENT REPORT
MOVB (R2)+,PERTXT+3
TYPE, SGNVAL
77$: JSR PC,CTRLCG ;TEST IF CTRL C/G
INC RO
CMP #255.,RO ;TEST IF DONE
BEQ READ ;BR IF DONE
JMP 1$ ;TRY NEXT VALUE

100$: 0
101$: 0
102$: 0
103$: 0
104$: 0
105$: 0
106$: 0
107$: 0
110$: 0
111$: 0

```

```

3200
3201      ;NOW ACCOUNT FOR ALL THE ERRORS
3202      READ:  CLR      DIFERR      ;CLEAR FLAG
3203      MOV      ABOVE,R2      ;GET HIGH VALUE
3204      ADD      BELOW,R2      ;ADD LOW
3205      CMPB     R2,OBSCNT      ;TEST IF EXCEEDS LIMIT <OBEASE ABOVE/BELOW>
3206      BHI      1$            ;BR IF ERRORS
3207      MOV      NARA,R2      ;GET LOW VALUE
3208      ADD      WIDA,R2      ;ADD HIGH
3209      CMPB     R2,NURCNT      ;TEST IF EXCEEDS LIMIT <LARGE ABOVE/BELOW>
3210      BHI      1$
3211      MOV      NARB,R2      ;GET LOW VALUE
3212      ADD      WIDB,R2      ;ADD HIGH
3213      CMPB     R2,NARCNT      ;TEST IF EXCEEDS LIMIT <SMALL ABOVE/BELOW>
3214      BHI      1$
3215      CMPB     NORCNT,NORMAL ;TEST IF MIN. NORMAL COUNT HAS BEEN HIT
3216      BHI      1$
3217      MOV      #PASMSG,READ7 ;LOAD MESSAGE POINTER
3218      BR       2$
3219      MOV      #FAIMSG,READ7 ;LOAD MESSAGE POINTER
3220      INC      DIFERR      ;SET ERROR FLAG
3221      MOV      BELOW,-(SP)    ;GET NO. OF STATES BELOW LIMITS
3222      TYPDS     ;REPORT VALUE
3223      TYPE      ,BELMSG      ;TYPE MESSAGE
3224      MOV      NARA,R2      ;GET NARROW LOW
3225      ADD      NARB,R2      ;ADD NARROW HIGH
3226      MOV      R2,-(SP)      ;GET NO. OF NARROW STATES
3227      TYPDS     ;REPORT VALUE
3228      TYPE      ,NARMSG      ;TYPE MESSAGE
3229      MOV      NORMAL,-(SP)  ;GET NORMAL COUNT
3230      TYPDS     ;REPORT VALUE
3231      TYPE      ,NORMSG      ;REPORT NORMAL TEXT
3232      MOV      WIDA,R2      ;GET WIDE LOW
3233      ADD      WIDB,R2      ;ADD WIDE HIGH
3234      MOV      R2,-(SP)
3235      TYPDS     ;REPORT VALUE
3236      TYPE      ,WIDMSG      ;TYPE MESSAGE
3237      MOV      ABOVE,-(SP)
3238      TYPDS     ;TYPE NO. OF STATES ABOVE LIMIT
3239      TYPE      ,ABOMSG      ;TYPE MESSAGE
3240
3241      ;REPORT PASS/FAIL MESSAGE
3242
3243      TYPE      ,GAIN         ;RE-TYPE THE GAIN MESSAGE
3244      TYPE
3245      READ7:  PASMSG         ;REPORT PASS/FAIL
3246

```



```
3248  
3249 ;TYPE OUT MEMORY RAM AND RAM MEMORY VALUES IF SELECTED  
3250 045354 032777 004000 133556 READXX: BIT #SW11,@SWR ;TEST IF THIS IS WANTED  
3251 045362 001440 BEQ 4$ ;BR IF NOT  
3252 045364 005737 041706 TST DIFEX1 ;TEST IF BLASTING MODE  
3253 045370 001435 BEQ 4$ ;BR IF NOT  
3254  
3255 045372 012700 042132 MOV #ROMVAL,R0 ;LOAD INITIAL POINTER  
3256 045376 004737 045426 1$: JSR PC,2$ ;TYPE OUT LINE  
3257 045402 022700 042232 CMP #ROMVAL+100,R0 ;TEST IF ALL DONE  
3258 045406 001373 BNE 1$ ;BR IF NOT  
3259 045410 104401 041566 TYPE, RTN  
3260 045414 104401 041566 TYPE, RTN  
3261 045420 104401 041566 TYPE, RTN  
3262 045424 000417 BR 4$  
3263  
3264 045426 012701 000020 2$: MOV #16.,R1 ;LOAD WIDE COUNTER  
3265 045432 104401 041566 TYPE, RTN  
3266 045436 112002 3$: MOVB (R0)+,R2 ;GET A BYTE  
3267 045440 042702 177400 BIC #177400,R2 ;MASK OFF HIGHER BITS  
3268 045444 010246 MOV R2,-(SP) ;LOAD VALUE  
3269 045446 104403 TYPOS  
3270 045450 003 001 .BYTE 3,1  
3271 045452 104401 040750 TYPE, DASH+3  
3272 045456 005301 DEC R1 ;FINISHED ?  
3273 045460 001366 BNE 3$  
3274 045462 000207 RTS PC ;EXIT  
3275  
3276  
3277 045464 005037 041706 4$: CLR DIFEX1  
3278 045470 000177 000310 JMP @DIFEX0 ;EXIT THIS CRAZYNES  
3279
```

```

3281          :DIF LIN SUBROUTINE LIST MODE DATA COLLECTOR
3282
3283 045474 012777 004000 134254 LISTDT: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3284 045502 005077 134242          CLR @OFF ;PRIME OFFSET REG.
3285 045506 012777 177770 134236          MOV #-8.,@WCR ;LOAD WC.
3286 045514 012777 060000 134232          MOV #BUF0,@BAR ;LOAD BUS ADDR.
3287 045522 012777 000000 134216          MOV #0,@CSR ;SET RES. AND MODE
3288 045530 032737 002000 046002          BIT #2000,PRIME1 ;TEST GAIN BIT
3289 045536 001403          BEQ 3$ ;BR IF CLEARED
3290 045540 052777 002000 134200          BIS #2000,@CSR ;SELECT GAIN = 2
3291 045546 052777 000002 134202 3$: BIS #TESTZ,@SFR ;SET Z PULSES
3292 045554 052777 000001 134164          BIS #BIT0,@CSR ;ENABLE THE DEVICE
3293
3294 045562 105777 134160          1$: TSTB @CSR ;WAIT FOR IDLE
3295 045566 100375          BPL 1$
3296 045570 012700 06000C          MOV #BUF0,R0 ;LOAD POINTER TO NEW DATA
3297 045574 005002          CLR R2
3298 045576 012703 000010          MOV #8.,R3 ;LOAD COUNTER
3299 045602 011001          2$: MOV (R0),R1 ;GET DATA WORD
3300 045604 005020          CLR (R0)+ ;CLR BUFFER WORD
3301 045606 042701 177400          BIC #177400,R1 ;MAKE OFF BITS
3302 045612 060102          ADD R1,R2 ;UPDATE TOTAL
3303 045614 005303          DEC R3 ;FINISHED ?
3304 045616 001371          BNE 2$ ;BR IF NOT
3305 045620 006202          ASR R2
3306 045622 006202          ASR R2
3307 045624 006202          ASR R2
3308 045626 000207          RTS PC ;EXIT
3309          :SUBROUTINE TO FIND THE TRUE EDGE OF ZERO DATA
3310 045630 012737 177777 045664 STDATO: MOV #-1,10$ ;SET ENTRY FLAG
3311 045636 004737 045474          1$: JSR PC,LISTDT ;GET DATA
3312 045642 005702          TST R2 ;CHECK DATA
3313 045644 001403          BEQ 2$ ;BR IF ZERO
3314 045646 005037 045664          CLR 10$ ;CLEAR ALREADY NON-ZERO DATA FLAG
3315 045652 000771          BR 1$
3316          :DATA WAS ZERO - CHECK IF WE STARTED IN 0 DATA REGION
3317 045654 005737 045664          2$: TST 10$ ;TEST FLAG
3318 045660 001366          BNE 1$ ;BR IF NOT A GOOD TIME TO EXIT
3319          :DATA HAS NOW GONE TO A GOOD ZERO DATA REGION
3320          :NOW EXIT
3321 045662 000207          RTS PC ;BYEBYE
3322 045664 000000          10$: 0
3323 045666 000000          DIFT1: 0
3324 045670 00000C          DIFT2: 0
3325 045672 000000          DIFT3: 0
3326 045674 000000          HT2T3: 0
  
```

```
3328 ;SUBROUTINE TO TEST IF OPERATOR TYPED CTRL C OR G
3329 045676 10577' 133242 CTRLCG: TSTB @STKS ;INPUT FLAG
3330 045702 100033 BPL 2$ ;BR IF NON
3331 045704 017737 133236 045774 MOV @STKB,CTRCHA ;GET CHARACTER
3332 045712 042737 177600 045774 BIC #177600,CTRCHA ;MASK OFF BITS
3333 045720 022737 000003 045774 CMP #3,CTRCHA ;TEST IF CTRL C
3334 045726 001014 BNE 1$ ;BR IF NOT ^C
3335 045730 052777 000400 134020 BIS #ENDDMA,@SFR ;STOP NPR'S
3336 045736 000240 NOP
3337 045740 000240 NOP
3338 045742 000240 NOP
3339 045744 052777 004000 134004 BIS #CLRALL,@SFR ;CLEAR THE DEVICE
3340 045752 005726 TST (SP)+ ;CLEAN STACK
3341 045754 000137 002014 JMP RTRT ;RE-START
3342 045760 022737 000007 045774 1$: CMP #7,CTRCHA ;TEST IF CTRL G
3343 045766 001001 BNE 2$ ;BR IF NOT
3344 045770 104406 GTSWR ;GET SWR VALUE
3345 045772 000207 2$: RTS PC ;EXIT
3346 045774 000000 CTRCHA: 0
3347 045776 177777 PRIME: -1.
3348 046000 177400 PRIME0: 177400
3349 046002 000036 PRIME1: 36
3350 046004 000000 DIFEX0: 0
3351
3352 ;SUBROUTINE TO DETERMINE ERROR LIMITS FOR DIFLIN
3353 046006 012500 LIMITS: MOV (R5)+,R0 ;GET 1ST ARG.
3354 046010 005737 050024 TST WFMODE ;TEST IF ON TESTER
3355 046014 001402 BEQ 1$ ;BR IF NOT
3356 046016 012500 MOV (R5)+,R0 ;GET TESTER LIMITS
3357 046020 000401 BR 2$
3358 046022 005725 1$: TST (R5)+ ;BUMP ADDRESS
3359 046024 012701 046052 2$: MOV #NORCNT,R1 ;GET POINTER
3362 046030 012021 MOV (R0)+,(R1)+ ;GET VALUE
(1) 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
3363 046050 000205 RTS R5 ;EXIT
```

```
3365 :ACTUAL VALUES FOR DIF LIN ERROR TOLERANCE
3366
3367
3368 046052 000342 NORCNT: .WORD 226. :'NORMAL'' MIN COUNT VALUE
3369 046054 000025 NORTOL: .WORD 21. :'NORMAL'' TOLERANCE VALUE
3370 046056 000024 NARCNT: .WORD 20. :'NARROW/WIDE'' MAX. COUNT VALUE
3371 046060 000063 NARTOL: .WORD 51. :'NARROW/WIDE'' TOLERANCE VALUE
3372 046062 000010 NURCNT: .WORD 8. :'LARGE/SMALL'' MAX. COUNT VALUE
3373 046064 000145 NURTOL: .WORD 101. :'LARGE/SMALL'' TOLERANCE VALUE
3374 046066 000000 OBSCNT: .WORD 0. :'OBESITY'' MAX. COUNT VALUE
3375 046070 000377 OBSTOL: .WORD 377 :'OBESITY'' TOLERANCE VALUE
3376
3377 :DIFLIN ERROR TOLERANCE GAIN = 1 USER
3378
3379 046072 000342 000025 G1LIMO: .WORD 226.,21. :226 MIN. COUNT, =<2.0%
3380 046076 000024 000063 :.WORD 20.,51. :20 MAX. COUNT, =<5.0%
3381 046102 000010 000145 :.WORD 8.,101. :8 MAX. COUNT, =<10.0%
3382 046106 000000 000377 :.WORD 0.,377 :0 MAX COUNT, >10%
3383
3384 :DIFLIN ERROR TOLERANCE GAIN = 1 OPTION AREA
3385
3386 046112 000342 000024 G1LIM1: .WORD 226.,20. :226 MIN COUNT, -<1.9%
3387 046116 000024 000061 :.WORD 20.,49. :20 MAX. COUNT, -<4.8%
3388 046122 000010 000140 :.WORD 8.,96. :8 MAX COUNT, =<9.5%
3389 046126 000000 000377 :.WORD 0.,377 :0 MAX COUNT, >9.5%
3390
3391 :DIFLIN ERROR TOLERANCE GAIN = 2 USER
3392
3393 046132 000330 000031 G2LIMO: .WORD 216.,25. :216. MIN COUNT, =<2.4%
3394 046136 000036 000075 :.WORD 30.,61. :30. MAX COUNT, =<6.0%
3395 046142 000010 000171 :.WORD 8.,121. :8. MAX COUNT, =<12.0%
3396 046146 000000 000377 :.WORD 0.,377 :0 MAX COUNT, >12.0%
3397
3398 :DIFLIN ERROR TOLERANCE GAIN = 2 OPTION AREA
3399
3400 046152 000330 000025 G2LIM1: .WORD 216.,21. :216. MIN COUNT, =<2.0%
3401 046156 000036 000063 :.WORD 30.,51. :30. MAX COUNT, =<5.0%
3402 046162 000010 000145 :.WORD 8.,101. :8. MAX COUNT, =<10.0%
3403 046166 000000 000377 :.WORD 0.,377 :0 MAX COUNT, >10.0%
3404
```

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 100
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0148

3406	046172	015	012	
3407	046174	020111	020075	047111
	046202	052111	040511	020114
	046210	030101	033461	040440
	046216	045104	051525	046524
	046224	047105	020124	051525
	046232	047111	020107	044124
	046240	020105	047101	046101
	046246	043517	052040	051505
	046254	042524	122	
3408	046257	015	012	
3409	046261	102	036440	041040
	046266	040514	052123	047111
	046274	020107	043117	052040
	046302	042510	046040	047111
	046310	040505	044522	054524
	046316	041440	051117	042522
	046324	052103	047511	020116
	046332	051120	046517	
3410	046336	015	012	
3411	046340	020103	020075	047503
	046346	052116	047522	020114
	046354	051120	043517	040522
	046362	020115	051120	046517
	046370	041040	040514	052123
	046376	047111	000107	
3412	046402	015	012	
3413	046404	020114	020075	047514
	046412	044507	020103	042524
	046420	052123	047440	020106
	046426	034115	031060	020066
	046434	047101	020104	034115
	046442	031460	066	
3414	046445	015	012	
3415	046447	104	036440	042040
	046454	043111	042506	042522
	046462	052116	040511	020114
	046470	044514	042516	051101
	046476	052111	020131	043117
	046504	052040	042510	040440
	046512	030460	067	
3416	046515	015	012	
3417	046517	106	036440	043040
	046524	047111	046101	040440
	046532	041503	050105	040524
	046540	041516	020105	046050
	046546	040440	042116	042040
	046554	051		
3418	046555	015	012	
3419	046557	101	036440	040440
	046564	045104	051525	046524
	046572	047105	020124	043117
	046600	040440	030460	020067
	046606	052101	043040	042511
	046614	042114	051440	052111
	046622	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/

CZNCB NCV11 D.AGNOSTIC
CZNCB.P 1 31-AUG-79 11.21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 100-1
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0149

3420	046623	015	012	
3421	046625	117	036440	047440
	046632	042520	040522	042524
	046640	053440	052111	020110
	046646	052124	020131	052101
	046654	040440	042104	042522
	046662	051523	047040	047116
	046670	047116	116	
3422	046673	015	012	
3423	046675	123	036440	051440
	046702	043117	053524	051101
	046710	020105	053523	052111
	046716	044103	051040	043505
	046724	051511	042524	020122
	046732	044103	047101	042507
3424	046740	015	012	
3425	046742	020110	020075	042510
	046750	050114	052040	042510
	046756	047440	042520	040522
	046764	047524	020122	047101
	046772	020104	042522	054524
	047000	042520	052040	044510
	047006	020123	044514	052123
	047014	000		
3426	047015	015	012	
3427	047017	056	000	
3428				
3429	047022			
3430				

.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

```

3432
3433 047022 000005          POTIME: RESET          :CLEAR THE WORLD
3434 047024 104401 052156 132650  TYPE, PRIM2           :TELL OPERATOR ABOUT CABLES
3435 047030 012777 006035          MOV #6035,@VTCHAR      :HOME AND ERASE SCREEN
3436 047036 005077 132646          CLR @VTYPOS           :CLEAR THE MAP P.C.
3437 047042 005037 045774          CLR CTRCHA           :CLEAR TTY CHARACTER
3438
3439          ;ERASE THE SCREEN MAP
3440 047046 105777 132644 1$:  TSTB @VTCSR           :WAIT FOR READY
3441 047052 100375          BPL 1$              :
3442 047054 052777 001000 132634  BIS #BIT9,@VTCSR      :CLEAR THE MAP
3443 047062 105777 132630 2$:  TSTB @VTCSR           :WAIT FOR READY AGAIN
3444 047066 100375          BPL 2$              :
3445 047070 042777 001000 132620  BIC #BIT9,@VTCSR      :CLEAR THE ERASE BIT
3446          ;NOW LOAD THE REF. PATTERN INTO THE MAP
3447 047076 012701 010421          MOV #10421,R1        :LOAD THE PIXEL VALUE
3448 047102 012700 003740          MOV #3740,R0         :LOAD THE PIXFL ADDRESS
3449 047106 004737 047760 3$:  JSR PC,DISPLY        :LOAD THE DATA
3450 047112 005200          INC R0              :UPDATE THE ADDRESS
3451 047114 022700 004000          CMP #4000,R0        :FINISHED ?
3452 047120 001372          BNE 3$             :BR UNTIL DONE
3453
3454 047122 012700 000000          MOV #0,R0           :LOAD PIXEL ADDRESS
3455 047126 004737 047760 4$:  JSR PC,DISPLY        :LOAD THE DATA
3456 047132 005200          INC R0              :UPDATE THE PIXEL ADDRESS
3457 047134 022700 000040          CMP #40,R0         :TEST IF DONE
3458 047140 001372          BNE 4$             :BR IF DONE THE VERTICAL
3459
3460 047142 012700 007740          MOV #7740,R0        :LOAD PIXEL ADDRESS
3461 047146 004737 047760 5$:  JSR PC,DISPLY        :LOAD THE DATA
3462 047152 005200          INC R0              :UPDATE THE PIXEL ADDRESS
3463 047154 022700 010000          CMP #10000,R0      :TEST IF DONE
3464 047160 001372          BNE 5$             :BR IF DONE
3465
3466          ;POSITION THE MAP AND DISPLAY
3467
3468 047162 012777 000005 132526  MOV #5,@VTCSR        :LOAD ORGIN VALUE
3469 047170 012777 000060 132526  MOV #60,@VTINT       :LOAD LUT 0
3470 047176 012777 000463 132520  MOV #463,@VTINT      :LOAD LUT 1
3471 047204 052777 000400 132504  BIS #BIT8,@VTCSR     :ENABLE THE MAP

```

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 102
A TO D FIELD SITE AND ADJUSTMENT LOOP

SEQ 0151

```

3473 ;NOW SAMPLE THE CAMERA CHANNELS
3474 047212 SAMCAM:
3475 ;CAMERA 0, X INPUT
3476 047212 004537 047336 JSR R5,ADJCAM
3477 047216 050042 ADMSG0 ;CAM 0 MESSAGE
3478 047220 001730 DAC0 ;DAC ADDRESS
3479 047222 000 000 .BYTE 0,0 ;CAMERA 0, X AXIS
3480 ;DO CAMERA 1, X INPUT
3481 047224 004537 047336 JSR R5,ADJCAM
3482 047230 050226 ADMSG2 ;CAM 1, X MESSAGE
3483 047232 001732 DAC1 ;DAC ADDRESS
3484 047234 001 000 .BYTE 1,0 ;CAMERA 1, X AXIS
3485 ;DO CAMERA 2, X INPUT
3486 047236 004537 047336 JSR R5,ADJCAM
3487 047242 050412 ADMSG4 ;CAM 2, X MESSAGE
3488 047244 001734 DAC2 ;DAC ADDRESS
3489 047246 002 000 .BYTE 2,0 ;CAMERA 2, X AXIS
3490 ;DO CAMERA 3, X INPUT
3491 047250 004537 047336 JSR R5,ADJCAM
3492 047254 050576 ADMSG6 ;CAM 3, X MESSAGE
3493 047256 001736 DAC3 ;DAC ADDRESS
3494 047260 003 000 .BYTE 3,0 ;CAMERA 3, X AXIS
3495 ;NOW SAMPLE THE Y CAMERA CHANNELS
3496 ;DO CAMERA 0 Y INPUT
3497 047262 004537 047336 JSR R5,ADJCAM
3498 047266 050134 ADMSG1 ;CAM 0 Y MESSAGE
3499 047270 001732 DAC1 ;DAC ADDRESS
3500 047272 000 001 .BYTE 0,1 ;CAMERA 0, Y AXIS
3501 ;DO CAMERA 1, Y INPUT
3502 047274 004537 047336 JSR R5,ADJCAM
3503 047300 050320 ADMSG3 ;CAM 1, Y MESSAGE
3504 047302 001730 DAC0 ;DAC ADDRESS
3505 047304 001 001 .BYTE 1,1 ;CAMERA 1, Y AXIS
3506 ;DO CAMERA 2, Y INPUT
3507 047306 004537 047336 JSR R5,ADJCAM
3508 047312 050504 ADMSG5 ;CAM 2, Y MESSAGE
3509 047314 001736 DAC3 ;DAC ADDRESS
3510 047316 002 001 .BYTE 2,1 ;CAMERA 2, Y AXIS
3511 ;DO CAMERA 3, Y INPUT
3512 047320 004537 047336 JSR R5,ADJCAM
3513 047324 050670 ADMSG7 ;CAM 3, Y MESSAGE
3514 047326 001734 DAC2 ;DAC ADDRESS
3515 047330 003 001 .BYTE 3,1 ;CAMERA 3 Y AXIS
3516 ;
3517 047332 000137 002646 JMP RBEGO

```



```

3519 ;REPORT THE MESSAGE FIRST
3520
3521 047336 012500 ADJCAM: MOV (R5)+,R0 ;GET ASCII POINTER
3522 047340 005777 132342 3$: TST @VTCHAR ;WAIT FOR READY
3523 047344 100375 BPL 3$
3524 047346 012777 012000 132340 MOV #12000,@VTCXY ;POSITION THE CAHRACTERS
3525 047354 005777 132326 1$: TST @VTCHAR ;WAIT FOR CHAR. READY
3526 047360 100375 BPL 1$
3527 047362 112077 132320 MOVB (R0)+,@VTCHAR ;LOAD THE CHAR.
3528 047366 105710 TSTB (R0) ;TEST IF TERM.
3529 047370 001371 BNE 1$ ;BR IF NOT
3530
3531 047372 013537 050020 MOV @ (R5)+,DACADR ;SAVE ADDRESS
3532 047376 012537 050026 MOV (R5)+,CAMVAL ;SAVE CAMERA DATA
3533
3534 047402 012777 002315 000410 2$: MOV #2315,@DACADR ;PRESET DAC TO - GAIN INPUT
3535 047410 004737 047512 JSR PC,SAMDAT
3536 047414 010137 050034 MOV R1,VAL0 ;SAVE RESULTS
3537 047420 004737 047736 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3538 047424 012777 004000 000366 MOV #4000,@DACADR ;PRESET DAC TO OFFSET
3539 047432 004737 047512 JSR PC,SAMDAT ;SAMPLE THE DATA
3540 047436 010137 050036 MOV R1,VAL1 ;SAVE AVERAGE RESULTS
3541 047442 004737 047736 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3542 047446 012777 005463 000344 MOV #5463,@DACADR ;PRESET DAC TO + GAIN INPUT
3543 047454 004737 047512 JSR PC,SAMDAT ;SAMPLE THE DATA
3544 047460 010137 050040 MOV R1,VAL2 ;SAVE AVERAGE RESULTS
3545 047464 004737 047736 JSR PC,FXLIN ;POSITION THE CROSS HAIR LINE
3546 047470 004737 045676 JSR PC,CTRLCG ;TEST FOR CTRL C/G
3547 047474 122737 000103 045774 CMPB #'C,CTRCHA ;CHECK IF CHARACTER WAS A \C\
3548 047502 001337 BNE 2$ ;BR IF NOT
3549 047504 005037 045774 CLR CTRCHA ;CLEAR CHAR
3550 047510 000205 RTS R5 ;EXIT
3551
3552 ;COLLECT 64 SAMPLES FROM THE SELECTED CAMERA
3553 ;R1 CONTAINS THE AVERAGE
3554
3555 047512 012700 060000 SAMDAT: MOV #BUFO,R0
3556 047516 012701 062000 MOV #BUF1,R1
3557 047522 005020 1$: CLR (R0)+ ;CLEAR THE BUFFER
3558 047524 020001 CMP R0,R1 ;FINISHED ?
3559 047526 001375 BNE 1$ ;BR IF NOT
3560
3561 047530 012777 177700 132214 MOV #-100,@WCR ;LOAD WORD COUNT
3562 047536 012777 060000 132210 MOV #BUFO,@BAR ;LOAD ADDRESS
3563 047544 113777 050026 132214 MOVB CAMVAL,@CSRHB ;SELECT THE CAMERA
3564 047552 052777 000002 132176 BIS #TESTZ,@SFR ;SET TEST Z ENABLE
3565 047560 052777 000001 132160 BIS #BIT0,@CSR ;ENABLE NCV11
3566
3567 047566 105777 132154 2$: TSTB @CSR ;WAIT UNTIL DONE
3568 047572 100375 BPL 2$
3569
3570 047574 012777 004000 132154 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3571 047602 005037 050014 CLR $TEMP1
3572 047606 005037 050016 CLR $TEMP2
3573 047612 005001 CLR R1
3574 047614 012700 060000 MOV #BUFO,R0 ;LOAD BUFFER POINTER

```

```

3575 047620 105737 050027          TSTB   CAMVAL+1      ;TEST IF X OR Y DATA
3576 047624 001413          BEQ    5$            ;BR IF X
3577 047626 012037 050014          MOV    (R0)+,$TEMP1 ;GET THE DATA
3578 047632 113737 050015 050016 3$:  MOVB  $TEMP1+1,$TEMP2 ;GET HIGH BYTE DATA
3579 047640 063701 050016          ADD    $TEMP2,R1    ;UPDATE AVERAGE
3580 047644 022700 060200          CMP    #BUF0+200,R0 ;FINISHED DATA ?
3581 047650 001366          BNE    3$           ;BR IF NOT
3582 047652 000412          BR     10$
3583 047654 012037 050014          MOV    (R0)+,$TEMP1 ;GET DATA
3584 047660 113737 050014 050016 5$:  MOVB  $TEMP1,$TEMP2
3585 047666 063701 050016          ADD    $TEMP2,R1    ;UPDATE THE AVERAGE
3586 047672 022700 060200          CMP    #BUF0+200,R0 ;FINISHED DATA ?
3587 047676 001366          BNE    5$           ;BR IF NOT
3588 047700 000240          10$:  NOP
3589 047702 000240          NOP
3593 047704 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047706 000240          NOP
(1) 047710 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047712 000240          NOP
(1) 047714 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047716 000240          NOP
(1) 047720 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047722 000240          NOP
(1) 047724 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047726 000240          NOP
(1) 047730 006201          ASR    R1            ;AVERAGE THE DATA
(1) 047732 000240          NOP
3594 047734 000207          RTS     PC           ;EXIT
3595
3596          ;SUBROUTINE TO LOAD X OR Y CROSSHAIRS
3597 047736 006201          FIXLIN: ASR    R1
3598 047740 000240          NOP
3599 047742 000240          NOP
3600 047744 000240          NOP
3601 047746 062701 000100          ADD    #100,R1
3602 047752 110177 131734          MOVB  R1,@VTXPOS
3603 047756 000207          RTS     PC
3604
3605          ;SUBROUTINE TO LOAD PIXEL DATA
3606 047760 042777 000400 131730  DISPLY: BIC    #400,@VTCSR ;DISABLE MAP
3607 047766 010077 131726          MOV    R0,@VTMAP    ;LOAD MAP PC
3608 047772 105777 131720          1$:  TSTB  @VTCSR        ;WAIT FOR MAP READY
3609 047776 100375          BPL    1$
3610 050000 010177 131716          MOV    R1,@VTPX     ;LOAD PIXEL DATA
3611 050004 052777 000400 131704  BIS    #400,@VTCSR   ;ENABLE MAP
3612 050012 000207          RTS     PC
3613
3614 050014 000000          $TEMP1: 0
3615 050016 000000          $TEMP2: 0
3616 050020 000000          DACADR: 0
3617 050022 000000          RUNDIF: 0
3618 050024 000000          WFMODE: 0
3619 050026 000000          CAMVAL: 0
3620 050030 000000          CURENT: 0
3621 050032 000000          AGAN: 0
3622 050034 000000          VAL0: 0
;WF OPTION AREA FLAG

```

3623	050036	000000			VAL1: 0
3624	050040	000000			VAL2: 0
3625	050042	040503	042515	040522	ADMSG0: .ASCII /CAMERA 00/
	050050	030040	060		
3626	050053	015	012		.BYTE 15,12
3627	050055	101	045104	051525	.ASCII /ADJUST R09 FOR X OFFSET/
	050062	020124	030122	020071	
	050070	047506	020122	020130	
	050076	043117	051506	052105	
3628	050104	015	012		.BYTE 15,12
3629	050106	042101	052512	052123	.ASCIIZ /ADJUST R24 FOR X GAIN/
	050114	051040	032062	043040	
	050122	051117	054040	043440	
	050130	044501	000116		
3630	050134	040503	042515	040522	ADMSG1: .ASCII /CAMERA 00/
	050142	030040	060		
3631	050145	015	012		.BYTE 15,12
3632	050147	101	045104	051525	.ASCII /ADJUST R13 FOR Y OFFSET/
	050154	020124	030522	020063	
	050162	047506	020122	020131	
	050170	043117	051506	052105	
3633	050176	015	012		.BYTE 15,12
3634	050200	042101	052512	052123	.ASCIIZ /ADJUST R20 FOR Y GAIN/
	050206	051040	030062	043040	
	050214	051117	054440	043440	
	050222	044501	000116		
3635	050226	040503	042515	040522	ADMSG2: .ASCII /CAMERA 01/
	050234	030040	061		
3636	050237	015	012		.BYTE 15,12
3637	050241	101	045104	051525	.ASCII /ADJUST R10 FOR X OFFSET/
	050246	020124	030522	020060	
	050254	047506	020122	020130	
	050262	043117	051506	052105	
3638	050270	015	012		.BYTE 15,12
3639	050272	042101	052512	052123	.ASCIIZ /ADJUST R23 FOR X GAIN/
	050300	051040	031462	043040	
	050306	051117	054040	043440	
	050314	044501	000116		
3640	050320	040503	042515	040522	ADMSG3: .ASCII /CAMERA 01/
	050326	030040	061		
3641	050331	015	012		.BYTE 15,12
3642	050333	101	045104	051525	.ASCII /ADJUST R14 FOR Y OFFSET/
	050340	020124	030522	020064	
	050346	047506	020122	020131	
	050354	043117	051506	052105	
3643	050362	015	012		.BYTE 15,12
3644	050364	042101	052512	052123	.ASCIIZ /ADJUST R19 FOR Y GAIN/
	050372	051040	034461	043040	
	050400	051117	054440	043440	
	050406	044501	000116		
3645	050412	040503	042515	040522	ADMSG4: .ASCII /CAMERA 02/
	050420	030040	062		
3646	050423	015	012		.BYTE 15,12
3647	050425	101	045104	051525	.ASCII /ADJUST R11 FOR X OFFSET/
	050432	020124	030522	020061	
	050440	047506	020122	020130	

3648	050446	043117	051506	052105	
	050454	015	012		.BYTE 15,12
3649	050456	042101	052512	052123	.ASCIZ /ADJUST R22 FOR X GAIN/
	050464	051040	031062	043040	
	050472	051117	054040	043440	
	050500	044501	000116		
3650	050504	040503	042515	040522	ADMSG5: .ASCII /CAMERA 02/
	050512	030040	062		
3651	050515	015	012		.BYTE 15,12
3652	050517	101	045104	051525	.ASCII /ADJUST R15 FOR Y OFFSET/
	050524	020124	030522	020065	
	050532	047506	020122	020131	
	050540	043117	051506	052105	
3653	050546	015	012		.BYTE 15,12
3654	050550	042101	052512	052123	.ASCIZ /ADJUST R18 FOR Y GAIN/
	050556	051040	034061	043040	
	050564	051117	054440	043440	
	050572	044501	000116		
3655	050576	040503	042515	040522	ADMSG6: .ASCII /CAMERA 03/
	050604	030040	063		
3656	050607	015	012		.BYTE 15,12
3657	050611	101	045104	051525	.ASCII /ADJUST R12 FOR X OFFSET/
	050616	020124	030522	020062	
	050624	047506	020122	020130	
	050632	043117	051506	052105	
3658	050640	015	012		.BYTE 15,12
3659	050642	042101	052512	052123	.ASCIZ /ADJUST R21 FOR X GAIN/
	050650	051040	030462	043040	
	050656	051117	054040	043440	
	050664	044501	000116		
3660	050670	040503	042515	040522	ADMSG7: .ASCII /CAMERA 03/
	050676	030040	063		
3661	050701	015	012		.BYTE 15,12
3662	050703	101	045104	051525	.ASCII /ADJUST R16 FOR Y OFFSET/
	050710	020124	030522	020066	
	050716	047506	020122	020131	
	050724	043117	051506	052105	
3663	050732	015	012		.BYTE 15,12
3664	050734	042101	052512	052123	.ASCIZ /ADJUST R17 FOR Y GAIN/
	050742	051040	033461	043040	
	050750	051117	054440	043440	
	050756	044501	000116		
3665					
3666	050762	015	012		.BYTE 15,12
3667	050764	044124	020105	042523	PRIMO: .ASCII /THE SELF-TEST CONNECTOR MUST BE INSTALLED ON A017/
	050772	043114	052055	051505	
	051000	020124	047503	047116	
	051006	041505	047524	020122	
	051014	052515	052123	041040	
	051022	020105	047111	052123	
	051030	046101	042514	020104	
	051036	047117	040440	030460	
	051044	067			
3668	051045	015	012		.BYTE 15,12
3669	051047	124	042510	051440	.ASCII \THE SWITCH ON A017 MUST BE IN 'MAINT.' POSITION (TOWARD THE I/O CONNECT
	051054	044527	041524	020110	

	051062	047117	040440	030460
	051070	020067	052515	052123
	051076	041040	020105	047111
	051104	021040	040515	047111
	051112	027124	020042	047520
	051120	044523	044524	047117
	051126	024040	047524	040527
	051134	042122	052040	042510
	051142	044440	047457	041440
	051150	047117	042516	052103
	051156	051117	051	
3670	051161	015	012	000
3671	051164	015	012	
3672	051166	044124	020105	042523
	051174	043114	052055	051505
	051202	020124	047503	047116
	051210	041505	047524	020122
	051216	052515	052123	041040
	051224	020105	042522	047515
	051232	042526	020104	051106
	051240	046517	052040	042510
	051246	040440	030460	067
3673	051253	015	012	
3674	051255	124	042510	051440
	051262	044527	041524	020110
	051270	047117	040440	030460
	051276	020067	052515	052123
	051304	041040	020105	047111
	051312	021040	050117	051105
	051320	021056	050040	051517
	051326	052111	047511	020116
	051334	040450	040527	020131
	051342	051106	046517	052040
	051350	042510	044440	047457
	051356	041440	047117	042516
	051364	052103	051117	051
3675	051371	015	012	000
3676	051374	047111	042523	052122
	051402	051040	046501	050055
	051410	045501	041440	041101
	051416	042514	044440	052116
	051424	020117	044124	020105
	051432	047523	045503	052105
	051440	047440	020116	044124
	051446	020105	030101	033461
3677	051454	015	012	
3678	051456	042523	020124	040522
	051464	026515	040520	020113
	051472	053523	052111	044103
	051500	051505	052040	020117
	051506	044442	021116	020054
	051514	047042	051117	021115
	051522	040440	042116	021040
	051530	043117	021106	
3679	051534	015	012	
3680	051536	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/

	051544	041040	040514	045516
	051552	051040	046517	044440
	051560	052116	020117	047522
	051566	020115	046102	051501
	051574	042524	020122	047523
3681	051602	045503	052105	
	051606	015	012	
3682	051610	046120	040505	042523
	051616	042040	050105	042522
	051624	051523	052040	042510
	051632	021040	027511	021117
	051640	040440	042116	021040
	051646	054105	041505	052125
	051654	021105	041040	052125
	051662	047524	051516	052040
	051670	043517	052105	042510
	051676	122		
3683	051677	015	012	
3684	051701	124	042510	032040
	051706	050040	047522	042503
	051714	051523	046040	042105
	051722	020123	044527	046114
	051730	046040	043511	052110
	051736	053440	042510	020116
	051744	047504	042516	041440
	051752	051117	042522	052103
	051760	054514		
3685	051762	015	012	000
3686	051765	015	012	
3687	051767	120	042514	051501
	051774	020105	042522	047515
	052002	042526	051040	046501
	052010	050055	045501	041440
	052016	041101	042514	
3688	052022	015	012	
3689	052024	040440	042116	044440
	052032	051516	051105	020124
	052040	044124	020105	046102
	052046	051501	042524	020104
	052054	047522	020115	047111
	052062	047524	051440	041517
	052070	042513	020124	047117
	052076	040440	030460	020067
	052104	047515	052504	042514
3690	052112	015	012	
3691	052114	042504	051120	051505
	052122	020123	044124	020105
	052130	041442	021122	045440
	052136	054505	053440	042510
	052144	020116	042522	042101
	052152	131		
3692	052153	015	012	000
3693	052156	015	012	
3694	052160	050042	030512	020042
	052166	040503	046102	020105
	052174	052515	052123	041040

.BYTE 15,12
.ASCII \PLEASE DEPRESS THE 'I/O' AND 'EXECUTE' BUTTONS TOGETHER\

.BYTE 15,12
.ASCII \THE 4 PROCESS LEDS WILL LIGHT WHEN DONE CORRECTLY\

PRIM1: .BYTE 15,12,0
.BYTE 15,12
.ASCII /PLEASE REMOVE RAM-PAK CABLE/

.BYTE 15,12
.ASCII / AND INSERT THE BLASTED ROM INTO SOCKET ON A017 MODULE/

.BYTE 15,12
.ASCII /DEPRESS THE 'CR' KEY WHEN READY/

PRIM2: .BYTE 15,12,0
.BYTE 15,12
.ASCII /'PJ1' CABLE MUST BE CONNECTED TO THE A017 CONNECTOR/

3695	052202	020105	047503	047116		
3696	052210	041505	042524	020104		
3697	052216	047524	052040	042510		
	052224	040440	030460	020067		
	052232	047503	047116	041505		
	052240	047524	122			
	052243	015	012	000	PRIM4:	.BYTE 15,12,0
	052246	015	012			.BYTE 15,12
	052250	050042	031512	020042		.ASCII /'PJ3'' CABLE MUST BE CONNECTED TO THE M8036 CONNECTOR/
	052256	040503	046102	020105		
	052264	052515	052123	041040		
	052272	020105	047503	047116		
	052300	041505	042524	020104		
	052306	047524	052040	042510		
	052314	046440	030070	033063		
	052322	041440	047117	042516		
3698	052330	052103	051117			
3699	052334	015	012	000		.BYTE 15,12,0
	052340					.EVEN
3700	052340	015	012	007	FATAL0:	.BYTE 15,12,7 ;'RUNIT'' MUST BE ON A WORD BOUNDARY
3701	052343	106	052101	046101		.ASCII /FATAL BUS TRAP WHEN PERFORMING TEST ''/
	052350	041040	051525	052040		
	052356	040522	020120	044127		
	052364	047105	050040	051105		
	052372	047506	046522	047111		
	052400	020107	042524	052123		
	052406	021040				
3702	052410	021103	040411	020124	RUNIT:	.ASCIZ /C'' AT LOCATION / ;'RUNIT'' MUST BE A WORD BOUNDARY
	052416	047514	040503	044524		
	052424	047117	000040			
3703	052430	005015	051120	043517	RUNITA:	.ASCIZ <15><12>/PROGRAM RESTARTING/ ;'RUNIT'' MUST BE A WORD BOUNDARY
	052436	040522	020115	042522		
	052444	052123	051101	044524		
	052452	043516	000			
3704	052455	015	012	007		.BYTE 15,12,7,0
	052460	000				
3705	052461	015	012		MSGMEM:	.BYTE 15,12
3706	052463	122	047125	044516		.ASCIZ /RUNNING WITH /
	052470	043516	053440	052111		
	052476	020110	000			
3707	052501	040	020113	043117	MSGK:	.ASCIZ / K OF MEMORY/<15><12>
	052506	046440	046505	051117		
	052514	006531	000012			
3708	052520	015	012	007	WARNO:	.BYTE 15,12,7
3709	052523	117	052103	046101		.ASCIZ /OCTAL ADDRESS OF TERMINAL /
	052530	040440	042104	042522		
	052536	051523	047440	020106		
	052544	042524	046522	047111		
	052552	046101	036440	000040		
3710	052560	033	000		ESCP:	.BYTE 33,0
3711	052562	043113	027462	062	KFO:	.ASCII \KF2/2\
3712	052567	015	000			.BYTE 15,0
3713	052571	106	031515	060	FMO:	.ASCII \FM30\
3714	052575	015	000			.BYTE 15,0
3715	052577	104	030111	031457	DIO:	.ASCII \DIO/37\
	052604	067				

3716	052605	015	002			.BYTE	15,2
3717	052607						
3720	052607	060	060	060	DIDATA:	.BYTE	60,60,60,40
(1)	052612	040					
(1)	052613	060	060	060		.BYTE	60,60,60,40
(1)	052616	040					
(1)	052617	060	060	060		.BYTE	60,60,60,40
(1)	052622	040					
(1)	052623	060	060	060		.BYTE	60,60,60,40
(1)	052626	040					
(1)	052627	060	060	060		.BYTE	60,60,60,40
(1)	052632	040					
(1)	052633	060	060	060		.BYTE	60,60,60,40
(1)	052636	040					
(1)	052637	060	060	060		.BYTE	60,60,60,40
(1)	052642	040					
(1)	052643	060	060	060		.BYTE	60,60,60,40
(1)	052646	040					
(1)	052647	060	060	060		.BYTE	60,60,60,40
(1)	052652	040					
(1)	052653	060	060	060		.BYTE	60,60,60,40
(1)	052656	040					
(1)	052657	060	060	060		.BYTE	60,60,60,40
(1)	052662	040					
(1)	052663	060	060	060		.BYTE	60,60,60,40
(1)	052666	040					
(1)	052667	060	060	060		.BYTE	60,60,60,40
(1)	052672	040					
(1)	052673	060	060	060		.BYTE	60,60,60,40
(1)	052676	040					
(1)	052677	060	060	060		.BYTE	60,60,60,40
(1)	052702	040					
(1)	052703	060	060	060		.BYTE	60,60,60,40
(1)	052706	040					
(1)	052707	060	060	060		.BYTE	60,60,60,40
(1)	052712	040					
(1)	052713	060	060	060		.BYTE	60,60,60,40
(1)	052716	040					
(1)	052717	060	060	060		.BYTE	60,60,60,40
(1)	052722	040					
(1)	052723	060	060	060		.BYTE	60,60,60,40
(1)	052726	040					
(1)	052727	060	060	060		.BYTE	60,60,60,40
(1)	052732	040					
(1)	052733	060	060	060		.BYTE	60,60,60,40
(1)	052736	040					
(1)	052737	060	060	060		.BYTE	60,60,60,40
(1)	052742	040					
(1)	052743	060	060	060		.BYTE	60,60,60,40
(1)	052746	040					
(1)	052747	060	060	060		.BYTE	60,60,60,40
(1)	052752	040					
(1)	052753	060	060	060		.BYTE	60,60,60,40
(1)	052756	040					
(1)	052757	060	060	060		.BYTE	60,60,60,40
(1)	052762	040					


```

(1) 052763 060 060 060 .BYTE 60,60,60,40
(1) 052766 040
(1) 052767 060 060 060 .BYTE 60,60,60,40
(1) 052772 040
(1) 052773 060 060 060 .BYTE 60,60,60,40
(1) 052776 040
(1) 052777 060 060 060 .BYTE 60,60,60,40
(1) 053002 040
(1) 053003 060 060 060 .BYTE 60,60,60,40
(1) 053006 040
3721 053007 003 000 .BYTE 3,0
3722 053011 103 030115 031457 CM0: .ASCII \CM0/37\
      053016 067
3723 053017 015 000 .BYTE 15,0
3724 053021 103 000103 CC0: .ASCII /CC/
3725 053024 043520 027460 033463 PG0: .ASCII \PG0/37\
3726 053032 015 000 .BYTE 15,0
3727 053034 050132 ZP0: .ASCII /ZP/
3728 053036 015 000 .BYTE 15,0
3729 .EVEN
3730
3731 ;BUFFER AREA
3732
3733 . =60000
3734 060000 001000 BUF0: .BLKW 512.
3735 062000 001000 BUF1: .BLKW 512.
3736 064000 000240 BUF2: NOP
3737
3738 000001 .END

```


AMTYP4=	000000	32												
APASS =	000000	32												
APRIOR=	000000	32												
APTC SU=	000040	2341	2349#											
APTE NV=	000001	2341	2345	2349#										
APTSIZ=	000200	147	2349#											
APTSP O=	000100	2341	2349#											
ASWREG=	000000	32												
A TESTN=	000000	32												
AUNIT =	000000	32												
AUSWR =	000000	32												
AVECT1-	140370	17#	32	111	112	113	114							
AVECT2=	000000	32												
AVGVAL	041714	2639#	2923	2924	2934	2999	3087*	3089	3095	3162				
AVRGO	041414	2614#	3161											
BAR	001754	101#	356	490*	491	517*	532	596*	601	617*	622	641*	659*	672*
		676	684*	715*	749*	795*	1034*	1052	1084*	1104	1129*	1159*	1188*	1222*
		1243*	1265*	1286*	1310*	1331*	1353*	1432*	1474*	1510*	1548*	1768*	1804*	1805*
		1811	1828*	1850*	1856	1870*	1876	1890*	1896	1924*	1991*	2056*	2156*	2498*
		3050*	3286*	3562*										
BAR1	001764	105#	359											
BEGIN	002034	22	145#	225	228									
BELMSG	041572	2624#	3223											
BELOW	041720	2641#	3136*	3204	3221									
BITO	000001	13#	131	473	486	499	566	582	598	619	643	661	686	717
		751	798	956	957	958	960	961	962	964	965	966	968	969
		970	972	973	974	976	977	978	980	981	982	989	1003	1017
		1036	1086	1132	1161	1198	1226	1247	1268	1289	1313	1334	1356	1376
		1410	1436	1479	1515	1554	1612	1613	1615	1616	1618	1619	1621	1622
		1624	1625	1627	1628	1641	1673	1674	1689	1715	1731	1753	1772	1807
		1808	1830	1831	1852	1853	1872	1873	1892	1893	1928	1993	2058	2113
		2117	2119	2158	2164	2166	2196	2206	2208	2224	2254	2264	2266	2420
		2434	2503	2732	2809	2816	2822	3053	3054	3292	3565			
BIT00 -	000001	13#												
BIT01 -	000002	13#												
BIT02 =	000004	13#												
BIT03 =	000010	13#												
BIT04 =	000020	13#												
BIT05 -	000040	13#												
BIT06 -	000100	13#												
BIT07 -	000200	13#												
BIT08 =	000400	13#	2340											
BIT09 -	001000	13#	2340	2345										
BIT1	000002	13#	130	563	565	574	597	618	642	660	685	716	738	750
		797	801	1374	1434	1448	1458	1770	1783	1806	1829	1851	1871	1891
		2111	2447	3017	3024	3031	2345	2382	2393	2404	2415			
BIT10 -	002000	13#	1272	1360	1770	1783								
BIT11 =	004000	13#	126	383	2340									
BIT12 -	010000	13#	1041	1089	1137	1166	1203	1228	1252	1273	1294	1317	1340	1361
		1381	1414	1443	1485	1525	1559	1612	1613	1615	1616	1618	1619	1621
		1622	1624	1625	1627	1628	1645	1676	1694	1718	1734	1756	1779	2116
		2162	2200	2227	2258									
BIT13 -	020000	13#	1421	1425	1448	1456	1490	1533	1783	1910	1932	1939	1972	1979
		1998	2005	2039	2061	2068	2345							
BIT14 -	040000	13#	125	610	633	652	691	693	723	725	756	758	801	1141
		1170	1179	1932	1939	1998	2005	2061	2068	2340				

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	2460					
BIT3 = 000010	13#	128	2473										
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	2111
	2222	2480	3052										
BIT5 = 000040	13#	1513	1533										
BIT6 = 000100	13#	774	1198	2428									
BIT7 = 000200	13#	311	388	400	409	574	581	705	738	801	826	845	850
	1141	1150	1170	1179	1497	1533	2441						
BIT8 = 000400	13#	127	621	663	1316	1851	1891	2388	2393	2410	2415	2454	3471
BIT9 = 001000	13#	1293	1360	1871	1891	2399	2404	2410	2415	2467	3442	3445	
BLAST 042272	205	2730#											
BLICNT 042130	2672*	2676	2679*	2683*	2703#								
BPTVEC= 000014	13#												
BRLEV 002002	118#	167*	1542	1566									
BTALK 042020	233	2681#											
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	1969	1978	1982	2008	2036	2043	2047	2056
	2071	2090	2137	2185	2226*	2228	2243	2498	2522	2681	3003	3048	3062
	3286	3296	3555	3562	3574	3580	3586	3734#					
BUF 1 062000	1081	1105	2917	3063	3074	3080	3097	3556	3735#				
BUF 2 064000	3005	3071	3736#										
BUSTRP 003212	152	258#											
CALRPT 037614	2419	2427#											
CAMUNP 040360	2431	2437	2444	2450	2457	2463	2470	2476	2483	2548#			
CAMVAL 050026	3532*	3563	3575	3619#									
CAMOG1 040442	2378	2432	2565#										
CAMOG2 040444	2383	2438	2566#										
CAMOTW 040503	2436	2577#											
CAMOTX 040464	2430	2575#											
CAM1G1 040446	2389	2445	2567#										
CAM1G2 040450	2394	2451	2568#										
CAM1TW 040541	2449	2581#											
CAM1TX 040522	2443	2579#											
CAM2G1 040452	2400	2458	2569#										
CAM2G2 040454	2405	2464	2570#										
CAM2TW 040577	2462	2585#											
CAM2TX 040560	2456	2583#											
CAM3G1 040456	2411	2471	2571#										
CAM3G2 040460	2416	2477	2572#										
CAM3TW 040635	2475	2589#											
CAM3TX 040616	2469	2587#											
CCO 053021	2780	2813	2834	3724#									
CKSWR 104407	2340	2345	2365#										
CKTSWR 040062	2427	2433	2440	2446	2453	2459	2466	2472	2479	2487#			
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

RESREG=	104414	2362	2365#											
RESVEC=	000010	13#												
ROMPNT	041710	2635#	2749*	2756*	2918									
ROMVAL	042132	2635	2708#	2740	2742	2749	2756	2795	2804	3255	3257			
RTN	041566	2623#	3164	3259	3260	3261	3265							
RTRT	002014	23	140#	265	3341									
RTRTA	002044	141	144	147#	279									
RUNDIF	050022	184*	208*	212*	221*	248*	2274	2992	3617#					
RUNIT	052410	151*	195*	196*	197*	198	203	206	210	216	219	223	226	231
		236	239	242	246	3153	3169	3702#						
RUNITA	052430	264	3703#											
SAMCAM	047212	3474#												
SAMDAT	047512	3535	3539	3543	3555#									
SAVREG=	104413	2362	2365#											
SDPAR0=	172260	14#												
SDPAR1=	172262	14#												
SDPAR2=	172264	14#												
SDPAR3=	172266	14#												
SDPAR4=	172270	14#												
SDPAR5=	172272	14#												
SDPAR6=	172274	14#												
SDPAR7=	172276	14#												
SDPDR0=	172220	14#												
SDPDR1=	172222	14#												
SDPDR2=	172224	14#												
SDPDR3=	172226	14#												
SDPDR4=	172230	14#												
SDPDR5=	172232	14#												
SDPDR6=	172234	14#												
SDPDR7=	172236	14#												
SFR	001756	102#	357	408*	435*	436	446*	449	456*	459	464*	465*	467	476*
		489*	502*	514*	518*	537	546*	550*	552*	558*	559*	570*	573*	583*
		593*	594*	599*	614*	615*	620*	638*	639*	644*	656*	657*	662*	670*
		674*	681*	682*	688*	699*	704*	712*	713*	719*	731*	732*	737*	746*
		747*	753*	764*	765*	780*	786*	792*	793*	799*	818*	819*	821	825
		831	837*	83F*	840	844	848*	849	856	858	875*	880*	885*	890*
		895*	900*	90L*	913*	919*	925*	931*	936*	956*	957*	958*	960*	961*
		962*	964*	965*	966*	968*	969*	970*	972*	973*	974*	976*	977*	978*
		980*	981*	982*	985*	986*	988*	990*	999*	1000*	1002*	1004*	1013*	1014*
		1016*	1018*	1030*	1035*	1037*	1041*	1046*	1078*	1085*	1088*	1089*	1103*	1126*
		1130*	1133*	1137*	1149*	1156*	1160*	1162*	1166*	1178*	1185*	1189*	1199*	1203*
		1213*	1218*	1224*	1227*	1228*	1239*	1245*	1248*	1252*	1263*	1266*	1272*	1273*
		1284*	1287*	1293*	1294*	1306*	1311*	1317*	1327*	1332*	1340*	1351*	1354*	1360*
		1361*	1371*	1375*	1377*	1381*	1406*	1409*	1413*	1414*	1425*	1430*	1435*	1437*
		1443*	1456*	1473*	1478*	1480*	1485*	1495*	1509*	1514*	1516*	1525*	1537*	1545*
		1552*	1555*	1559*	1587*	1612*	1613*	1615*	1616*	1618*	1619*	1621*	1622*	1624*
		1625*	1627*	1628*	1637*	1639*	1642*	1645*	1668*	1670*	1672*	1674*	1676*	1685*
		1687*	1690*	1691*	1694*	1711*	1713*	1716*	1718*	1727*	1729*	1732*	1734*	1747*
		1751*	1754*	1756*	1766*	1771*	1773*	1779*	1801*	1802*	1824*	1825*	1847*	1848*
		1867*	1868*	1887*	1888*	1921*	1925*	1927*	1989*	1992*	1994*	2054*	2057*	2059*
		2109*	2112*	2114*	2116*	2153*	2157*	2159*	2162*	2192*	2195*	2197*	2200*	2218*
		2222*	2225*	2227*	2250*	2252*	2255*	2258*	2420*	2421	2496*	2500*	3047*	3052*
		3057*	3283*	3291*	3335*	3339*	3564*	3570*						
SFRHB	001770	107#	448*											
SGNVAL	041546	2620#	3181											
SGNVL 1	041554	2621#	3102*	3104*	3134	3140	3146							

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	025666	1941	1951	1963	1965	1968	1973#	
TST153	026246	2007	2017	2024	2030	2032	2035	2041#
TST154	026470	2070	2078	2083#				
TST155	026764	2085	2124	2126	2131#			
TST156	027262	2133	2171	2173	2178#			
TST157	027476	2183	2210	2213#				
TST16	005314	548#						
TST160	027636	2220	2232	2236#				
TST161	030056	2241	2269	2273#				
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	035702	2349#*																	
\$MNEW	033551	2335#																	
\$MSGAD	001210	32#	2349*																
\$MSGLG	001212	32#	2349*																
\$MSGTY	001174	32#	2349*																
\$MSWR	033540	2335#																	
\$MTYP1	001225	32#																	
\$MTYP2	001231	32#																	
\$MTYP3	001235	32#																	
\$MTYP4	001241	32#																	
\$MULT	036506	2358#	2928	3107															
\$MXCNT	034372	2340#																	
\$NULL	001154	32#	2341																
\$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#	1973#	2041#	2083#	2131#	2178#	2213#	2236#	2273#									
\$SOCNT	035100	2343#*																	
\$SOMODE	035102	2343#*																	
\$OVER	034356	2340#																	
\$PASS	001202	32#	147*	299	1962	2029	2123	2170	2277*	2340									
\$PASTM	001006	31#																	
\$POWER	036054	2350#																	
\$PWDRN	035706	147	2350#																
\$PWARMG	036042	2350#																	
\$PWRUP	035760	2350#																	
\$QUES	001170	32#	253	2335	2341	2345													
\$RDCHR	033270	2335#	2365																
\$RDDEC=	***** U	2365																	
\$RDLIN	033410	2335#	2365																
\$RDOCT	033562	2337#	2365																
\$RDSZ =	000010	2335#																	
\$RESRE	036450	2356#	2365																
\$RTNAD	030200	2277#																	
\$R2A =	***** U	2365																	
\$SAVRE	036412	2356#	2365																
\$SAVR6	036052	2350#*																	
\$SB2D	037234	2363#	3175																
\$SCOPE	034110	147	2340#																
\$SETUP=	000137	139#	147	149	2277	2335	2340	2345											
\$SIZE	036064	305	312	2352#															
\$SIZEX	036342	2352#																	
\$STUP -	177777	139#																	
\$SVLAD	034322	2340#																	
\$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					

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 105
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0180

COMMEN	13#														
ENDCOM	13#														
ERROR	13#	368	371	391	404	413	423	430	440	452	461	469	481	494	507
	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	2006
	2016	2023	2069	2077	2122	2169	2211	2233	2270						
ESCAPE	13#														
GETPRI	13#	2352	2360												
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	1973	2041	2083	2131	2178	2213	2236	2273						
OFFBIT	1591#	1612	1613	1615	1616	1618	1619	1621	1622	1624	1625	1627	1628		
POP	13#	2337	2339	2349	2350	2356	2358	2360							
PUSH	13#	2337	2339	2349	2350	2356	2358	2360							
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	1973	2041	2083	2131	2178	2213	2236	2273						
SETPRI	13#														
SETTRA	2365#														
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	1994	2114	2159	2197
	2225	2255													
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

CZNCCB NCV11 DIAGNOSTIC
CZNCCB.P11 31-AUG-79 11:21

MACY11 30G(1063) 31-AUG-79 13:10 PAGE 105-2
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0182

.\$APT	11#	32#	
.\$APTH	11#	31	
.\$APTY	11#	2349	
.\$CATC	7#	22	
.\$CKSW	9#		
.\$CMTA	7#	32	
.\$DB2D	10#	2362	
.\$DIV	10#	2360	
.\$EOP	7#	2277	
.\$ERRO	7#	2345	
.\$ERRT	9#	2347	
.\$MUL	10#	2358	
.\$PARM	8#		
.\$POWE	8#	2350	
.\$RDOC	10#	2337	
.\$READ	8#	2335	
.\$SAVE	8#	10#	2356
.\$SB2D	10#	2363	
.\$SCOP	8#	2340	
.\$SIZE	10#	2352	
.\$SPAC	8#		
.\$SWDO	8#		
.\$TRAP	8#	2365	
.\$TYPB	9#		
.\$TYPD	9#	2339	
.\$TYPE	7#	8#	2341
.\$TYPO	7#	2343	

. ABS. 064002 000 CON RW ABS LCL I

ERRORS DETECTED: 0

CZNCCB,CZNCCB/CRF CZNCCB
RUN-TIME: 42 31 3 SECONDS
RUN-TIME RATIO: 120/76=1.5
CORE USED: 30K (59 PAGES)