

NCV-11

DIAGNOSTIC
CZNCCA0

AH-E772A-MC
COPYRIGHT © 73-78
FICHE 1 OF 1

DEC 1978
digital
MADE IN USA

This microfiche card contains a grid of frames, each containing diagnostic data. The data is organized into columns and rows, with some frames containing text and others containing graphical representations like bar charts or waveforms. The text is too small to read clearly, but it appears to be technical information related to the diagnostic code CZNCCA0.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-E771A-MC
PRODUCT NAME: CZNCCA0 NCV11 DIAGNOSTIC TEST
DATE: AUGUST 1978
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1973,1978
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

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

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 OPTION INCLUDING KVV11 REAL TIME CLOCK (M7952)
4. A017 SELF-TEST CONNECTOR (70-12894) (REQUIRED FOR SECTION 9.2 + 9.3)
5. H3060 JOYSTICK (OPTIONAL)
6. HARDWARE TESTER AND PROM BLASTER (OPTIONAL)

2.2 STORAGE

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

2.3 PRELIMINARY PROGRAMS

THE KVV11 DIAGNOSTIC (MD-11-DVKWA) 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.
- LSI11: 6. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.
7. THE PROGRAM WILL REQUEST SWITCH REGISTER VALUE.
8. THE PROGRAM WILL TYPE THE TEST SELECTION MENU FOR THE OPERATOR.
9. THE OPERATOR SELECT'S THE APPROPRIATE TEST BY TYPING THE TEST LETTER AND 'CR'

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

4.1 PROGRAM START

200
204
210

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

5.0 SOFTWARE SWITCH REGISTER

5.1 LOGIC TEST OPTIONS

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

5.2 FIELD ADJUSTMENT LOOP

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

5.3 CONTROL

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

6.0 ERROR REPORTING

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 M8C26 '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																
RES0	RES1	WORD*16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
1	1	1	B	G	Z	G	Z	G	1	1	1	G	Z	G	Z	G	1	1	1
1	1	0	0	B	G	Z	G	Z	G	1	1	G	Z	G	Z	G	1	1	0
1	0	1	0	0	B	G	Z	G	Z	G	1	0	G	Z	G	Z	G	1	0
1	0	0	0	0	0	B	G	Z	G	Z	G	1	G	Z	G	Z	G	1	0
0	1	1	0	0	0	0	B	G	Z	G	Z	G	0	G	Z	G	Z	G	0
0	1	0	0	0	0	0	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 SET TO 'OPER.' POSITION>

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

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

AUXILIARY SECTION

9.5 O OTHER TERMINAL ADDRESS

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

9.6 H HELP THE OPERATOR

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

9.7 S SOFTWARE SWITCH REGISTER

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

FACTORY OPTION CHECK-OUT SECTION

9.8 I INITIAL ADJUSTMENT OF A017

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

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

9.9

B BLASTING THE LINEARITY CORRECTION PROM

J 1

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	116	01001110
21	144	01100100
22	216	10001110
23	244	10100100

10.0

LISTING

17	BASIC DEFINITIONS
127	MEMORY MANAGEMENT DEFINITIONS
277	OPERATIONAL SWITCH SETTINGS
288	TRAP CATCHER
297	STARTING ADDRESS(ES)
304	ACT11 HOOKS
315	APT PARAMETER BLOCK
337	COMMON TAGS
381	APT MAILBOX-ETABLE
430	ERROR POINTER TABLE
586	INITIALIZE THE COMMON TAGS
629	TYPE PROGRAM NAME
636	GET VALUE FOR SOFTWARE SWITCH REGISTER
650	PRIME THE NCV11 ADDRESSES FROM THE DEFAULT VALUES
678	DIAGNOSTIC IDENTIFICATION TYPEOUT
691	KEYBOARD COMMAND DECODER
840	
841	TEST
842	-----
843	
844	T1 VERIFY THE 8 NCV11 AND 2 NCV11 CLOCK BUS ADDRESSES RESPOND
878	T2 FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.
899	T3 VERIFY THAT "INIT" CLEARS THE CSR REGISTER
913	T4 VERIFY THAT "CLEAR ALL" CLEARS THE CSR REGISTER
927	T5 VERIFY LOW BYTE OPERATION OF THE "CSR" REGISTER
947	T6 FLOAT A 1 ACROSS 4 BITS OF THE SPECIAL FUNCTION REGISTER
965	T7 VERIFY THAT CLEARING HIGH BYTE OF SFR DOES NOT CLEAR LOW BYTE
979	T10 VERIFY THAT "INIT" CLEARS THE SFR REGISTER
992	T11 VERIFY THAT "CLEAR ALL" CLEARS THE SFR REGISTER
1005	T12 FLOAT A 1 ACROSS THE WORD COUNT REGISTER
1023	T13 FLOAT A 1 ACROSS THE BUS ADDRESS REGISTER
1041	T14 FLOAT A 1 ACROSS THE OFFSET REGISTER
1059	T15 VERIFY NO DUAL REGISTER SELECTION
1099	T16 VERIFY "CLR ALL" CLEARS THE EXTENDED OFFSET BITS
1113	T17 TEST THE "ACTIVE" FLOP CAN SET AND CLEAR
1148	
1149	T20 VERIFY Z INPUTS CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN MATRIX MODE
1176	T21 VERIFY Z INPUTS CAUSE THE LOW 16 BITS OF 32 BIT COUNTER TO CHANGE
1205	T22 VERIFY Z INPUTS CAUSE THE LOW 24 BITS OF 32 BIT COUNTER TO CHANGE
1229	T23 VERIFY Z INPUTS CAUSE THE HIGH 8 BITS OF THE 32 BIT COUNTER TO CHANGE
1248	T24 VERIFY Z INPUTS DO NOT CAUSE THE LOW 8 BITS OF 32 BIT COUNTER TO CHANGE IN LIST MODE
1265	T25 TEST THAT "WCA OVFL" SETS Z/WC OVERFLOW FLOP AND "CLR ALL" CLEARS IT
1301	T26 TEST THAT "WCA OVFL" SETS Z/WC OVERFLOW FLOP AND "CLR WC OVFL" CLEARS IT
1339	T27 TEST THAT "WCA OVFL" GENERATES AN INTERRUPT
1390	T30 VERIFY "WCA OVFL" CLEARS "ACTIVE"
1417	
1418	T31 VERIFY JOYSTICK DONE FLOP SETS
1442	T32 VERIFY THAT "RESET" INSTRUCTION CLEARS THE JOY READY FLOP
1473	T33 JOYSTICK DATA PATH = GAIN =0 ZB ENABLE =0 RES. = 000
1491	T34 JOYSTICK DATA PATH = GAIN =1
1509	T35 JOYSTICK DATA PATH = ZB ENABLE =1
1526	T36 JOYSTICK DATA PATH RES. = 001
1544	T37 JOYSTICK DATA PATH RES. = 010
1562	T40 JOYSTICK DATA PATH RES. = 100

1579		
1580	T41	VERIFY THE DATA INCREMENT FUNCTION
1598	T42	VERIFY THE DATA INCREMENT CARRY BIT
1617	T43	VERIFY THE DATA INCREMENT FUNCTION IS INHIBITED
1635	T44	VERIFY THE DATA DECREMENT FUNCTION
1654	T45	VERIFY THE DATA DECREMENT BORROW
1672	T46	VERIFY THE DATA DECREMENT FUNCTION IS INHIBITED
1690	T47	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 0
1708	T50	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 1 ZB ENABLE = 0
1726	T51	TEST ADDRESS MAKER - MATRIX MODE - RES = 7 GAIN = 0 ZB ENABLE = 1
1744	T52	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 0
1762	T53	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 1 ZB ENABLE = 0
1780	T54	TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 1
1798	T55	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 0
1816	T56	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 1 ZB ENABLE = 0
1834	T57	TEST ADDRESS MAKER - MATRIX MODE - RES = 5 GAIN = 0 ZB ENABLE = 1
1852	T60	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 0
1870	T61	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 1 ZB ENABLE = 0
1888	T62	TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 1
1906	T63	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 0
1924	T64	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 1 ZB ENABLE = 0
1942	T65	TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 1
1960	T66	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 0
1978	T67	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 1 ZB ENABLE = 0
1996	T70	TEST ADDRESS MAKER - MATRIX MODE - RES = 2 GAIN = 0 ZB ENABLE = 1
2014	T71	TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 0
2032	T72	TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 1 ZB ENABLE = 0
2050	T73	TEST ADDRESS MAKER - MATRIX MODE - RES = 1 GAIN = 0 ZB ENABLE = 1
2068	T74	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 0
2087	T75	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 1 GAIN = 0
2106	T76	TEST ADDRESS MAKER - LIST MODE - ZB ENABLE = 0 GAIN = 1
2124		
2125	T77	ENABLE A ONE WORD TRANSFER SECTION LIST MODE
2178	T100	ENABLE A 512 WORD TRANSFER SECTION LIST MODE
2232	T101	VERIFY "TIMEOUT" FLOP SETS AND "CLR ALL" CLEARS IT
2268	T102	VERIFY "TIMEOUT" FLOP SETS AND "CLR TIMEOUT" CLEARS IT
2303	T103	VERIFY "TIMEOUT" INTERRUPT
2341	T104	VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 16
2368	T105	VERIFY INCREMENTING INTO EXTENDED ADDRESS BIT 17
2397	T106	VERIFY "SET EVENT" DATA GENERATES A 177777 DATA WORD
2423	T107	VERIFY "SET TIME" DATA GENERATES A 000000 DATA WORD
2449	T110	VERIFY "CLOCK ST1" GENERATES A EVENT (177777) DATA WORD
2475	T111	VERIFY "CLOCK OVERFLOW" GENERATES A TIME (000000) DATA WORD
2503	T112	VERIFY "SET TIME" OVERRIDES "SET EVENT" DATA
2528	T113	DO A ONE WORD MATRIX MODE TRANSFER -CHECK FOR INCREMENT FUNCTION
2556	T114	VERIFY EACH BIT OF THE INCREMENT REG. DATA PATH
2597	T115	CHECK FOR LOW BYTE "INC OVFL" TO SET CELL OVERFLOW AND "CLR CELL" TO CLEAR IT
2645	T116	CHECK FOR WORD "INC OVFL" TO SET CELL OVERFLOW AND "CLR ALL" TO CLEAR IT
2686	T117	CHECK FOR "CELL OVERFLOW" INTERRUPT
2724	T120	VERIFY CORRECT BR LEVEL THE NCV-11
2779		
2780	T121	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 13
2802	T122	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 12
2824	T123	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 11

2846	T124	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 10
2868	T125	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 09
2890	T126	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 08
2912	T127	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 07
2934	T130	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 06
2956	T131	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 05
2978	T132	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 04
3000	T133	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 03
3022	T134	VERIFY A 1 WORD MATRIX MODE XFR USING OFFSET - 02
3044		
3045	T135	VERIFY THE ADM INPUT TO THE MATRIX MUX. USING GAIN ENABLE
3081	T136	VERIFY THE ADM INPUT TO THE MATRIX MUX. ADDER USING ZB ENABLE
3106	T137	VERIFY LOW BYTE OPERATION OF THE "TESTX" FLOP
3137	T140	VERIFY BIT 12 ADDER INPUT TO MATRIX MODE MUX
3159	T141	VERIFY BIT 13 ADDER INPUT TO MATRIX MODE MUX
3181	T142	VERIFY BIT 14 ADDER INPUT TO MATRIX MODE MUX
3208	T143	CHECK FOR HIGH BYTE "INC OVFL" TO SET CELL OVERFLOW
3243	T144	VERIFY EACH BIT OF THE LOWER 16 BIT Z COUNTER (TESTER ONLY)
3271	T145	VERIFY EACH BIT OF THE HIGHER 16 BIT Z COUNTER (TESTER ONLY)
3299	T146	VERIFY THAT CAMERA 01 CHANNEL IS OPERATIONAL (TESTER ONLY)
3324	T147	VERIFY THAT CAMERA 10 CHANNEL IS OPERATIONAL (TESTER ONLY)
3348	T150	VERIFY THAT CAMERA 11 CHANNEL IS OPERATIONAL (TESTER ONLY)
3372	T151	DYNAMIC MATRIX MODE ADDRESS
3448	T152	DYNAMIC LIST MODE ADDRESS
3521	T153	DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K
3567	T154	ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY
3621	T155	ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY
3674	T156	VERIFY BIT 15 MATRIX ADDER INPUT
3714	T157	VERIFY HIGH BYTE OPERATION OF THE "TEST X"
3742	T160	VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER
3783	T161	DETERMINE IF DIFLIN IS TO BE RUN (F)
3792		END OF PASS ROUTINE
3831		ERROR ASCII MESSAGES
4094		TTY INPUT ROUTINE
4233		READ AN OCTAL NUMBER FROM THE TTY
4271		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4338		SCOPE HANDLER ROUTINE
4404		TYPE ROUTINE
4483		BINARY TO OCTAL (ASCII) AND TYPE
4561		ERROR HANDLER ROUTINE
4616		ERROR MESSAGE TYPEOUT ROUTINE
4663		APT COMMUNICATIONS ROUTINE
4720		POWER DOWN AND UP ROUTINES
4763		ROUTINE TO SIZE MEMORY
4844		SAVE AND RESTORE R0-R5 ROUTINES
4889		INTEGER MULTIPLY ROUTINE
4936		INTEGER DIVIDE ROUTINE
5018		DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5080		SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
5098		TRAP DECODER
5121		TRAP TABLE
5144		A TO D FIELD SITE AND ADJUSTMENT LOOP

```
1      .LIST ME
2      .NLIST MD,MC,CND
3      .ENABL ABS,AMA
4      167400 $SWR=167400
5      000001 $TN=1
6      .MCALL .SCATCH,.SCMTAG,.STYPE,.SEOP,.EQUAT,.HEADER,.$ERROR,.STYPOCT
7      .MCALL .SPARM,.SPOWER,.SREAD,.SSAVE,.SCOPE,.SETUP,.SPACE,.SWDOC,.STYPE,.STRA
8      .MCALL .SWRHI,.SERRTYP,.STYPDS,.STYPBIN,SOFTSWR,.SCKSWR
9      .MCALL .SRDOCT,.SSIZE,.KT11,.SDIV,.SMULT,.SSAVE,.SB2D,.SDB2D
10     .MCALL .SACT11,.SAPTHDR,.SAPTBL5,.SAPTYPE
11     .TITLE CZNCCA NCV11 DIAGNOSTIC
12     ;*COPYRIGHT (C) 1978
13     ;*DIGITAL EQUIPMENT CORP.
14     ;*MAYNARD, MASS. 01754
15     ;*
16     ;*PROGRAM BY RAYMOND SHOOP
17     ;*
18     ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
19     ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
20     ;*
21     .SBTTL BASIC DEFINITIONS
22
23     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
24     001100 STACK= 1100
25     .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
26     .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
27
28     ;*MISCELLANEOUS DEFINITIONS
29     000011 HT= 11        ;;CODE FOR HORIZONTAL TAB
30     000012 LF= 12        ;;CODE FOR LINE FEED
31     000015 CR= 15        ;;CODE FOR CARRIAGE RETURN
32     000200 CRLF= 200     ;;CODE FOR CARRIAGE RETURN-LINE FEED
33     177776 PS= 177776   ;;PROCESSOR STATUS WORD
34     .EQUIV PS,PSW
35     177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
36     177772 PIRQ= 177772  ;;PROGRAM INTERRUPT REQUEST REGISTER
37     177570 DSWR= 177570  ;;HARDWARE SWITCH REGISTER
38     177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
39
40     ;*GENERAL PURPOSE REGISTER DEFINITIONS
41     000000 R0= %0        ;;GENERAL REGISTER
42     000001 R1= %1        ;;GENERAL REGISTER
43     000002 R2= %2        ;;GENERAL REGISTER
44     000003 R3= %3        ;;GENERAL REGISTER
45     000004 R4= %4        ;;GENERAL REGISTER
46     000005 R5= %5        ;;GENERAL REGISTER
47     000006 R6= %6        ;;GENERAL REGISTER
48     000007 R7= %7        ;;GENERAL REGISTER
49     000006 SP= %6        ;;STACK POINTER
50     000007 PC= %7        ;;PROGRAM COUNTER
51
52     ;*PRIORITY LEVEL DEFINITIONS
53     000000 PRO= 0        ;;PRIORITY LEVEL 0
54     000040 PR1= 40      ;;PRIORITY LEVEL 1
```

55	000100	PR2=	100	::PRIORITY LEVEL 2
56	000140	PR3=	140	::PRIORITY LEVEL 3
57	000200	PR4=	200	::PRIORITY LEVEL 4
58	000240	PR5=	240	::PRIORITY LEVEL 5
59	000300	PR6=	300	::PRIORITY LEVEL 6
60	000340	PR7=	340	::PRIORITY LEVEL 7

;'SWITCH REGISTER' SWITCH DEFINITIONS

63	100000	SW15=	100000
64	040000	SW14=	40000
65	020000	SW13=	20000
66	010000	SW12=	10000
67	004000	SW11=	4000
68	002000	SW10=	2000
69	001000	SW09=	1000
70	000400	SW08=	400
71	000200	SW07=	200
72	000100	SW06=	100
73	000040	SW05=	40
74	000020	SW04=	20
75	000010	SW03=	10
76	000004	SW02=	4
77	000002	SW01=	2
78	000001	SW00=	1
79		.EQUIV	SW09,SW9
80		.EQUIV	SW08,SW8
81		.EQUIV	SW07,SW7
82		.EQUIV	SW06,SW6
83		.EQUIV	SW05,SW5
84		.EQUIV	SW04,SW4
85		.EQUIV	SW03,SW3
86		.EQUIV	SW02,SW2
87		.EQUIV	SW01,SW1
88		.EQUIV	SW00,SW0

;'DATA BIT DEFINITIONS (BIT00 TO BIT15)

91	100000	BIT15=	100000
92	040000	BIT14=	40000
93	020000	BIT13=	20000
94	010000	BIT12=	10000
95	004000	BIT11=	4000
96	002000	BIT10=	2000
97	001000	BIT09=	1000
98	000400	BIT08=	400
99	000200	BIT07=	200
100	000100	BIT06=	100
101	000040	BIT05=	40
102	000020	BIT04=	20
103	000010	BIT03=	10
104	000004	BIT02=	4
105	000002	BIT01=	2
106	000001	BIT00=	1
107		.EQUIV	BIT09,BIT9
108		.EQUIV	BIT08,BIT8

```
109 .EQUIV BIT07,BIT7
110 .EQUIV BIT06,BIT6
111 .EQUIV BIT05,BIT5
112 .EQUIV BIT04,BIT4
113 .EQUIV BIT03,BIT3
114 .EQUIV BIT02,BIT2
115 .EQUIV BIT01,BIT1
116 .EQUIV BIT00,BIT0
117
118 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
119 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
120 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
121 000014 TBITVEC=14 ;: 'T' BIT
122 000014 TRTVEC= 14 ;:TRACE TRAP
123 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
124 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
125 000024 PWRVEC= 24 ;:POWER FAIL
126 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
127 000034 TRAPVEC=34 ;: 'TRAP' TRAP
128 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
129 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
130 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
131 .SBTTL MEMORY MANAGEMENT DEFINITIONS
132
133 ;*KT11 VECTOR ADDRESS
134
135 000250 MMVEC= 250
136
137 ;*KT11 STATUS REGISTER ADDRESSES
138
139 177572 SR0= 177572
140 177574 SR1= 177574
141 177576 SR2= 177576
142 172516 SR3= 172516
143
144 ;*USER 'I' PAGE DESCRIPTOR REGISTERS
145
146 177600 UIPDR0= 177600
147 177602 UIPDR1= 177602
148 177604 UIPDR2= 177604
149 177606 UIPDR3= 177606
150 177610 UIPDR4= 177610
151 177612 UIPDR5= 177612
152 177614 UIPDR6= 177614
153 177616 UIPDR7= 177616
154
155 ;*USER 'D' PAGE DESCRIPTOR REGISTORS
156
157 177620 UDPDR0= 177620
158 177622 UDPDR1= 177622
159 177624 UDPDR2= 177624
160 177626 UDPDR3= 177626
161 177630 UDPDR4= 177630
162 177632 UDPDR5= 177632
```


163	177634	UDPDR6= 177634
164	177636	UDPDR7= 177636
165		
166		;*USER 'I' PAGE ADDRESS REGISTERS
167		
168	177640	UIPAR0= 177640
169	177642	UIPAR1= 177642
170	177644	UIPAR2= 177644
171	177646	UIPAR3= 177646
172	177650	UIPAR4= 177650
173	177652	UIPAR5= 177652
174	177654	UIPAR6= 177654
175	177656	UIPAR7= 177656
176		
177		;*USER 'D' PAGE ADDRESS REGISTERS
178		
179	177660	UDPAR0= 177660
180	177662	UDPAR1= 177662
181	177664	UDPAR2= 177664
182	177666	UDPAR3= 177666
183	177670	UDPAR4= 177670
184	177672	UDPAR5= 177672
185	177674	UDPAR6= 177674
186	177676	UDPAR7= 177676
187		
188		;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
189		
190	172200	SIPDR0= 172200
191	172202	SIPDR1= 172202
192	172204	SIPDR2= 172204
193	172206	SIPDR3= 172206
194	172210	SIPDR4= 172210
195	172212	SIPDR5= 172212
196	172214	SIPDR6= 172214
197	172216	SIPDR7= 172216
198		
199		;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
200		
201	172220	SDPDR0= 172220
202	172222	SDPDR1= 172222
203	172224	SDPDR2= 172224
204	172226	SDPDR3= 172226
205	172230	SDPDR4= 172230
206	172232	SDPDR5= 172232
207	172234	SDPDR6= 172234
208	172236	SDPDR7= 172236
209		
210		;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
211		
212	172240	SIPAR0= 172240
213	172242	SIPAR1= 172242
214	172244	SIPAR2= 172244
215	172246	SIPAR3= 172246
216	172250	SIPAR4= 172250

217	172252	SIPAR5= 172252
218	172254	SIPAR6= 172254
219	172256	SIPAR7= 172256
220		
221		;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
222		
223	172260	SDPAR0= 172260
224	172262	SDPAR1= 172262
225	172264	SDPAR2= 172264
226	172266	SDPAR3= 172266
227	172270	SDPAR4= 172270
228	172272	SDPAR5= 172272
229	172274	SDPAR6= 172274
230	172276	SDPAR7= 172276
231		
232		;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
233		
234	172300	KIPDR0= 172300
235	172302	KIPDR1= 172302
236	172304	KIPDR2= 172304
237	172306	KIPDR3= 172306
238	172310	KIPDR4= 172310
239	172312	KIPDR5= 172312
240	172314	KIPDR6= 172314
241	172316	KIPDR7= 172316
242		
243		;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
244		
245	172320	KDPDR0= 172320
246	172322	KDPDR1= 172322
247	172324	KDPDR2= 172324
248	172326	KDPDR3= 172326
249	172330	KDPDR4= 172330
250	172332	KDPDR5= 172332
251	172334	KDPDR6= 172334
252	172336	KDPDR7= 172336
253		
254		;*KERNEL 'I' PAGE ADDRESS REGISTERS
255		
256	172340	KIPAR0= 172340
257	172342	KIPAR1= 172342
258	172344	KIPAR2= 172344
259	172346	KIPAR3= 172346
260	172350	KIPAR4= 172350
261	172352	KIPAR5= 172352
262	172354	KIPAR6= 172354
263	172356	KIPAR7= 172356
264		
265		;*KERNEL 'D' PAGE ADDRESS REGISTERS
266		
267	172360	KDPAR0= 172360
268	172362	KDPAR1= 172362
269	172364	KDPAR2= 172364
270	172366	KDPAR3= 172366

CZNCCA NCV11
CZNCCA.P11

DIAGNOSTIC
MEMORY MANAGEMENT DEFINITIONS

MACY11 27(654)

8-AUG-78 15:05

F 2
PAGE 6

SEQ 0018

271	172370	KDPAR4= 172370
272	172372	KDPAR5= 172372
273	172374	KDPAR6= 172374
274	172376	KDPAR7= 172376

275			
276			
277	172760	ABASE=172760	:NCV11 BASE ADDRESS
278	140370	AVECT1=140370	:BR LEVEL 6 VECTOR TO 370
279	170420	ACDW1=170420	:NCV11 CLOCK ADDRESS

280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307

```
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:* SWITCH USE
:* -----
:* 15 HALT ON ERROR
:* 14 LOOP ON TEST
:* 13 INHIBIT ERROR TYPEOUTS
:* 11 INHIBIT ITERATIONS
:* 10 BELL ON ERROR
:* 9 LOOP ON ERROR
:* 8 LOOP ON TEST IN SWR<7:0>
```

.SBTTL TRAP CATCHER

000000

```
.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
```

000174 000000
000176 000000

```
.=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
```

000200 000137 002032
000204 000137 002012
000210 000137 002020

```
.SBTTL STARTING ADDRESS(ES)
JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
JMP RTRT ;RESTART ADDRESS
JMP TESTER ;STARTING ADDRESS WHEN IN THE OPTION AREA
```

000100 000104 000200 000002

```
.=100
104,200,2 ;B EVENT SAFE GUARD
```

```

308      .SBTTL  ACT11 HOOKS
309
310      :*****
311      :HOOKS REQUIRED BY ACT11
312      $SVPC=.          ;SAVE PC
313      .=46
314  000046  030164      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
315      000052  000052      .=52
316      000052  000000      .WORD  0          ;;2)SET LOC.52 TO ZERO
317      000106      .= $SVPC          ;; RESTORE PC
318      001000      .=1000
319      .SBTTL  APT PARAMETER BLOCK
320
321      :*****
322      :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
323      :*****
324      $X=.          ;;SAVE CURRENT LOCATION
325      =24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
326  000024  000200      200          ;;FOR APT START UP
327      =44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
328  000044  001000      $APTHDR ;;POINT TO APT HEADER BLOCK
329      =.$X        ;;RESET LOCATION COUNTER
330      :*****
331      :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
332      :INTERFACE SPEC.
333
334  001000      $APTHD:
335  001000  000000      $HIBTS: .WORD  0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
336  001002  001174      $MBADR: .WORD  $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
337  001004  000030      $TSTM:  .WORD  30          ;;RUN TIM OF LONGEST TEST
338  001006  000010      $PASTM: .WORD  10          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
339  001010  000030      $UNITM: .WORD  30          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
340  001012  000031      .WORD  $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394

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

.SBTTL COMMON TAGS

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

.=1100
\$CMTAG: .WORD 0 ::START OF COMMON TAGS
\$STNM: .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

395	001206	000000	\$UNIT: .WORD	AUNIT	:: I/O UNIT NUMBER
396	001210	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
397	001212	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
398	001214		\$ETABLE:		:: APT ENVIRONMENT TABLE
399	001214	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
400	001215	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
401	001216	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
402	001220	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
403	001222	000000	\$CPUOP: .WORD	ACPUOP	:: CPU TYPE, OPTIONS
404			::*		BITS 15-11=CPU TYPE
405			::*		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
406			::*		11/70=06, PDQ=07, Q=10
407			::*		BIT 10=REAL TIME CLOCK
408			::*		BIT 9=FLOATING POINT PROCESSOR
409			::*		BIT 8=MEMORY MANAGEMENT
410	001224	000	\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
411	001225	000	\$MTYP1: .BYTE	AMTYP1	:: MEM. TYPE, BLK#1
412			::*		MEM. TYPE BYTE -- (HIGH BYTE)
413			::*		900 NSEC CORE=001
414			::*		300 NSEC BIPOLAR=002
415			::*		500 NSEC MOS=003
416	001226	000000	\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
417			::*		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF 'TYPE' ABOVE
418	001230	000	\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
419	001231	000	\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
420	001232	000000	\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
421	001234	000	\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
422	001235	000	\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
423	001236	000000	\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
424	001240	000	\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
425	001241	000	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
426	001242	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
427	001244	140370	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
428	001246	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
429	001250	172760	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
430	001252	000000	\$DEVN: .WORD	ADEVN	:: DEVICE MAP
431	001254	170420	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
432	001256		\$ETEND:		
433			.MEXIT		

434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487

.SBTTL ERROR POINTER TABLE

THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 *THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 *LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 *NOTE 1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 *NOTE 2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

* EM ::POINTS TO THE ERROR MESSAGE
 * DH ::POINTS TO THE DATA HEADER
 * DT ::POINTS TO THE DATA
 * DF ::POINTS TO THE DATA FORMAT

\$ERRTB:

EM1,DH2,DT2,DF0	;M8026	NCV11 TIMEOUT
EM2,DH2,DT2,DF0	;M8026	COMMAND-STATUS REGISTER ERROR
EM3,DH2,DT2,DF0	;M8026	SPECIAL FUNCTION REGISTER ERROR
EM4,DH2,DT2,DF0	;M8026	WORD COUNT REGISTER ERROR
EM5,DH2,DT2,DF0	;M8026	BUS ADDRESS REGISTER ERROR
EM6,DH2,DT2,DF0	;M8026	OFFSET REGISTER ERROR
EM7,DH2,DT2,DF0	;M8026	DUAL REGISTER SELECTION ERROR
EM10,DH2,DT2,DF0	;M8026-M8036	LOW 16 BIT Z COUNT ERROR
EM11,DH2,DT2,DF0	;M8026-M8036	HIGH 16 BIT Z COUNT ERROR
EM12,DH2,DT2,DF0	;M8026	Z COUNT STATUS ERROR
EM13,DH2,DT2,DF0	;M8026	Z COUNT INTERRUPT ERROR
EM14,DH2,DT2,DF0	;M8036	JOYSTICK STATUS ERROR
EM15,DH2,DT2,DF0	;M8036	JOYSTICK DATA ERROR
EM16,DH2,DT2,DF0	;M8036	DATA INCREMENT ERROR
EM17,DH2,DT2,DF0	;M8036	DATA DECREMENT ERROR
EM20,DH2,DT2,DF0	;M8026-M8036	MATRIX MODE ADDRESS MAKER DATA ERROR
EM21,DH2,DT2,DF0	;M8026	LIST MODE ADDRESS MAKER DATA ERROR
EM22,DH2,DT2,DF0	;M8026	LIST MODE TRANSFER BUS ADDRESS ERROR
EM23,DH2,DT2,DF0	;M8026	LIST MODE TRANSFER WORD COUNT ERROR
EM24,DH2,DT2,DF0	;M8026	LIST MODE TRANSFER OFFSET ERROR

488	001514	033002							
489	001516	031500	032500	032730	EM25,DH2,DT2,DF0		:M8026	TIMEOUT STATUS ERROR	
490	001524	033002							
491	001526	031533	032500	032730	EM26,DH2,DT2,DF0		:M8026	TIMEOUT INTERRUPT ERROR	
492	001534	033002							
493	001536	031571	032500	032730	EM27,DH2,DT2,DF0		:M8026	SET 'EVENT' OR 'TIME' DATA ERROR	
494	001544	033002							
495	001546	031640	032500	032730	EM30,DH2,DT2,DF0		:M8026	CELL INCREMENT DATA ERROR	
496	001554	033002							
497	001556	031700	032500	032730	EM31,DH2,DT2,DF0		:M8026	CELL OVERFLOW STATUS ERROR	
498	001564	033002							
499	001566	031741	032500	032730	EM32,DH2,DT2,DF0		:M8026	CELL OVERFLOW INTERRUPT ERROR	
500	001574	033002							
501	001576	032005	032524	032742	EM33,DH3,DT3,DF0		:M8026	MATRIX MODE ADDRESS MUX ERROR	
502	001604	033002							
503	001606	032051	032500	032730	EM34,DH2,DT2,DF0		:M8026	'TESTX' FUNCTION ERROR	
504	001614	033002							
505	001616	032106	032557	032756	EM35,DH4,DT4,DF0		:M8026	DATA ERROR WHEN TRANSFERING TO EXTENDED MEMORY	
506	001624	033002							
507	001626	032173	032500	032730	EM36,DH2,DT2,DF0		:M8026	LIST MODE TRANSFER STATUS ERROR	
508	001634	033002							
509	001636	032241	032524	032742	EM37,DH3,DT3,DF0		:M8026	LIST MODE TRANSFER DATA ERROR	
510	001644	033002							
511	001646	032305	032500	032730	EM40,DH2,DT2,DF0		:JUMPER-M8026-M7952	'EVENT' OR 'TIME' MARK ERROR	
512	001654	033002							
513	001656	032365	032465	032774	EM41,DH1,DT5,DF0		:M7952	CLOCK TIMEOUT	
514	001664	033002							
515	001666	032425	032465	032722	EM42,DH1,DT1,DF0		:M8217	INTERRUPT LEVEL ERROR	
516	001674	033002							
517									
518	001676	176510							
519	001700	176512							
520	001702	176514							
521	001704	176516							
522									
523									
524	001706	172600							
525	001710	172602							
526	001712	172603							
527	001714	172604							
528	001716	172620							
529	001720	172622							
530	001722	172624							
531	001724	172630							
532									
533									
534	001726	176416							
535	001730	176420							
536	001732	176422							
537	001734	176424							
538	001736	176426							

:DL11 BLASTER COMM. ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
 DLICSR: 176510 ;INPUT STATUS REG.
 DLIBD: 176512 ;INPUT DATA
 DLOCSR: 176514 ;OUTPUT STATUS
 DLODB: 176516 ;OUTPUT DATA

:VSV01 ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
 VTCHAR: 172600 ;CHAR. STATUS
 VTYPOS: 172602
 VTXPOS: 172603
 VTCXY: 172604
 VTCRSR: 172620 ;MAP STATUS
 VTMAP: 172622
 VTPX: 172624
 VTINT: 172630

:TESTER ADDRESSES (THE OPERATOR MUST CHANGE IF DIFFERENT)
 DACSR: 176416 ;KNOWN GOOD D/A STATUS
 DAC0: 176420
 DAC1: 176422
 DAC2: 176424
 DAC3: 176426

```
539 ;KVV11 PROGRAM GENERATED ADDRESSES (THE OPER. ONLY HAS TO CHANGE '$CDW1')
540
541 001740 170420 KWCSR: ACDW1 ;KVV11 STATUS REGISTER
542 001742 170421 KWCSR1: ACDW1+1 ;KVV11 STATUS REG. HIGH BYTE
543 001744 170422 KWPSR: ACDW1+2 ;KVV11 PRESET REGISTER
544
545 ;NCV11 PROGRAM GENERATER ADDRESSES (THE OPER. ONLY HAS TO CHANGE '$BASE')
546
547 001746 172760 CSR: ABASE
548 001750 172762 OFF: ABASE+2
549 001752 172764 WCR: ABASE+4
550 001754 172766 BAR: ABASE+6
551 001756 172770 SFR: ABASE+10
552 001760 172772 ADM: ABASE+12
553 001762 172774 JOY: ABASE+14
554 001764 172776 BAR1: ABASE+16
555 001766 172761 CSRHB: ABASE+1
556 001770 172771 SFRHB: ABASE+11
557
558 ;NCV11 PROGRAM GENERATED VECTORS (THE OPER. ONLY HAS TO CHANGE LOW 9 BITS OF '$VECT1')
559
560 001772 140370 VECTA0: AVECT1
561 001774 140372 VECTA1: AVECT1+2
562 001776 140374 VECTB0: AVECT1+4
563 002000 140376 VECTB1: AVECT1+6
564
565 ;NCV11 PROGRAM GENERATED BR LEVEL (THE OPER. ONLY HAS TO CHANGE TOP 3 BITS OF '$VECT1')
566
567 002002 000300 BRLEV: 300 ;BR LEVEL 6
568
569 002004 000000 $TEMP: 0
570 002006 000001 CPUDLO: 1
571 002010 000020 CPUDL1: 20
572
573 040000 CLRWCO=BIT14
574 004000 CLRALL=BIT11
575 000400 ENDDMA=BIT8
576 000010 TSTDMA=BIT3
577 000004 TSTCON=BIT2
578 000002 TESTZ=BIT1
579 000001 REDJOY=BIT0
580 000000 MORTST=0 ;DEFINE TO RESTORE TEST CODE
581
```

```

582 002012 005237 050116      RTRT:  INC      AGAN      ;SET RESTART FLAG
583 002016 000411              BR      RTRTA
584 002020 005237 050110      TESTER: INC      WFMODE   ;SET FLAG INDICATING OPTION AREA MODE
585 002024 005037 050116      CLR      AGAN
586 002030 000404              BR      RTRTA
587 002032 005037 050116      BEGIN: CLR      AGAN
588 002036 005037 050110      CLR      WFMODE   ;CLEAR FLAG INDIC. OPTION TEST AREA MODE
589 002042
590
591      .SBTTL  INITIALIZE THE COMMON TAGS
      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
592 002042 012706 001100      MOV      #$CMTAG,R6    ;;FIRST LOCATION TO BE CLEARED
593 002046 005026              CLR      (R6)+         ;;CLEAR MEMORY LOCATION
594 002050 022706 001140      CMP      #SWR,R6    ;;DONE?
595 002054 001374              BNE      -6           ;;LOOP BACK IF NO
596 002056 012706 001100      MOV      #STACK,SP    ;;SETUP THE STACK POINTER
597      ;;INITIALIZE A FEW VECTORS
598 002062 012737 034106 000020  MOV      #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
599 002070 012737 000340 000022  MOV      #340,@#IOTVEC+2 ;;LEVEL 7
600 002076 012737 035102 000030  MOV      #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
601 002104 012737 000340 000032  MOV      #340,@#EMTVEC+2 ;;LEVEL 7
602 002112 012737 037266 000034  MOV      #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
603 002120 012737 000340 000036  MOV      #340,@#TRAPVEC+2;LEVEL 7
604 002126 012737 035704 000024  MOV      #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
605 002134 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;LEVEL 7
606 002142 013737 030132 030124  MOV      $ENDCT,$EOPCT  ;;SETUP END-OF-PROGRAM COUNTER
607 002150 005037 001160      CLR      $TIMES       ;;INITIALIZE NUMBER OF ITERATIONS
608 002154 005037 001162      CLR      $ESCAPE      ;;CLEAR THE ESCAPE ON ERROR ADDRESS
609 002160 112737 000001 001115  MOVB     #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
610 002166 012737 002166 001106  MOV      #,$SLPADR     ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
611 002174 012737 002174 001110  MOV      #,$SLPERR     ;;SETUP THE ERROR LOOP ADDRESS
612      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
613      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
614 002202 013746 000004      MOV      @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
615 002206 012737 002242 000004  MOV      #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
616 002214 012737 177570 001140  MOV      #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
617 002222 012737 177570 001142  MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
618 002230 022777 177777 176702  CMP      #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
619 002236 001012      BNE     66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
620      ;;AND THE HARDWARE SWR IS NOT = -1
621 002240 000403      BR      65$         ;;BRANCH IF NO TIMEOUT
622 002242 012716 002250      64$:  MOV      #65$,(SP)   ;;SET UP FOR TRAP RETURN
623 002246 000002
624 002250 012737 000176 001140  65$:  MOV      #SWREG,SWR   ;;POINT TO SOFTWARE SWR
625 002256 012737 000174 001142  MOV      #DISPREG,DISPLAY
626 002264 012637 000004      66$:  MOV      (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
627
628 002270 005037 001202      CLR      $PASS       ;;CLEAR PASS COUNT
629 002274 132737 000200 001215  BITB    #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
630 002302 001403      BEQ     67$         ;;YES,USE NON-APT SWITCH
631 002304 012737 001216 001140  MOV      #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
632 002312      67$:

```

```

633 .SBTTL TYPE PROGRAM NAME
634 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
635 002312 005227 177:77 INC #-1 ;;FIRST TIME?
636 002316 001044 BNE 68$ ;;BRANCH IF NO
637 002320 022737 030164 000042 CMP #SENDAD,@#42 ;;ACT-11?
638 002326 001440 BEQ 68$ ;;BRANCH IF YES
639 002330 104401 002376 TYPE ,69$ ;;TYPE ASCIZ STRING
640 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
641 002334 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
642 002340 001012 BNE 70$ ;;BRANCH IF YES
643 002342 123727 001214 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
644 002350 001406 BEQ 70$ ;;BRANCH IF YES
645 002352 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
646 002360 001005 BNE 71$ ;;BRANCH IF NO
647 002362 104406 GTSWR ;;GET SOFT-SWR SETTINGS
648 002364 000403 BR 71$
649 002366 112737 000001 001134 70$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
650 002374 71$:
651 002374 000415 BR 68$ ;;GET OVER THE ASCIZ
652 ;;69$: .ASCIZ <CRLF>#CZNCCA NCV11 DIAGNOSTIC#<CRLF>
653 002430 68$:
654 .SBTTL PRIME THE NCV11 ADDRESSES FROM THE DEFAULT VALUES
655 002430 112737 000103 052474 MOVB #'C,RUNIT ;:LOAD "CONTROL" SECTION BUS TRAP FLAG
656 002436 012737 003210 000004 MOV #BUSTRP,@#ERRVEC ;:LOAD BUS TRAP RETURN ADDRESS
657 002444 012700 001746 MOV #CSR,R0 ;:GET ADDRESS POINTER
658 002450 013701 001250 MOV $BASE,R1 ;:GET DEFAULT ADDRESS
659 002454 010120 1$: MOV R1,(R0)+ ;:LOAD AN VALUE
660 002456 062701 000002 ADD #2,R1 ;:BUMP THE VALUE
661 002462 022700 001766 CMP #CSRHB,R0 ;:TEST IF DONE
662 002466 001372 BNE 1$ ;:BR IF NOT
663 002470 013710 001250 MOV $BASE,(R0) ;:LOAD ODD BYTE
664 002474 005220 INC (R0)+ ;:ADDRESSES
665 002476 013710 001250 MOV $BASE,(R0)
666 002502 062720 000011 ADD #11,(R0)+
667 002506 013701 001244 MOV $VECT1,R1 ;:GET DEFAULT VECTOR
668 002512 010102 MOV R1,R2 ;:COPY R1
669 002514 000302 SWAB R2 ;:EXCHANGE BYTES
670 002516 042702 177437 BIC #177437,R2 ;:STRIP OFF ALL BUT BR LEVEL BITS
671 002522 010237 002002 MOV R2,BRLEV ;:SAVE FOR USE LATER
672 002526 042701 160000 BIC #160000,R1 ;:CLEAR OFF BR LEVEL
673 002532 010120 2$: MOV R1,(R0)+ ;:LOAD VECTOR
674 002534 005721 TST (R1)+ ;:BUMP THE VALUE
675 002536 022700 002002 CMP #VECTB1+2,R0 ;:TEST IF DONE
676 002542 001373 BNE 2$
677 002544 013737 001254 001740 MOV $CDW1,KWCSR ;:GET CLOCK ADDRESS
678 002552 013737 001740 001742 MOV KWCSR,KWCSR1 ;:SET CLOCK HIGH BYTE ADDR.
679 002560 013737 001740 001744 MOV KWCSR,KWPSR ;:SET CLOCK PRESET ADDRESS
680 002566 005237 001742 INC KWCSR1
681 002572 062737 000002 001744 ADD #2,KWPSR

```

```

682 .SBTTL DIAGNOSTIC IDENTIFICATION TYPEOUT
683 002600 005737 050116 TST AGAN ;TEST IF RESTART
684 002604 001017 BNE RBEGO ;BR IF YES
685 002606 105737 001134 TSTB $AUTOB ;TEST IF ACT/APT
686 002612 001405 BEQ 3$ ;BR IF NOT
687 002614 012737 177777 050106 MOV #-1,RUNDIF ;INDICATE TO RUN DIFLIN IN CHAIN MODE
688 002622 000137 003356 JMP LOGIC ;YES; RUN LOGIC AND DIFLIN TESTS
689 002626 005737 050110 3$: TST WFMODE ;TEST IF OPTION CHECKOUT MODE
690 002632 001402 BEQ 4$ ;BR IF YES
691 002634 104401 TYPE
692 002636 046256 LISTO ;TELL OPERATOR ABOUT OPTION AREA TESTS
693 002640 104401 4$: TYPE
694 002642 046466 LIST ;TELL OPERATOR ABOUT THE TEST
695 .SBTTL KEYBOARD COMMAND DECODER
696 002644 104401 047101 RBEGO: TYPE, LIST1 ;PRINT 'DOT'
697 002650 104411 1$: RDLIN ;READ THE RESPONSE
698 002652 013637 052474 MOV @(SP)+,RUNIT ;GET THE CHARACTER
699 002656 142737 000040 052474 BICB #40,RUNIT ;ENSURE UPPER CASE
700 002664 112737 000042 052475 MOVB #'',RUNIT+1 ;FIX ASCII MESSAGE
701 002672 122737 000101 052474 CMPB #'A',RUNIT ;DETERMINE IF AN A
702 002700 001002 BNE 2$ ;BR IF NOT
703 002702 000137 003276 JMP TFSITE ;RUN SITE ADJUSTMENT LOOP
704 002706 005737 050110 2$: TST WFMODE ;TEST IF ON THE TESTER
705 002712 001406 BEQ 3$ ;BR IF NOT
706 002714 122737 000102 052474 CMPB #'B',RUNIT ;DETERMINE IF AN B
707 002722 001002 BNE 3$ ;BR IF NOT
708 002724 000137 042322 JMP BLAST ;RUN THE BLASTER SECTION
709 002730 122737 000104 052474 3$: CMPB #'D',RUNIT ;DETERMINE IF AN D
710 002736 001004 BNE 4$ ;BR IF NOT
711 002740 005037 050106 CLR RUNDIF ;INDICATE DIFLIN IS NOT TO BE RUN
712 002744 000137 043712 JMP DIFLIN ;AND RUN IT
713 002750 122737 000106 052474 4$: CMPB #'F',RUNIT ;DETERMINE IF AN F
714 002756 001005 BNE 5$ ;BR IF NOT
715 002760 012737 177777 050106 MOV #-1,RUNDIF ;INDICATE RUN DIFLIN AFTER THE LOGIC TEST
716 002766 000137 003356 JMP LOGIC ;AND RUN LOGIC TEST
717 002772 005737 050110 5$: TST WFMODE ;TEST IF OPTION CHECKOUT MODE
718 002776 001406 BEQ 6$ ;BR IF NOT
719 003000 122737 000111 052474 CMPB #'I',RUNIT ;DETERMIN IF AN I
720 003006 001002 BNE 6$ ;BR IF NOT
721 003010 000137 047106 JMP POTIME ;RUN IN-HOUSE ADJUSTMENT LOOP
722 003014 122737 000114 052474 6$: CMPB #'L',RUNIT ;DETERMINE IF AN L
723 003022 001004 BNE 7$ ;BR IF NOT
724 003024 005037 050106 CLR RUNDIF ;INDICATE NO DIFLIN
725 003030 000137 003356 JMP LOGIC
726 003034 122737 000110 052474 7$: CMPB #'H',RUNIT ;DETERMINE IF AN H
727 003042 001002 BNE 10$ ;BR IF NOT
728 003044 000137 002032 JMP BEGIN ;START OVER
729 003050 122737 000037 052474 10$: CMPB #37,RUNIT ;DETERMINE IF AN '?'
730 003056 001002 BNE 11$ ;BR IF NOT
731 003060 000137 002032 JMP BEGIN ;START OVER
732 003064 005737 050110 11$: TST WFMODE ;TEST IF OPTION CHECKOUT MODE
733 003070 001406 BEQ 12$ ;BR IF NOT
734 003072 122737 000124 052474 CMPB #'T',RUNIT ;DETERMINE IF AN T
735 003100 001002 BNE 12$ ;BR IF NOT

```

736	003102	000137	042050			JMP	BTALK	:KEYBOARD LOOP WITH BLASTER
737	003106	005737	050110	12\$:		TST	WFMODE	:TEST IF OPTION CHECKOUT MODE
738	003112	001406				BEQ	13\$:BR IF NOT
739	003114	122737	000103	052474		CMPB	#'C,RUNIT	:DETERMINE IF AN C
740	003122	001002				BNE	13\$:BR IF NOT
741	003124	000137	043004			JMP	PBLAST	:RUN M8036 PROM BLASTING CODE
742	003130	122737	000117	052474	13\$:	CMPB	#'O,RUNIT	:DETERMINE IF AN O
743	003136	001002				BNE	14\$:BR IF NOT
744	003140	000137	003240			JMP	XITY	:CHANGE THE TTY ADDRESS
745	003144	122737	000123	052474	14\$:	CMPB	#'S,RUNIT	:DETERMINE IF AN S
746	003152	001002				BNE	15\$:BR IF NOT
747	003154	104406				GTSWR		:GET SWITCH REGISTER VALUE
748	003156	000632				BR	RBEGO	:RESTART
749	003160	122737	000115	052474	15\$:	CMPB	#'M,RUNIT	:DETERMINE IF AN M
750	003166	001005				BNE	FATFNG	:BR IF NOT
751	003170	012737	000377	050106		MOV	#377,RUNDIF	:INDICATE DIFLIN WITH EXPAND REPORT
752	003176	000137	043712			JMP	DIFLIN	:RUN DIF LIN

```
753      ;THE OPERATOR SELECTED THE WRONG THING - THEY SOMETIME HAVE 'FAT FINGERS''
754 003202 104401 FATFNG: TYPE
755 003204 001170      $QUES      ;TYPE QUESTION MARK
756 003206 000616      BR      RBEGO
757
758
759      ;RETURN TO HERE UPON UNEXPECTED BUS TRAP
760 003210 104401 052424 BUSTRP: TYPE, FATALO      ;REPORT FATAL TRAP TO THE OPERATOR
761 003214 011646      MOV      (SP),-(SP)      ;GET TRAP ADDRESS
762 003216 104402      TYPOC
763 003220 005777 175714 1$: TST      @SWR      ;TEST IF HALT ON ERROR
764 003224 100001      BPL      2$      ;BR IF SET
765 003226 000000      HALT      ;FATAL BUS TRAP DETECTED
766 003230 104401 052514 2$: TYPE, RUNITA
767 003234 000137 002012      JMP      RTRT      ;RESTART
768
769      ;ROUTINE TO INPUT NEW TTY ADDRESS FROM THE OPERATOR
770 003240 104401 052604 XTTY: TYPE, WARNO      ;TELL THE OPERATOR THE RULES
771 003244 104412      RDOCT      ;LISTEN TO THE OPERATOR
772 003246 012600      MOV      (SP)+,R0      ;READ THEIR INPUT
773 003250 001754      BEQ      FATFNG      ;BR IF JUST 'CR'
774 003252 005710      TST      (R0)      ;ADDRESS THE DEVICE AND CHECK BUS TRAP
775      ;IF NO BUS TRAP - THEY MUST KNOW WHAT THEY ARE DOING !
776 003254 012701 001144      MOV      #$TKS,R1      ;GET POINTER
777 003260 010021      1$: MOV      R0,(R1)+      ;LOAD THE VALUE
778 003262 005720      TST      (R0)+      ;BUMP VALUE
779 003264 020127 001154      CMP      R1,$$TKS+10      ;TEST IF DONE ALL ADDRESSES
780 003270 001373      BNE      1$      ;BR IF NOT
781 003272 000137 002042      JMP      RTRTA      ;RETYPE THE HEADER AGAIN ON THE NEW TERMINAL
782
783      ;OPERATOR CHOSE THE FIELD ADJUSTMENT LOOP
784 003276 104401 051250 TFSITE: TYPE, PRIM6      ;TELL OPER. ABOUT A017
785 003302 104401 041374      TYPE, FIELDI      ;ASK THE OPERATOR FOR INPUT Z MODE
786 003306 005037 040410      CLR      INOUTZ      ;DEFAULT TO USER SUPPLIED Z PULSES
787 003312 104411      RDLIN      ;READ OPER. INPUT
788 003314 013637 003350      MOV      @(SP)+,10$      ;GET CHAR
789 003320 142737 000040 003350      BICB      #40,10$      ;ENSURE UPPER CASE
790 003326 122737 000131 003350      CMPB      #'Y,10$      ;TEST FOR 'Y' INPUT
791 003334 001003      BNE      1$      ;BR IF NOT 'Y'
792 003336 012737 000002 040410      MOV      #2,INOUTZ      ;SET MAINT Z CONSTANT
793 003344 000137 037354      1$: JMP      FSITE      ;NOW DO THE LOOP
794 003350 000000      10$: 0
795
796 003352 000000      ADNOKT: 0      ;LAST ADDRESS WITHOUT M.M. OR XXDP
797 003354 000000      NLSI11: 0      ;NON-ZERO INDICATES LSI-11 PROCESSOR
```

```

798
799 003356 000005          LOGIC:  RESET
800 003360 005737 001202      TST      $PASS          ;TEST IF FIRST PASS
801 003364 001124          BNE      TST1           ;:BR IF NOT
802 003366 005737 050110      TST      WFMODE        ;CHECK IF ON TESTER
803 003372 001402          BEQ      5$             ;BR IF NOT
804 003374 104401 052332      TYPE,   PRIM4         ;TELL OPERATOR CABLE MUST BE CONNECTED
805 003400 005037 036120      5$:    CLR      $KT11   ;INDICATE NO KT11
806 003404 004737 036062      JSR     PC,$SIZE      ;SIZE BASIC MEMORY SIZE
807 003410 162737 001000 036400  SUB     #1000,$LSTAD   ;PROTECT ABSLDR
808 003416 005737 000042      TST     @#42          ;TEST IF XXDP CHAIN MODE ?
809 003422 001403          BEQ      1$             ;BR IF NOT CHAIN MODE
810 003424 162737 006200 036400  SUB     #3200.,$LSTAD ;LEAVE ROOM FOR XXDP MONITOR
811 003432 013737 036400 003352 1$:    MOV     $LSTAD,ADNOKT ;SAVE BASIC MEMORY STAGE
812 003440 012737 000200 036120  MOV     #BIT7,$KT11   ;WITH M.M.
813 003446 004737 036062      JSR     PC,$SIZE      ;DETERMINE THE AMOUNT OF MEMORY
814 003452 005737 036120      TST     $KT11        ;TEST IF KT11 IS HERE
815 003456 100025          BPL     2$             ;BR IF NOT
816 003460 012737 000200 172342  MOV     #200,@#KIPAR1 ;
817 003466 012737 077406 172302  MOV     #77406,@#KIPDR1
818 003474 012737 000400 172344  MOV     #400,@#KIPAR2
819 003502 012737 077406 172304  MOV     #77406,@#KIPDR2
820 003510 012737 077406 172306  MOV     #77406,@#KIPDR3 ;4K, UP R/W FOR 60000
821 003516 012737 007600 172356  MOV     #7600,@#KIPAR7 ;I/O PAGE MAP
822 003524 012737 077406 172316  MOV     #77406,@#KIPDR7 ;4K, UP, R/W FOR I/O PAGE ACCESS
823 003532 105737 001134      2$:    TSTB   $AUTOB    ;TEST IF AUTO MODE
824 003536 001016          BNE     4$             ;:BR IF AUTO MODE
825 003540 104401 052545      TYPE,   MSGMEM        ;TELL OPERATOR THE AMOUNT OF MEMORY
826 003544 013700 036402      MOV     $LSTBK,RO     ;GET LAST GOOD BANK
827 003550 006200          ASR     R0
828 003552 006200          ASR     R0
829 003554 006200          ASR     R0
830 003556 006200          ASR     R0
831 003560 006200          ASR     R0
832 003562 005200          INC     R0
833 003564 010046          MOV     R0,-(SP)      ;CONVERT
834 003566 104405          TYPDS   ;TYPE OUT DEC. VALUE
835 003570 104401 052565      TYPE,   MSGK          ;AND THEN TYPE OUT THE REMAINDER OF MEMORY MESSAGE
836 003574 013737 000004 002004 4$:    MOV     @#ERRVEC,$TEMP ;SAVE LOC 4 VALUE
837 003602 012737 003630 000004  MOV     #3$,@#ERRVEC  ;LOAD LOC 4
838 003610 005037 003354          CLR     NLSI11       ;INDICATE NOT A LSI-11 CPU
839 003614 005037 177776          CLR     PSW          ;SHOULD FAIL IF LSI-11
840 003620 013737 002004 000004  MOV     $TEMP,@#ERRVEC ;RESTORE LOC 4 VALUE
841 003626 000403          BR      TST1         ;:BR IF TEST
842 003630 005237 003354      3$:    INC     NLSI11    ;SET AN LSI-11 FLAG
843 003634 000002          RTI                    ;RETURN

```



```
844  
845  
846  
847 003636 000004  
848 003640 012737 000100 001160  
849 003646 013737 000004 004014  
850 003654 012737 003752 000004  
851 003662 005777 176060  
852 003666 005777 176056  
853 003672 005777 176054  
854 003676 005777 176052  
855 003702 005777 176050  
856 003706 005777 176046  
857 003712 005777 176046  
858 003716 012737 003760 000004  
859 003724 005777 176010  
860 003730 005777 176010  
861  
862 003734 012737 000001 004016  
863 003742 013737 004014 000004  
864 003750 000423  
865 003752 022626  
866 003754 104001  
867 003756 000411  
868 003760 022626  
869 003762 104041  
870 003764 012737 000001 004016  
871 003772 013737 004014 000004  
872 004000 000407  
873 004002 013737 004014 000004  
874 004010 000137 030076  
875 004014 000000  
876 004016 000000  
:*****  
:*TEST 1 VERIFY THE 8 NCV11 AND 2 NCV11 CLOCK BUS ADDRESSES RESPOND  
:*****  
TST1: SCOPE  
MOV #100,$TIMES ;;DO 100 ITERATIONS  
MOV @#ERRVEC,10$ ;SAVE BUS TRAP VECTOR  
MOV #1$,@#ERRVEC ;LOAD BUS TRAP VECTOR  
TST @CSR ;ADDRESS  
TST @OFF ;  
TST @WCR ;THE  
TST @BAR ;  
TST @SFR ;NCV11  
TST @ADM ;  
TST @BAR1 ;ADDRESSES  
MOV #2$,@#ERRVEC ;LOAD NEW RETURN  
TST @KWCSR ;TEST CLOCK  
TST @KWPSR ;ADDRESSES  
;TEMP FIX THE DIAG TO NOT USE CLOCK JUMPERS  
MOV #1,DEADKW ;INDICATE NCV11 CLOCK IS NOT THERE;TEMP. FIX UNTIL ECO M  
MOV 10$,@#ERRVEC ;RESTORE THE BUS TRAP VECTOR  
BR TST2 ;;BR AND TEST THE NCV11  
1$: CMP (SP)+,(SP)+ ;CLEAN THE STACK  
ERROR 1 ;BUS TRAP WHEN REFERENCING THE NCV11  
BR 3$  
2$: CMP (SP)+,(SP)+ ;CLEAN THE STACK  
ERROR 41 ;BUS TRAP WHEN REFERENCING THE NCV11 CLOCK  
MOV #1,DEADKW ;INDICATE NCV11 CLOCK IS NOT THERE  
MOV 10$,@#ERRVEC ;RESTORE LOC 4  
BR TST2 ;;  
3$: MOV 10$,@#ERRVEC ;RESTORE BUS TRAP VECTOR  
JMP $EOP ;EXIT  
10$: 0  
DEADKW: 0 ;NON-ZERO SAYS NO NCV11 CLOCK
```

```

877
878
879
880 004020 000004
881 004022 012737 000100 001160
882 004030 012737 004044 001110
883 004036 012737 004000 002004
884
885 004044 013777 002004 175674 1$: MOV $TEMP,@CSR ;LOAD CSR REG.
886 004052 017737 175670 001126 MOV @CSR,$BDDAT ;READ CSR
887 004060 013737 002004 001124 MOV $TEMP,$GDDAT ;LOAD EXPECTED
888 004066 052737 000200 001124 BIS #BIT7,$GDDAT ;FUDGE THE 'READY' BIT
889 004074 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE THE VALUES
890 004102 001401 BEQ 2$ ;;BR IF SAME
891 004104 104002 ERROR 2 ;UNEXPECTED VALUE IN THE CSR REGISTER
892
893 004106 006237 002004 2$: ASR $TEMP ;TRY THE NEXT DATA BIT
894 004112 022737 000001 002004 CMP #1,$TEMP ;TEST IF NOW BIT 0
895 004120 001351 BNE 1$ ;BR IF NOT
896
897
898
899
900 004122 000004
901 004124 012737 000010 001160
902 004132 012777 007776 175606
903 004140 000005
904 004142 012737 000200 001124
905 004150 017737 175572 001126
906 004156 023737 001124 001126
907 004164 001401
908 004166 104002
909
910
911
912
913 004170 000004
914 004172 012737 000010 001160
915 004200 012777 007776 175540
916 004206 012777 004000 175542
917 004214 012737 000200 001124
918 004222 017737 175520 001126
919 004230 023737 001124 001126
920 004236 001401
921 004240 104002
922
  
```

 *TEST 2 FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.

 TST2: SCOPE
 MOV #100,\$TIMES ;;DO 100 ITERATIONS
 MOV #1\$,\$LPERR ;LOAD LOOP ADDRESS
 MOV #BIT11,\$TEMP ;LOAD INITIAL REG. VALUE
 1\$: MOV \$TEMP,@CSR ;LOAD CSR REG.
 MOV @CSR,\$BDDAT ;READ CSR
 MOV \$TEMP,\$GDDAT ;LOAD EXPECTED
 BIS #BIT7,\$GDDAT ;FUDGE THE 'READY' BIT
 CMP \$GDDAT,\$BDDAT ;COMPARE THE VALUES
 BEQ 2\$;;BR IF SAME
 ERROR 2 ;UNEXPECTED VALUE IN THE CSR REGISTER
 2\$: ASR \$TEMP ;TRY THE NEXT DATA BIT
 CMP #1,\$TEMP ;TEST IF NOW BIT 0
 BNE 1\$;BR IF NOT

 *TEST 3 VERIFY THAT 'INIT' CLEARS THE CSR REGISTER

 TST3: SCOPE
 MOV #10,\$TIMES ;;DO 10 ITERATIONS
 MOV #7776,@CSR ;LOAD CSR REG.
 RESET ;INITILIZE THE REGISTER
 MOV #BIT7,\$GDDAT ;LOAD EXPECTED VALUE
 MOV @CSR,\$BDDAT ;READ CSR REG.
 CMP \$GDDAT,\$BDDAT ;COMPARE THE VALUES
 BEQ TST4 ;;BR IF EQUAL
 ERROR 2 ;'BUS INIT' FAILED TO CLEAR CSR REG.

 *TEST 4 VERIFY THAT 'CLEAR ALL' CLEARS THE CSR REGISTER

 TST4: SCOPE
 MOV #10,\$TIMES ;;DO 10 ITERATIONS
 MOV #7776,@CSR ;LOAD CSR REG.
 MOV #CLRALL,@SFR ;GENERATE 'CLR ALL L'
 MOV #BIT7,\$GDDAT ;LOAD EXPECTED VALUE
 MOV @CSR,\$BDDAT ;READ THE CSR REG.
 CMP \$GDDAT,\$BDDAT ;COMPARE VALUES
 BEQ TST5 ;;BR IF SAME
 ERROR 2 ;'CLR ALL L' FAILED TO CLEAR CSR REG.

```

923
924
925
926 004242 000004
927 004244 012737 000100 001160
928 004252 012777 003636 175466
929 004260 012737 003600 001124
930 004266 105077 175454
931 004272 017737 175450 001126
932 004300 023737 001124 001126
933 004306 001401
934 004310 104002
935 004312 012777 003636 175426 1$:
936 004320 012737 000236 001124
937 004326 105077 175434
938 004332 017737 175410 001126
939 004340 023737 001124 001126
940 004346 001401
941 004350 104003
942
943
944
945 004352 000004
946 004354 012737 004370 001110
947 004362 012737 000020 002004
948
949 004370 013777 002004 175360 1$:
950 004376 017737 175354 001126
951 004404 013737 002004 001124
952 004412 023737 001124 001126
953 004420 001401
954 004422 104003
955 004424 006237 002004 2$:
956 004430 022737 000001 002004
957 004436 001354
958
959
960
961
962 004440 000004
963 004442 012737 000010 001160
964 004450 012777 000014 175300
965 004456 012737 000014 001124
966 004464 105077 175300
967 004470 017737 175262 001126
968 004476 023737 001124 001126
969 004504 001401
970 004506 104003
971
  
```

```

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

```
972  
973  
974  
975 004510 000004  
976 004512 012737 000010 001160  
977 004520 012777 000016 175230  
978 004526 000005  
979 004530 005037 001124  
980 004534 017737 175216 001126  
981 004542 001401  
982 004544 104003  
983  
984  
985  
986  
987 004546 000004  
988 004550 012737 000010 001160  
989 004556 012777 000016 175172  
990 004564 052777 004000 175164  
991 004572 005037 001124  
992 004576 017737 175154 001126  
993 004604 001401  
994 004606 104003  
995
```

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

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

```

996
997
998
999 004610 000004
1000 004612 012737 000100 001160
1001 004620 012737 000001 001124
1002 004626 012737 004634 001110
1003
1004 004634 012777 004000 175114
1005 004642 013777 001124 175102
1006 004650 017737 175076 001126
1007 004656 023737 001124 001126
1008 004664 001401
1009 004666 104004
1010
1011 004670 006337 001124
1012 004674 001357
1013
1014
1015
1016 004676 000004
1017 004700 012737 000100 001160
1018 004706 012737 000001 001124
1019 004714 012737 004722 001110
1020
1021 004722 012777 004000 175026
1022 004730 013777 001124 175016
1023 004736 017737 175012 001126
1024 004744 023737 001124 001126
1025 004752 001401
1026 004754 104005
1027
1028 004756 006337 001124
1029 004762 001357
1030
1031
1032
1033 004764 000004
1034 004766 012737 000100 001160
1035 004774 012737 000001 001124
1036 005002 012737 005010 001110
1037
1038 005010 012777 004000 174740
1039 005016 013777 001124 174724
1040 005024 017737 174720 001126
1041 005032 023737 001124 001126
1042 005040 001401
1043 005042 104006
1044
1045 005044 006337 001124
1046 005050 001357

*****
*TEST 12      FLOAT A 1 ACROSS THE WORD COUNT REGISTER
*****
TST12:  SCOPE
        MOV      #100,$TIMES      ;;DO 100 ITERATIONS
        MOV      #BIT0,$GDDAT     ;LOAD EXPECTED VALUE
        MOV      #1$,$LPERR      ;LOAD LOOP ADDRESS ON ERROR
1$:     MOV      #CLRALL,@SFR     ;RESET THE DEVICE
        MOV      $GDDAT,@WCR     ;LOAD WORD COUNT 'A'
        MOV      @WCR,$BDDAT     ;READ WORD COUNT
        CMP      $GDDAT,$BDDAT   ;COMPARE VALUES
        BEQ      2$              ;;BR IF SAME
        ERROR    4              ;;WORD COUNT REG. IN ERROR
2$:     ASL      $GDDAT          ;CHANGE THE DATA
        BNE      1$              ;BR IF MORE DATA TO LOAD
*****
*TEST 13      FLOAT A 1 ACROSS THE BUS ADDRESS REGISTER
*****
TST13:  SCOPE
        MOV      #100,$TIMES     ;;DO 100 ITERATIONS
        MOV      #BIT0,$GDDAT    ;LOAD EXPECTED VALUE
        MOV      #1$,$LPERR     ;LOAD LOOP ADDRESS ON ERROR
1$:     MOV      #CLRALL,@SFR    ;RESET THE DEVICE
        MOV      $GDDAT,@BAR     ;LOAD BUS ADDRESS 'A'
        MOV      @BAR,$BDDAT    ;READ BUS ADDRESS
        CMP      $GDDAT,$BDDAT  ;COMPARE VALUES
        BEQ      2$              ;;BR IF SAME
        ERROR    5              ;BUS ADDRESS REG. IN ERROR
2$:     ASL      $GDDAT          ;CHANGE THE DATA
        BNE      1$              ;BR IF MORE DATA TO LOAD
*****
*TEST 14      FLOAT A 1 ACROSS THE OFFSET REGISTER
*****
TST14:  SCOPE
        MOV      #100,$TIMES     ;;DO 100 ITERATIONS
        MOV      #BIT0,$GDDAT    ;LOAD EXPECTED VALUE
        MOV      #1$,$LPERR     ;LOAD LOOP ADDRESS ON ERROR
1$:     MOV      #CLRALL,@SFR    ;RESET THE DEVICE
        MOV      $GDDAT,@OFF     ;LOAD OFFSET 'A'
        MOV      @OFF,$BDDAT    ;READ OFFSET
        CMP      $GDDAT,$BDDAT  ;COMPARE VALUES
        BEQ      2$              ;;BR IF SAME
        ERROR    6              ;OFFSET REG. IN ERROR
2$:     ASL      $GDDAT          ;CHANGE THE DATA
        BNE      1$              ;BR IF MORE DATA TO LOAD

```

```

1047
1048
1049
1050 005052 000004
1051 005054 012737 000010 001160
1052
1053 005062 012777 004000 174666
1054 005070 012777 011111 174652
1055 005076 012777 022222 174646
1056 005104 012777 033333 174642
1057 005112 012777 000036 174636
1058 005120 012777 007636 174620
1059
1060 005126 012737 011111 001124
1061 005134 017737 174610 001126
1062 005142 023737 001124 001126
1063 005150 001401
1064 005152 104007
1065 005154 012737 022222 001124 1$:
1066 005162 017737 174564 001126
1067 005170 023737 001124 001126
1068 005176 001401
1069 005200 104007
1070 005202 012737 033333 001124 2$:
1071 005210 017737 174540 001126
1072 005216 023737 001124 001126
1073 005224 001401
1074 005226 104007
1075 005230 012737 000036 001124 3$:
1076 005236 017737 174514 001126
1077 005244 023737 001124 001126
1078 005252 001401
1079 005254 104007
1080 005256 012737 007636 001124 4$:
1081 005264 017737 174456 001126
1082 005272 023737 001124 001126
1083 005300 001401
1084 005302 104007
1085 005304 012777 004000 174444 5$:
  
```

```

*****
*TEST 15 VERIFY NO DUAL REGISTER SELECTION
*****
TST15: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
;LOAD DIFFERENT NUMBERS INTO THE WCR, BAR AND OFF REGISTERS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #11111,@OFF ;LOAD OFFSET REGISTER
MOV #22222,@WCR ;LOAD W.C. REGISTER
MOV #33333,@BAR ;LOAD B.A. REGISTER
MOV #36,@SFR ;LOAD SPECIAL FUNCTION REGISTER
MOV #7636,@CSR ;LOAD COMMAND/STATUS REGISTER
;NOW READ EACH REGISTER AND CHECK THE VALUE
MOV #11111,$GDDAT ;LOAD EXPECTED VALUE
MOV @OFF,$BDDAT ;READ A OFFSET REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF SAME
ERROR 7 ;DUAL ADDRESS ERROR
MOV #22222,$GDDAT ;LOAD EXPECTED VALUE
MOV @WCR,$BDDAT ;READ W.C. REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;;BR IF SAME
ERROR 7 ;DUAL ADDRESS ERROR
MOV #33333,$GDDAT ;LOAD EXPECTED VLAUE
MOV @BAR,$BDDAT ;READ B.A. REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 3$ ;;BR IF SAME
ERROR 7 ;DUAL ADDRESS ERROR
MOV #36,$GDDAT ;LOAD EXPECTED VALUE
MOV @SFR,$BDDAT ;READ SPECIAL FUNCTION REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 4$ ;;BR IF SAME
ERROR 7 ;DUAL ADDRESS ERROR
MOV #7636,$GDDAT ;LOAD EXPECTED VALUE
MOV @CSR,$BDDAT ;READ COMMAND/STATUS REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 5$ ;;BR IF SAME
ERROR 7 ;DUAL ADDRESS ERROR
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
  
```

```

1086
1087
1088
1089 005312 000004
1090 005314 012737 000010 001160
1091 005322 005037 001124
1092 005326 012777 004000 174422
1093 005334 012777 000003 174406
1094 005342 012777 004000 174406
1095 005350 017737 174374 001126
1096 005356 001401
1097 005360 104006
1098
1099
1100
1101
1102 005362 000004
1103 005364 012777 004000 174364
1104 005372 052777 000010 174356
1105 005400 000240
1106 005402 000240
1107 005404 000240
1108 005406 052777 000022 174332
1109
1110 005414 012737 000022 001124
1111 005422 052777 000001 174316
1112 005430 017737 174312 001126
1113 005436 023737 001124 001126
1114 005444 001404
1115 005446 052777 000400 174302
1116 005454 104001
1117
1118 005456 052777 000400 174272
1119 005464 012737 000222 001124
1120 005472 017737 174250 001126
1121 005500 023737 001124 001126
1122 005506 001401
1123 005510 104001
1124
1125
1126 005512 012737 000200 001124
1127 005520 052777 000001 174220
1128 005526 052777 004000 174222
1129 005534 017737 174206 001126
1130 005542 023737 001124 001126
1131 005550 001401
1132 005552 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
;NOW SET 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 1 ;"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 1 ;"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

```

```

1133
1134
1135
1136 005554 000004
1137 005556 012737 000100 001160
1138 005564 012777 004000 174164
1139 005572 012777 000010 174156
1140 005600 012777 000000 174144
1141 005606 012777 000000 174140
1142 005614 012777 000022 174124
1143 005622 052777 000001 174116
1144 005630 052777 000002 174120
1145 005636 042777 000002 174112
1146 005644 012737 000001 001124
1147 005652 017737 174076 001126
1148 005660 001002
1149 005662 104010
1150 005664 000420
1151 005666 017737 174060 001126 1$:
1152 005674 001401
1153 005676 104011
1154 005700 012737 000020 001124 2$:
1155 005706 017737 174034 001126
1156 005714 032737 040000 001126
1157 005722 001401
1158 005724 104012
1159
1160
1161
1162 005726 000004
1163 005730 012737 000100 001160
1164 005736 012777 004000 174012
1165 005744 012777 000010 174004
1166 005752 012777 000000 173772
1167 005760 012777 000376 173766
1168 005766 012777 000022 173752
1169 005774 052777 000001 173744
1170 006002 052777 000002 173746
1171 006010 042777 000002 173740
1172 006016 012737 000400 001124
1173 006024 017737 173724 001126
1174 006032 105737 001127
1175 006036 001002
1176 006040 104010
1177 006042 000423
1178 006044 012737 000000 001124 1$:
1179 006052 017737 173674 001126
1180 006060 001401
1181 006062 104011
1182 006064 012737 000020 001124 2$:
1183 006072 017737 173650 001126
1184 006100 032737 040000 001126
1185 006106 001401
1186 006110 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 ;;
MOV @WCR,$BDDAT ;READ HIGH 16 BITS
BEQ 2$ ;;BR IF CLEARED
ERROR 11 ;HIGH 16 BIS CHANGED IN ERROR
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

*****
*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 ;;
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
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
    
```



```
1187
1188
1189
1190 006112 000004
1191 006114 012737 000100 001160
1192 006122 012777 004000 173626
1193 006130 012777 000010 173620
1194 006136 012777 000000 173606
1195 006144 012777 177776 173602
1196 006152 012777 000022 173566
1197 006160 052777 000001 173560
1198 006166 052777 000002 173562
1199 006174 042777 000002 173554
1200 006202 012737 000001 001124
1201 006210 017737 173536 001126
1202 006216 001002
1203 006220 104011
1204 006222 000413
1205 006224 012737 000020 001124
1206 006232 017737 173510 001126
1207 006240 032737 040000 001126
1208 006246 001401
1209 006250 104012
1210
1211
1212
1213 006252 000004
1214 006254 012737 000100 001160
1215 006262 012777 004000 173466
1216 006270 012777 000010 173460
1217 006276 012777 000377 173446
1218 006304 012777 177776 173442
1219 006312 012777 000022 173426
1220 006320 052777 000001 173420
1221 006326 052777 000002 173422
1222 006334 042777 000002 173414
1223 006342 012737 000400 001124
1224 006350 017737 173376 001126
1225 006356 105737 001127
1226 006362 001001
1227 006364 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
*****
*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 12 ;HIGH 8 BITS OF THE 32 BIT COUNTER FAILED TO CHANGE
```

```
1228
1229
1230
1231 006366 000004
1232 006370 012737 000100 001160
1233 006376 012777 004000 173352
1234 006404 012777 000000 173340
1235 006412 012777 000000 173334
1236 006420 005077 173322
1237 006424 052777 000002 173324
1238 006432 042777 000002 173316
1239 006440 012737 000000 001124
1240 006446 017737 173302 001126
1241 006454 001401
1242 006456 104010
1243
1244
1245
1246
1247 006460 000004
1248 006462 012737 000010 001160
1249 006470 012777 004000 173260
1250 006476 012777 000010 173252
1251 006504 012777 177777 173240
1252 006512 012777 177776 173234
1253 006520 012777 000022 173220
1254 006526 052777 000001 173212
1255
1256 006534 052777 000002 173214
1257 006542 042777 000002 173206
1258 006550 013700 002006
1259 006554 005001
1260 006556 012737 040020 001124
1261
1262 006564 032777 040000 173154 1$:
1263 006572 001011
1264 006574 005301
1265 006576 001372
1266 006600 005300
1267 006602 001370
1268 006604 012777 000400 173144
1269 006612 104012
1270 006614 000-16
1271
1272
1273 006616 052777 004000 173132 2$:
1274 006624 012737 000200 001124
1275 006632 017737 173110 001126
1276 006640 023737 001124 001126
1277 006646 001401
1278 006650 104012

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

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

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

```

1279
1280
1281
1282 006652 000004
1283 006654 012737 000010 001160
1284 006662 012777 004000 173066
1285 006670 012777 000010 173060
1286 006676 012777 177777 173046
1287 006704 012777 177776 173042
1288 006712 012777 000022 173026
1289 006720 052777 000001 173020
1290
1291 006726 052777 000002 173022
1292 006734 042777 000002 173014
1293
1294 006742 013700 002006
1295 006746 005001
1296 006750 012737 040020 001124
1297
1298 006756 032777 040000 172762 1$:
1299 006764 001014
1300 006766 005301
1301 006770 001372
1302 006772 005300
1303 006774 001370
1304 006776 012777 000400 172752
1305 007004 042777 000002 172744
1306 007012 104012
1307 007014 000416
1308
1309
1310 007016 052777 040000 172732 2$:
1311 007024 012737 000022 001124
1312 007032 017737 172710 001126
1313 007040 023737 001124 001126
1314 007046 001401
1315 007050 104012

```

```

*****
*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$: BIS #CLRWC0,@SFR ;CLEAR Z/WC FLOP
MOV #BIT4!BIT1,$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

```

```

1316
1317
1318
1319 007052 000004
1320 007054 012737 000010 001160
1321 007062 012777 004000 172666
1322 007070 012777 000010 172660
1323 007076 012777 177777 172646
1324 007104 012777 177776 172642
1325 007112 012777 000022 172626
1326 007120 052777 000001 172620
1327
1328 007126 052777 000002 172622
1329 007134 042777 000002 172614
1330 007142 013700 002006
1331 007146 005001
1332 007150 012737 040020 001124
1333
1334 007156 032777 040000 172562 1$:
1335 007164 001014
1336 007166 005301
1337 007170 001372
1338 007172 005300
1339 007174 001370
1340 007176 012777 000400 172552
1341 007204 042777 000002 172544
1342 007212 104012
1343 007214 000430
1344
1345
1346 007216 012746 000000
1347 007222 012746 007230
1348 007226 000002
1349 007230 012777 007274 172534 3$:
1350 007236 052777 000100 172502
1351 007244 000240
1352 007246 000240
1353 007250 000240
1354 007252 000240
1355 007254 017737 172466 001126
1356 007262 012777 004000 172466
1357 007270 104013
1358 007272 000401
1359
1360 007274 022626
1361 007276 005077 172444
1362 007302 012777 004000 172446
1363 007310 013777 001774 172454
1364 007316 005077 172452
1365

*****
*TEST 27 TEST THAT 'WCA OVFL' GENERATES AN INTERRUPT
*****
TST27: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #-1,@WCR ;LOAD Z COUNTER
MOV #-2,@BAR ;LOAD Z COUNTER
MOV #BIT4!BIT1,@CSR ;SET MATRIX MODE
BIS #BIT0,@CSR ;SET NCV11 ACTIVE
;NOW ENABLE 'TEST Z' INPUTS
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV CPUDLO,RO ;LOAD GROSS TIMER
CLR R1
MOV #BIT14!BIT4,$GDDAT ;LOAD EXPECTED STATUS
1$: BIT #BIT14,@CSR ;TEST FOR Z/WC OVERFLOW
BNE 2$ ;BR IF SET
DEC R1 ;DELAY
BNE 1$
DEC RO ;DELAY
BNE 1$
MOV #ENDDMA,@SFR ;STOP DMA TRANSFERS
BIC #TESTZ,@SFR ;STOP 'Z' INPUTS
ERROR 12 ;AFTER A GROSS TIME, THE Z/WC OVERFLOW FLOP FAILED TO SET
BR 5$ ;;BR TO CLEAN UP

;NOW ENABLE THE WC/Z OVERFLOW INTERRUPT BIT AND WAIT FOR AN INTERRUPT
2$: MOV #0,-(SP)
MOV #3$,-(SP) ;LSI-11 HACK
RTI
3$: MOV #4$,@VECTAO ;LOAD INTR. VECTOR
BIS #BIT6,@CSR ;ENABLE INTERRUPT
NOP
NOP
NOP
MOV @CSR,$BDDAT ;READ STATUS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
ERROR 13 ;WC/Z OVERFLOW FAILED TO GENERATE INTERRUPT
BR 5$ ;;BR TO CLEAN UP

4$: CMP (SP)+,(SP)+
5$: CLR @CSR
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV VECTA1,@VECTAO
CLR @VECTA1
    
```

```
1366  
1367  
1368  
1369 007322 000004  
1370 007324 012737 000010 001160  
1371 007332 012777 004000 172416  
1372 007340 012777 000014 172410  
1373 007346 012777 177777 172376  
1374 007354 012777 177776 172372  
1375 007362 005077 172362  
1376 007366 012777 000022 172352  
1377 007374 052777 000001 172344  
1378 007402 052777 000002 172346  
1379 007410 042777 000002 172340  
1380 007416 017737 172324 001126  
1381 007424 012737 040022 001124  
1382 007432 023737 001124 001126  
1383 007440 001402  
1384 007442 104012  
1385 007444 000407  
1386  
1387 007446 005037 001124 001126 1$: CLR $GDDAT ;CLEAR THE EXPECTED  
1388 007452 017737 172272 001126 MOV @OFF,$BDDAT ;READ THE ACTUAL  
1389 007460 001401 BEQ TST31 ;:BR IF CLEARED  
1390 007462 104012 ERROR 12 ;'WCA INC X OFF' IN MATRIX MODE CHANGED  
1391 ;THE OFFSET REGISTER
```

: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
CMP \$GDDAT,\$BDDAT ;COMPARE
BEQ 1\$;:BR IF SAME
ERROR 12 ;'WCA OVFL' FAILED TO CLEAR 'ACTIVE'
BR TST31 ;:

```
1392
1393
1394
1395 007464 000004
1396 007466 012737 000040 001160
1397 007474 012777 004000 172254
1398 007502 052777 000001 172246
1399 007510 013700 002010
1400 007514 105777 172236
1401 007520 100411
1402 007522 005300
1403 007524 001373
1404 007526 017737 172224 001126
1405 007534 012737 000200 001124
1406 007542 104014
1407
1408
1409 007544 005777 172212
1410 007550 017737 172202 001126
1411 007556 105737 001126
1412 007562 100003
1413 007564 005037 001124
1414 007570 104014
1415
1416
1417
1418 007572 000004
1419 007574 012737 000040 001160
1420 007602 012777 004000 172146
1421 007610 052777 000001 172140
1422 007616 013700 002010
1423 007622 105777 172130
1424 007626 100411
1425 007630 005300
1426 007632 001373
1427 007634 017737 172116 001126
1428 007642 012737 000200 001124
1429 007650 104014
1430
1431 007652 052777 004000 172076
1432 007660 017737 172072 001126
1433 007666 012737 000200 001124
1434 007674 023737 001124 001126
1435 007702 001401
1436 007704 104014
1437
1438 007706 000005
1439 007710 017737 172042 001126
1440 007716 012737 000000 001124
1441 007724 017737 172026 001126
1442 007732 001401
1443 007734 104014

*****
*TEST 31 VERIFY JOYSTICK DONE FLOP SETS
*****
TST31: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
BIS #REDJOY,@SFR ;REQUEST JOYSTICK DATA
MOV CPUDL1,R0 ;LOAD DELAY COUNTER
1$: TSTB @SFR ;WAIT FOR JOYSTICK READY
BMI 2$ ;BR IF SET
DEC R0 ;DELAY
BNE 1$ ;BR IF NOT EXHAUSTED
MOV @SFR,$BDDAT ;READ STATUS
MOV #BIT7,$GDDAT ;LOAD EXPECTED
ERROR 14 ;"JOYSTICK READY" FAILED TO SET
;NOW PERFORM A "TST" INSTRUCTION TO THE JOYSTICK REGISTER
; THE ACCESS OF THIS BUS ADDRESS SHOULD CLEAR JOY READY FLOP"
2$: TST @JOY ;ADDRESS THE REGISTER
MOV @SFR,$BDDAT ;READ STATUS
TSTB $BDDAT ;TEST IF THE BIT CLEARED
BPL TST32 ;;BR IF YES
CLR $GDDAT ;LOAD EXPECTED
ERROR 14 ;ADDRESSING THE JOYSTICK ADDRESS FAILED TO CLEAR
*****
*TEST 32 VERIFY THAT "RESET" INSTRUCTION CLEARS THE JOY READY FLOP
*****
TST32: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
BIS #REDJOY,@SFR ;REQUEST JOYSTICK DATA
MOV CPUDL1,R0 ;LOAD DELAY
1$: TSTB @SFR ;WAIT FOR JOYSTICK READY
BMI 2$ ;BR IF SET
DEC R0 ;DELAY
BNE 1$ ;BR IF NOT EXHAUSTED
MOV @SFR,$BDDAT ;READ STATUS
MOV #BIT7,$GDDAT ;LOAD EXPECTED VALUE
ERROR 14 ;JOY READY FAILED TO SET
;NOW ISSUE A "CLR ALL" AND VERIFY THE "JOY READY" DOES NOT CLEAR
2$: BIS #CLRALL,@SFR ;GENERATE "CLR ALL"
MOV @SFR,$BDDAT ;READ STATUS
MOV #BIT7,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 3$ ;;BR IF SAME
ERROR 14 ;"CLR ALL" CLEARED THE "JOY READY" FLOP IN ERROR
;NOW ISSUE A BUS "RESET" AND VERIFY THE JOY READY" CLEARS
3$: RESET ;BUS "INIT"
MOV @SFR,$BDDAT ;READ STATUS
MOV #0,$GDDAT ;LOAD EXPECTED
MOV @SFR,$BDDAT ;READ STATUS
BEQ TST33 ;;BR IF CLEARED
ERROR 14 ;BUS INIT FAILED TO CLEAR JOY READY
```

1444
1445
1446
1447
1448 007736 000004
1449 007740 012737 000010 001160
1450 007746 012777 004000 172002
1451 007754 012777 000014 171774
1452 007762 012777 000000 171756
1453 007770 052777 000001 171760
1454 007776 105777 171754
1455 010002 100375
1456 010004 017737 171752 001126
1457 010012 012737 000000 001124
1458 010020 023737 001124 001126
1459 010026 001401
1460 010030 104015
1461
1462
1463
1464
1465 010032 000004
1466 010034 012737 000010 001160
1467 010042 012777 004000 171706
1468 010050 012777 000014 171700
1469 010056 012777 002000 171662
1470 010064 052777 000001 171664
1471 010072 105777 171660
1472 010076 100375
1473 010100 017737 171656 001126
1474 010106 012737 124250 001124
1475 010114 023737 001124 001126
1476 010122 001401
1477 010124 104015
1478
1479
1480
1481
1482 010126 000004
1483 010130 012737 000010 001160
1484 010136 012777 004000 171612
1485 010144 012777 000014 171604
1486 010152 012777 004000 171566
1487 010160 052777 000001 171570
1488 010166 105777 171564
1489 010172 100375
1490 010174 017737 171562 001126
1491 010202 012737 050120 001124
1492 010210 023737 001124 001126
1493 010216 001401
1494 010220 104015

```
::*****  
:*TEST 33 JOYSTICK DATA PATH = GAIN =0 ZB ENABLE =0 RES. = 000  
:*****  
TST33: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #0,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #0,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST34 ;;BR IF SAME  
ERROR 15 ;JOYSTICK DATA PATH BIT SET
```

```
::*****  
:*TEST 34 JOYSTICK DATA PATH = GAIN =1  
:*****  
TST34: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #2000,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #124250,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST35 ;;BR IF SAME  
ERROR 15 ;JOYSTICK DATA PATH ERROR
```

```
::*****  
:*TEST 35 JOYSTICK DATA PATH = ZB ENABLE =1  
:*****  
TST35: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROLLER FLOP  
MOV #4000,@CSR ;LOAD CSR REGISTER  
BIS #REDJOY,@SFR ;REQUEST JOYSTICE DATA  
1$: TSTB @SFR ;WAIT FOR JOYSTICE READY  
BPL 1$  
MOV @JOY,$BDDAT ;READ THE REGISTER  
MOV #050120,$GDDAT ;LOAD EXPECTED  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST36 ;;BR IF SAME  
ERROR 15 ;JOYSTICK DATA PATH ERROR
```

```
1495  
1496  
1497  
1498 010222 000004  
1499 010224 012737 000010 001160  
1500 010232 012777 004000 171516  
1501 010240 012777 000014 171510  
1502 010246 012777 000023 171472  
1503 010254 052777 000001 171474  
1504 010262 105777 171470  
1505 010266 100375  
1506 010270 017737 171466 001126  
1507 010276 012737 000401 001124  
1508 010304 023737 001124 001126  
1509 010312 001401  
1510 010314 104015  
1511  
1512  
1513  
1514  
1515 010316 000004  
1516 010320 012737 000010 001160  
1517 010326 012777 004000 171422  
1518 010334 012777 000014 171414  
1519 010342 012777 000025 171376  
1520 010350 052777 000001 171400  
1521 010356 105777 171374  
1522 010362 100375  
1523 010364 017737 171372 001126  
1524 010372 012737 001002 001124  
1525 010400 023737 001124 001126  
1526 010406 001401  
1527 010410 104015  
1528  
1529  
1530  
1531  
1532 010412 000004  
1533 010414 012737 000010 001160  
1534 010422 012777 004000 171326  
1535 010430 012777 000014 171320  
1536 010436 012777 000031 171302  
1537 010444 052777 000001 171304  
1538 010452 105777 171300  
1539 010456 100375  
1540 010460 017737 171276 001126  
1541 010466 012737 002004 001124  
1542 010474 023737 001124 001126  
1543 010502 001401  
1544 010504 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
```



```

1597
1598
1599
1600 010772 000004
1601 010774 012737 000010 001160
1602 011002 012777 004000 170746
1603 011010 012777 000014 170740
1604 011016 012777 001023 170722
1605 011024 052777 000001 170724
1606 011032 105777 170720
1607 011036 100375
1608 011040 017737 170716 001126
1609 011046 012737 000000 001124
1610 011054 023737 001124 001126
1611 011062 001401
1612 011064 104017
1613
1614
1615
1616
1617
1618 011066 000004
1619 011070 012737 000010 001160
1620 011076 012777 004000 170652
1621 011104 012777 000014 170644
1622 011112 012777 007037 170626
1623 011120 052777 000001 170630
1624 011126 105777 170624
1625 011132 100375
1626 011134 017737 170622 001126
1627 011142 012737 177376 001124
1628 011150 023737 001124 001126
1629 011156 001401
1630 011160 104017
1631
1632
1633
1634
1635 011162 000004
1636 011164 012737 000010 001160
1637 011172 012777 004000 170556
1638 011200 012777 000014 170550
1639 011206 012777 001021 170532
1640 011214 052777 000001 170534
1641 011222 105777 170530
1642 011226 100375
1643 011230 017737 170526 001126
1644 011236 012737 000000 001124
1645 011244 023737 001124 001126
1646 011252 001401
1647 011254 104017
1648
    
```

```

*****
*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
    
```

```

*****
*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
    
```

```

*****
*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
    
```

```

1649
1650
1651
1652 011256 000004
1653 011260 012737 000010 001160
1654 011266 012777 004000 170462
1655 011274 012777 000014 170454
1656 011302 012777 000036 170436
1657 011310 052777 000002 170440
1658 011316 052777 000001 170422
1659 011324 042777 000002 170424
1660 011332 017737 170422 001126
1661 011340 012737 003407 001124
1662 011346 023737 001124 001126
1663 011354 001401
1664 011356 104020
1665
1666
1667
1668
1669 011360 000004
1670 011362 012737 000010 001160
1671 011370 012777 004000 170360
1672 011376 012777 000014 170352
1673 011404 012777 002036 170334
1674 011412 052777 000002 170336
1675 011420 052777 000001 170320
1676 011426 042777 000002 170322
1677 011434 017737 170320 001126
1678 011442 012737 127657 001124
1679 011450 023737 001124 001126
1680 011456 001401
1681 011460 104020
1682
1683
1684
1685
1686 011462 000004
1687 011464 012737 000010 001160
1688 011472 012777 004000 170256
1689 011500 012777 000014 170250
1690 011506 012777 004036 170232
1691 011514 052777 000002 170234
1692 011522 052777 000001 170216
1693 011530 042777 000002 170220
1694 011536 017737 170216 001126
1695 011544 012737 053527 001124
1696 011552 023737 001124 001126
1697 011560 001401
1698 011562 104020
1699
    
```

```

:*****
:*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
:*****
:*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
:*****
:*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
    
```

1700
1701
1702
1703 011564 000004
1704 011566 012737 000010 001160
1705 011574 012777 004000 170154
1706 011602 012777 000014 170146
1707 011610 012777 000034 170130
1708 011616 052777 000002 170132
1709 011624 052777 000001 170114
1710 011632 042777 000002 170116
1711 011640 017737 170114 001126
1712 011646 012737 001406 001124
1713 011654 023737 001124 001126
1714 011662 001401
1715 011664 104020
1716
1717
1718
1719
1720 011666 000004
1721 011670 012737 000010 001160
1722 011676 012777 004000 170052
1723 011704 012777 000014 170044
1724 011712 012777 002034 170026
1725 011720 052777 000002 170030
1726 011726 052777 000001 170012
1727 011734 042777 000002 170014
1728 011742 017737 170012 001126
1729 011750 012737 053656 001124
1730 011756 023737 001124 001126
1731 011764 001401
1732 011766 104020
1733
1734
1735
1736
1737 011770 000004
1738 011772 012737 000010 001160
1739 012000 012777 004000 167750
1740 012006 012777 000014 167742
1741 012014 012777 004034 167724
1742 012022 052777 000002 167726
1743 012030 052777 000001 167710
1744 012036 042777 000002 167712
1745 012044 017737 167710 001126
1746 012052 012737 125526 001124
1747 012060 023737 001124 001126
1748 012066 001401
1749 012070 104020
1750

```
*****  
*TEST 52 TEST ADDRESS MAKER - MATRIX MODE - RES = 6 GAIN = 0 ZB ENABLE = 0  
*****  
TST52: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #34,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #2034,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #4034,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BITO,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

```

1751
1752
1753
1754 012072 000004
1755 012074 012737 000010 001160
1756 012102 012777 004000 167646
1757 012110 012777 000014 167640
1758 012116 012777 000032 167622
1759 012124 052777 000002 167624
1760 012132 052777 000001 167606
1761 012140 042777 000002 167610
1762 012146 017737 167606 001126
1763 012154 012737 000402 001124
1764 012162 023737 001124 001126
1765 012170 001401
1766 012172 104020
1767
1768
1769
1770
1771 012174 000004
1772 012176 012737 000010 001160
1773 012204 012777 004000 167544
1774 012212 012777 000014 167536
1775 012220 012777 002032 167520
1776 012226 052777 000002 167522
1777 012234 052777 000001 167504
1778 012242 042777 000002 167506
1779 012250 017737 167504 001126
1780 012256 012737 025526 001124
1781 012264 023737 001124 001126
1782 012272 001401
1783 012274 104020
1784
1785
1786
1787
1788 012276 000004
1789 012300 012737 000010 001160
1790 012306 012777 004000 167442
1791 012314 012777 000014 167434
1792 012322 012777 004032 167416
1793 012330 052777 000002 167420
1794 012336 052777 000001 167402
1795 012344 042777 000002 167404
1796 012352 017737 167402 001126
1797 012360 012737 052452 001124
1798 012366 023737 001124 001126
1799 012374 001401
1800 012376 104020
1801

```

```

*****
*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
*****
*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
*****
*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

```

1802
1803
1804
1805 012400 000004
1806 012402 012737 000010 001160
1807 012410 012777 004000 167340
1808 012416 012777 000014 167332
1809 012424 012777 000030 167314
1810 012432 052777 000002 167316
1811 012440 052777 000001 167300
1812 012446 042777 000002 167302
1813 012454 017737 167300 001126
1814 012462 012737 000202 001124
1815 012470 023737 001124 001126
1816 012476 001401
1817 012500 104020
1818
1819
1820
1821
1822 012502 000004
1823 012504 012737 000010 001160
1824 012512 012777 004000 167236
1825 012520 012777 000014 167230
1826 012526 012777 002030 167212
1827 012534 052777 000002 167214
1828 012542 052777 000001 167176
1829 012550 042777 000002 167200
1830 012556 017737 167176 001126
1831 012564 012737 012726 001124
1832 012572 023737 001124 001126
1833 012600 001401
1834 012602 104020
1835
1836
1837
1838
1839 012604 000004
1840 012606 012737 000010 001160
1841 012614 012777 004000 167134
1842 012622 012777 000014 167126
1843 012630 012777 004030 167110
1844 012636 052777 000002 167112
1845 012644 052777 000001 167074
1846 012652 042777 000002 167076
1847 012660 017737 167074 001126
1848 012666 012737 025252 001124
1849 012674 023737 001124 001126
1850 012702 001401
1851 012704 104020
1852

```
::*****  
:*TEST 60 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 0  
:*****  
TST60: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #30,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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  
:*****  
:*TEST 61 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 1 ZB ENABLE = 0  
:*****  
TST61: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #2030,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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  
:*****  
:*TEST 62 TEST ADDRESS MAKER - MATRIX MODE - RES = 4 GAIN = 0 ZB ENABLE = 1  
:*****  
TST62: SCOPE  
MOV #10,$TIMES ;;DO 10 ITERATIONS  
MOV #CLRALL,@SFR ;CLEAR DEVICE  
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'  
MOV #4030,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE  
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
BIS #BIT0,@CSR ;SET ENABLE NCV11  
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
MOV @ADM,$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
```

```

1853
1854
1855
1856 012706 000004
1857 012710 012737 000010 001160
1858 012716 012777 004000 167032
1859 012724 012777 000014 167024
1860 012732 012777 000026 167006
1861 012740 052777 000002 167010
1862 012746 052777 000001 166772
1863 012754 042777 000002 166774
1864 012762 017737 166772 001126
1865 012770 012737 000000 001124
1866 012776 023737 001124 001126
1867 013004 001401
1868 013006 104020
1869
1870
1871
1872
1873 013010 000004
1874 013012 012737 000010 001160
1875 013020 012777 004000 166730
1876 013026 012777 000014 166722
1877 013034 012777 002026 166704
1878 013042 052777 000002 166706
1879 013050 052777 000001 166670
1880 013056 042777 000002 166672
1881 013064 017737 166670 001126
1882 013072 012737 005252 001124
1883 013100 023737 001124 001126
1884 013106 001401
1885 013110 104020
1886
1887
1888
1889
1890 013112 000004
1891 013114 012737 000010 001160
1892 013122 012777 004000 166626
1893 013130 012777 000014 166620
1894 013136 012777 004026 166602
1895 013144 052777 000002 166604
1896 013152 052777 000001 166566
1897 013160 042777 000002 166570
1898 013166 017737 166566 001126
1899 013174 012737 012424 001124
1900 013202 023737 001124 001126
1901 013210 001401
1902 013212 104020
1903

```

```

:*****
:*TEST 63 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 0
:*****
TST63: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #26,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$BDDAT ;READ THE ADDRESS DATA MAKER
MOV #0,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED TO READ
BEQ TST64 ;;BR IF SAME
ERROR 20 ;INCORRECT ADDRESS MAKER DATA
RESOLUTION = 3 GAIN = 0 ZB ENABLE = 0
:*****
:*TEST 64 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 1 ZB ENABLE = 0
:*****
TST64: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #2026,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$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
:*****
:*TEST 65 TEST ADDRESS MAKER - MATRIX MODE - RES = 3 GAIN = 0 ZB ENABLE = 1
:*****
TST65: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #TSTDMA!TSTCON,@SFR ;SET 'TEST DMA AND CONTROL'
MOV #4026,@CSR ;LOAD RESOLUTION, GAIN, ZB ENABLE VALUE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BITO,@CSR ;SET ENABLE NCV11
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
MOV @ADM,$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

```

1904
1905
1906
1907 013214 000004
1908 013216 012737 000010 001160
1909 013224 012777 004000 166524
1910 013232 012777 000014 166516
1911 013240 012777 000024 166500
1912 013246 052777 000002 166502
1913 013254 052777 000001 166464
1914 013262 042777 000002 166466
1915 013270 017737 166464 001126
1916 013276 012737 000000 001124
1917 013304 023737 001124 001126
1918 013312 001401
1919 013314 104020
1920
1921
1922
1923
1924 013316 000004
1925 013320 012737 000010 001160
1926 013326 012777 004000 166422
1927 013334 012777 000014 166414
1928 013342 012777 002024 166376
1929 013350 052777 000002 166400
1930 013356 052777 000001 166362
1931 013364 042777 000002 166364
1932 013372 017737 166362 001126
1933 013400 012737 002552 001124
1934 013406 023737 001124 001126
1935 013414 001401
1936 013416 104020
1937
1938
1939
1940
1941 013420 000004
1942 013422 012737 000010 001160
1943 013430 012777 004000 166320
1944 013436 012777 000014 166312
1945 013444 012777 004024 166274
1946 013452 052777 000002 166276
1947 013460 052777 000001 166260
1948 013466 042777 000002 166262
1949 013474 017737 166260 001126
1950 013502 012737 005224 001124
1951 013510 023737 001124 001126
1952 013516 001401
1953 013520 104020
1954

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


1955
1956
1957
1958 013522 000004
1959 013524 012737 000010 001160
1960 013532 012777 004000 166216
1961 013540 012777 000014 166210
1962 013546 012777 000022 166172
1963 013554 052777 000002 166174
1964 013562 052777 000001 166156
1965 013570 042777 000002 166160
1966 013576 017737 166156 001126
1967 013604 012737 000000 001124
1968 013612 023737 001124 001126
1969 013620 001401
1970 013622 104020
1971
1972
1973
1974
1975 013624 000004
1976 013626 012737 000010 001160
1977 013634 012777 004000 166114
1978 013642 012777 000014 166106
1979 013650 012777 002022 166070
1980 013656 052777 000002 166072
1981 013664 052777 000001 166054
1982 013672 042777 000002 166056
1983 013700 017737 166054 001126
1984 013706 012737 001265 001124
1985 013714 023737 001124 001126
1986 013722 001401
1987 013724 104020
1988
1989
1990
1991
1992 013726 000004
1993 013730 012737 000010 001160
1994 013736 012777 004000 166012
1995 013744 012777 000014 166004
1996 013752 012777 004022 165766
1997 013760 052777 000002 165770
1998 013766 052777 000001 165752
1999 013774 042777 000002 165754
2000 014002 017737 165752 001126
2001 014010 012737 002512 001124
2002 014016 023737 001124 001126
2003 014024 001401
2004 014026 104020
2005

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

```

2006
2007
2008
2009 014030 000004
2010 014032 012737 000010 001160
2011 014040 012777 004000 165710
2012 014046 012777 000014 165702
2013 014054 012777 000000 165664
2014 014062 052777 000002 165666
2015 014070 052777 000001 165650
2016 014076 042777 000002 165652
2017 014104 017737 165650 001126
2018 014112 012737 003407 001124
2019 014120 023737 001124 001124
2020 014126 001401
2021 014130 104021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
  
```

```

*****
*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
  
```

```

2059
2060
2061
2062 014336 000004
2063 014340 012737 000040 001160
2064 014346 012777 004000 165402
2065 014354 012737 125252 060000
2066 014362 012777 011110 165360
2067 014370 012777 177777 165354
2068 014376 012777 060000 165350
2069 014404 012777 000014 165344
2070 014412 052777 000001 165326
2071 014420 052777 000002 165330
2072 014426 042777 000002 165322
2073 014434 000240
2074 014436 000240
2075 014440 000240
2076 014442 052777 010000 165306
2077 014450 000240
2078 014452 000240
2079 014454 000240
2080 014456 017737 165264 001126
2081 014464 012777 004000 165264
2082 014472 012737 040200 001124
2083 014500 023737 001124 001126
2084 014506 001402
2085 014510 104036
2086 014512 000453
2087 014514 017737 165234 001126 4$:
2088 014522 012737 060002 001124
2089 014530 023737 001124 001126
2090 014536 001401
2091 014540 104022
2092
2093
2094 014542 017737 165204 001126 1$:
2095 014550 012737 000000 001124
2096 014556 023737 001124 001126
2097 014564 001401
2098 014566 104023
2099
2100 014570 005037 001124 2$:
2101 014574 017737 165150 001126
2102 014602 001401
2103 014604 104024
2104
2105 014606 013737 060000 001126 3$:
2106 014614 012737 003407 001124
2107 014622 012737 060000 001122
2108 014630 023737 001124 001126
2109 014636 001401
2110 014640 104037

```

```

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

```

```

2111      ::*****
2112      ::*TEST 100      ENABLE A 512 WORD TRANSFER      SECTION LIST MODE
2113      ::*****
2114      TST100: SCOPE
2115      014642 000004      MOV      #40,$TIMES      ;;DO 40 ITERATIONS
2116      014644 012737 000040 001160      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
2117      014652 012777 004000 165076      MOV      #BUF0,R0      ;LOAD BUFFER POINTER
2118      014660 012700 060000      MOV      #125252,(R0)+      ;PRESET THE BUFFER WITH DATA
2119      014664 012720 125252      1$:      MOV      R0,#BUF1      ;TEST IF DONE
2120      014670 020027 062000      CMP      R0,#BUF1      ;BR IF NOT
2121      014674 001373      BNE      1$
2122      014676 012777 177000 165046      MOV      #-512,@WCR      ;SET UP 512. WORD TRANSFER
2123      014704 012777 060000 165042      MOV      #BUF0,@BAR      ;LOAD BUS ADDRESS FOR RESULT
2124      014712 012777 000014 165036      MOV      #TSTDMA!TSTCON,@SFR      ;ENABLE TEST CONTROL AND DMA FLOPS
2125      014720 052777 000001 165020      BIS      #BIT0,@CSR      ;ENABLE DEVICE
2126      014726 012737 001000 002004      MOV      #512,$TEMP      ;LOAD THE COUNTER
2127      014734 052777 000002 165014      2$:      BIS      #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
2128      014742 042777 000002 165006      BIC      #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
2129      014750 052777 010000 165000      BIS      #BIT12,@SFR      ;ALLOW 1 DMA TRANSFER
2130      014756 005337 002004      DEC      $TEMP      ;FINISHED ALL WORDS?
2131      014762 001364      BNE      2$      ;BR UNTILL DONE
2132      ;THE TRANSFER IS NOW COMPLETE
2133      014764 017737 164756 001126      MOV      @CSR,$BDDAT      ;READ STATUS
2134      014772 012737 040200 001124      MOV      #40200,$GDDAT      ;LOAD EXPECTED STATUS
2135      015000 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA
2136      015006 001402      BEQ      3$      ;;BR IF EXPECTED STATUS
2137      015010 104036      ERROR    36      ;UNEXPECTED STATUS AFTER 512 WORD TRANSFER
2138      015012 000465      BR      TST101      ;;
2139      015014 005037 001124      3$:      CLR      $GDDAT      ;CLEAR EXPECTED
2140      015020 017737 164724 001126      MOV      @OFF,$BDDAT      ;READ OFFSET REG.
2141      015026 001401      BEQ      4$      ;;BR IF CLEARED
2142      015030 104006      ERROR    6      ;UNEXPECTED OFFSET REGISTER BIT SET
2143      015032 012777 004000 164716      4$:      MOV      #CLRALL,@SFR      ;CLEAR THE DEVICE
2144      015040 017737 164710 001126      MOV      @BAR,$BDDAT      ;READ BUS ADDRESS
2145      015046 012737 062000 001124      MOV      #BUF1,$GDDAT      ;LOAD EXPECTED BAR VALUE
2146      015054 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
2147      015062 001401      BEQ      5$      ;;BR IF SAME
2148      015064 104022      ERROR    22      ;INCORRECT BUS ADDRESS VALUE AFTER A 1 WORD TRANSFER
2149
2150      015066 017737 164660 001126      5$:      MOV      @WCR,$BDDAT      ;READ W.C. REGISTER
2151      015074 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED W.C. VALUE
2152      015102 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE VALUES
2153      015110 001401      BEQ      6$      ;;BR IF SAME
2154      015112 104023      ERROR    23      ;INCORRECT WORD COUNT REGISTER VALUE AFTER A 1 WORD TRANSFER
2155      015114 012737 003407 001124      6$:      MOV      #3407,$GDDAT      ;LOAD EXPECTED DATA
2156      015122 012737 060000 001122      MOV      #BUF0,$BDADR      ;LOAD STARTING ADDRESS
2157      015130 017737 163766 001126      7$:      MOV      @$BDADR,$BDDAT      ;READ DATA WORD
2158      015136 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE DATA
2159      015144 001401      BEQ      10$      ;;BR IF EXPECTED
2160      015146 104037      ERROR    37      ;INCORRECT DATA IN LIST MODE XFER.
2161      015150 062737 000002 001122      10$:      ADD      #2,$BDADR      ;UPDATE POINTER
2162      015156 022737 062000 001122      CMP      #BUF0+1024,$BDADR      ;TEST IF END OF BUFFER
2163      015164 001361      BNE      7$      ;;BR IF NOT DONE

```

```

2164
2165
2166
2167 015166 000004
2168 015170 012737 000040 001160
2169 015176 012777 004000 164552
2170 015204 012777 177777 164540
2171 015212 012777 000003 164530
2172 015220 012777 160000 164526
2173 015226 012777 000014 164522
2174 015234 012777 000016 164504
2175 015242 052777 000001 164476
2176 015250 052777 000002 164500
2177 015256 042777 000002 164472
2178 015264 000240
2179 015266 000240
2180 015270 000240
2181 015272 052777 010000 164456
2182 015300 000240
2183 015302 000240
2184 015304 000240
2185 015306 012737 140200 001124
2186 015314 017737 164426 001126
2187 015322 100402
2188 015324 104025
2189 015326 000423
2190 015330 023737 001124 001126 1$:
2191 015336 001401
2192 015340 104025
2193 015342 052777 004000 164406 2$:
2194 015350 012737 000200 001124
2195 015356 017737 164364 001126
2196 015364 023737 001124 001126
2197 015372 001401
2198 015374 104025
2199
2200
2201
2202 015376 000004
2203 015400 012737 000040 001160
2204 015406 012777 004000 164342
2205 015414 012777 177777 164330
2206 015422 012777 000003 164320
2207 015430 012777 160000 164316
2208 015436 012777 000014 164312
2209 015444 052777 000001 164274
2210 015452 052777 000002 164276
2211 015460 042777 000002 164270
2212 015466 000240
2213 015470 000240
2214 015472 000240
2215 015474 052777 010000 164254
2216 015502 000240
2217 015504 000240

```

```

*****
*TEST 101 VERIFY 'TIMEOUT' FLOP SETS AND 'CLR ALL' CLEARS IT
*****
TST101: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #CLRALL,@SFR ;CLEAR DEVICE
MOV #-1,@WCR ;LOAD W.C. REGISTER
MOV #3,@OFF ;LOAD EXTENDED ADDRESS BITS
MOV #160000,@BAR ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDR
MOV #TSTCON!TSTDMA,@SFR ;SET TEST CONTROL AND DMA
MOV #16,@CSR ;LOAD RESOLUTION TO VERIFY 'SFR INIT' CLEARS
BIS #BIT0,@CSR ;ENABLE THE DEVICE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
NOP
NOP
NOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP
MOV #BIT15!BIT14!BIT7,$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 ;;
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 2$ ;;BR IF SAME
ERROR 25 ;'TIMEOUT' FLOP DID SET BUT FAILED TO GENERATE ''
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
NOP
BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER
NOP
NOP

```

J 5
PAGE 49

CZNCCA NCV11 DIAGNOSTIC MACY11 27(654) 8-AUG-78 15:05
CZNCCA.P11 T102 VERIFY 'TIMEOUT' FLOP SETS AND 'CLR TIMEOUT' CLEARS IT SEQ 0061

```

2218 015506 000240      NOP
2219 015510 012737 140200 001124  MOV    #BIT15:BIT14!BIT7,$GDDAT      ;LOAD EXPECTED
2220 015516 017737 164224 001126  MOV    @CSR,$BDDAT                    ;READ STATUS REG.
2221 015524 100402      BMI    1$                             ;BR IF 'TIMEOUT' FLOP IS SET
2222 015526 104025      ERROR  25                             ;'TIMEOUT' FLOP FAILED TO SET
2223 015530 000423      BR     TST103                          ;;
2224 015532 023737 001124 001126 1$:  CMP    $GDDAT,$BDDAT                  ;COMPARE VALUES
2225 015540 001401      BEQ    2$                             ;;BR IF SAME
2226 015542 104025      ERROR  25                             ;'TIMEOUT' FLOP DID SET BUT FAILED TO GENERATE ''
2227 015544 052777 100000 164204 2$:  BIS    #BIT15,@SFR                    ;GENERATE AN 'CLR TIMEOUT' TO CLEAR TIMEOUT FLOP
2228 015552 012737 040200 001124  MOV    #BIT14!BIT7,$GDDAT            ;LOAD EXPECTED
2229 015560 017737 164162 001126  MOV    @CSR,$BDDAT                    ;READ STATUS
2230 015566 023737 001124 001126  CMP    $GDDAT,$BDDAT                  ;COMPARE VALUE
2231 015574 001401      BEQ    TST103                          ;;BR IF SAME
2232 015576 104025      ERROR  25                             ;'CLR TIMEOUT' FAILED TO CLEAR TIMEOUT FLOP
2233
2234
2235
*****
;*TEST 103  VERIFY 'TIMEOUT' INTERRUPT
*****
TST103: SCOPE
2236 015600 000004      MOV    #40,$TIMES                      ;;DO 40 ITERATIONS
2237 015602 012737 000040 001160  MOV    #CLRALL,@SFR                    ;CLEAR DEVICE
2238 015610 012777 004000 164140  MOV    #-1,@WCR                         ;LOAD W.C. REGISTER
2239 015616 012777 177777 164126  MOV    #3,@OFF                          ;LOAD EXTENDED ADDRESS BITS
2240 015624 012777 000003 164116  MOV    #160000,@BAR                     ;LOAD BUS ADDRESS REGISTER TO A NON-EXISTENT ADDRE
2241 015632 012777 160000 164114  MOV    #TSTCON!TSTDMA,@SFR              ;SET TEST CONTROL AND DMA
2242 015640 012777 000014 164110  MOV    #0,-(SP)
2243 015646 012746 000000  MOV    #1$,-(SP)
2244 015652 012746 015660  RTI
2245 015656 000002      MOV    #2$,@VECTA0
2246 015660 012777 015752 164104 1$:  NOP
2247 015666 000240      NOP
2248 015670 000240      NOP
2249 015672 000240      NOP
2250 015674 000240      NOP
2251 015676 052777 000101 164042  BIS    #BIT6!BIT0,@CSR                  ;ENABLE THE DEVICE
2252 015704 052777 000002 164044  BIS    #TESTZ,@SFR                      ;ENABLE 'TEST Z' PULSES
2253 015712 042777 000002 164036  BIC    #TESTZ,@SFR                      ;DISABLE 'TEST Z' PULSES
2254 015720 000240      NOP
2255 015722 000240      NOP
2256 015724 000240      NOP
2257 015726 052777 010000 164022  BIS    #BIT12,@SFR                      ;ALLOW 1 DMA TRANSFER
2258 015734 000240      NOP
2259 015736 000240      NOP
2260 015740 000240      NOP
2261 015742 005077 164000  CLR    @CSR                             ;CLEAR ENABLE
2262 015746 104026  ERROR  26                             ;'TIMEOUT' FAILED TO INTERRUPT
2263 015750 000401      BR     3$                             ;;BR TO CLEAN UP
2264
2265 015752 022626      CMP    (SP)+,(SP)+                      ;CLEAN THE STACK
2266 015754 005077 163766 3$:  CLR    @CSR
2267 015760 012777 004000 163770  MOV    #CLRALL,@SFR                      ;CLEAR THE DEVICE
2268 015766 013777 001774 163776  MOV    VECTA1,@VECTA0                   ;RESET VECTOR
2269 015774 005077 163774  CLR    @VECTA1

```

```

2270
2271
2272
2273 016000 000004
2274 016002 012737 000040 001160
2275 016010 012777 004000 163740
2276 016016 023727 036402 002140
2277 016024 103450
2278 016026 012777 177777 163716
2279 016034 012777 177776 163712
2280 016042 005077 163702
2281 016046 012777 000014 163702
2282 016054 012777 000016 163664
2283 016062 052777 000001 163656
2284 016070 052777 000002 163660
2285 016076 042777 000002 163652
2286 016104 052777 010000 163644
2287 016112 000240
2288 016114 000240
2289 016116 000240
2290 016120 012737 000001 001124
2291 016126 017737 163616 001126
2292 016134 023737 001124 001126
2293 016142 001401
2294 016144 104006
2295
2296
2297
2298
2299 016146 000004
2300 016150 012737 000040 001160
2301 016156 012777 004000 163572
2302 016164 023727 036402 006140
2303 016172 103454
2304 016174 012777 177777 163550
2305 016202 012777 177776 163544
2306 016210 012777 000001 163532
2307 016216 012777 000014 163532
2308 016224 012777 000016 163514
2309 016232 052777 000001 163506
2310 016240 052777 000002 163510
2311 016246 042777 000002 163502
2312 016254 000240
2313 016256 000240
2314 016260 000240
2315 016262 052777 010000 163466
2316 016270 000240
2317 016272 000240
2318 016274 000240
2319 016276 012737 000002 001124
2320 016304 017737 163440 001126
2321 016312 023737 001124 001126
2322 016320 001401
2323 016322 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    6               ;EXTENDED ADDRESS BIT 16 FAILED TO SET
    
```

```

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

2324
2325
2326
2327 016324 000004
2328 016326 012737 000040 001160
2329 016334 012777 004000 163414
2330 016342 012777 177777 163402
2331 016350 012777 060000 163376
2332 016356 012777 000014 163372
2333 016364 012737 007070 060000
2334 016372 052777 000001 163346
2335 016400 000240
2336 016402 000240
2337 016404 000240
2338 016406 052777 002000 163342
2339 016414 052777 010000 163334
2340 016422 000240
2341 016424 000240
2342 016426 000240
2343 016430 013737 060000 001126
2344 016436 012737 177777 001124
2345 016444 023737 001124 001126
2346 016452 001401
2347 016454 104027
2348
2349
2350
2351
2352 016456 000004
2353 016460 012737 000040 001160
2354 016466 012777 004000 163262
2355 016474 012777 177777 163250
2356 016502 012777 060000 163244
2357 016510 012777 000014 163240
2358 016516 012737 007070 060000
2359 016524 052777 000001 163214
2360 016532 000240
2361 016534 000240
2362 016536 000240
2363 016540 052777 001000 163210
2364 016546 052777 010000 163202
2365 016554 000240
2366 016556 000240
2367 016560 000240
2368 016562 013737 060000 001126
2369 016570 012737 000000 001124
2370 016576 023737 001124 001126
2371 016604 001401
2372 016606 104027
2373

```

*****
*TEST 106      VERIFY 'SET EVENT' DATA GENERATES A 177777 DATA WORD
*****
TST106: SCOPE
MOV      #40,$TIMES      ;;DO 40 ITERATIONS
MOV      #CLRALL,@SFR    ;CLEAR THE DEVICE
MOV      #-1,@WCR       ;SET UP 1 WORD TRANSFER
MOV      #BUFO,@BAR     ;LOAD BUS ADDRESS FOR RESULT
MOV      #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV      #7070,BUFO     ;PRESET DATA
BIS      #BIT0,@CSR     ;ENABLE DEVICE
NOP
NOP
NOP
BIS      #BIT10,@SFR    ;SET 'EVENT' FLOP
BIS      #BIT12,@SFR    ;ALLOW 1 DMA TRANSFER
NOP
NOP
NOP
MOV      BUFO,$BDDAT    ;READ DATA
MOV      #-1,$GDDAT    ;LOAD EXPECTED
CMP      $GDDAT,$BDDAT  ;COMPARE VALUES
BEQ      TST107        ;;BR IF SAME
ERROR    27            ;INCORRECT DATA VALUE FOR 'EVENT' MARK
                        ;AFTER A 1 WORD TRANSFER

*****
*TEST 107      VERIFY 'SET TIME' DATA GENERATES A 000000 DATA WORD
*****
TST107: SCOPE
MOV      #40,$TIMES      ;;DO 40 ITERATIONS
MOV      #CLRALL,@SFR    ;CLEAR THE DEVICE
MOV      #-1,@WCR       ;SET UP 1 WORD TRANSFER
MOV      #BUFO,@BAR     ;LOAD BUS ADDRESS FOR RESULT
MOV      #TSTDMA!TSTCON,@SFR ;ENABLE TEST CONTROL AND DMA FLOPS
MOV      #7070,BUFO     ;PRESET THE DATA
BIS      #BIT0,@CSR     ;ENABLE DEVICE
NOP
NOP
NOP
BIS      #BIT9,@SFR     ;SET 'TIME' FLOP
BIS      #BIT12,@SFR    ;ALLOW 1 DMA TRANSFER
NOP
NOP
NOP
MOV      BUFO,$BDDAT    ;READ DATA
MOV      #0,$GDDAT     ;LOAD EXPECTED
CMP      $GDDAT,$BDDAT  ;COMPARE VALUES
BEQ      TST110        ;;BR IF SAME
ERROR    27            ;INCORRECT DATA VALUE FOR 'TIME' MARK
                        ;AFTER A 1 WORD TRANSFER

```



```

2374
2375
2376
2377 016610 000004
2378 016612 012737 000040 001160
2379 016620 012777 004000 163130
2380 016626 005737 004016
2381 016632 001044
2382 016634 012777 177777 163110
2383 016642 012777 060000 163104
2384 016650 012777 000014 163100
2385 016656 012737 007070 060000
2386 016664 052777 000001 163054
2387 016672 000240
2388 016674 000240
2389 016676 052777 000400 163034
2390 016704 052777 010000 163044
2391 016712 000240
2392 016714 000240
2393 016716 013737 060000 001126
2394 016724 012737 177777 001124
2395 016732 023737 001124 001126
2396 016740 001401
2397 016742 104040
2398
2399
2400
2401
2402 016744 000004
2403 016746 012737 000040 001160
2404 016754 012777 004000 162774
2405 016762 005737 004016
2406 016766 001052
2407 016770 012777 177777 162754
2408 016776 012777 060000 162750
2409 017004 012777 000014 162744
2410 017012 012737 007070 060000
2411 017020 052777 000001 162720
2412 017026 005077 162706
2413 017032 012777 177776 162704
2414 017040 012777 000011 162672
2415 017046 105777 162666 1$:
2416 017052 100375
2417 017054 052777 010000 162674
2418 017062 000240
2419 017064 000240
2420 017066 013737 060000 001126
2421 017074 012737 000000 001124
2422 017102 023737 001124 001126
2423 017110 001401
2424 017112 104040
2425

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

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

```

2426
2427
2428
2429 017114 000004
2430 017116 012737 000040 001160
2431 017124 012777 004000 162624
2432 017132 012777 177777 162612
2433 017140 012777 060000 162606
2434 017146 012777 000014 162602
2435 017154 012737 007070 060000
2436 017162 052777 000001 162556
2437 017170 000240
2438 017172 000240
2439 017174 000240
2440 017176 052777 003000 162552
2441 017204 052777 010000 162544
2442 017212 000240
2443 017214 000240
2444 017216 000240
2445 017220 013737 060000 001126
2446 017226 012737 000000 001124
2447 017234 023737 001124 001126
2448 017242 001401
2449 017244 104027
2450
2451
2452
2453 017246 000004
2454 017250 012737 000040 001160
2455 017256 012777 004000 162472
2456 017264 005037 060000
2457 017270 012777 060000 162452
2458 017276 012777 000022 162442
2459 017304 012777 000014 162444
2460 017312 052777 000001 162426
2461 017320 052777 000002 162430
2462 017326 042777 000002 162422
2463 017334 000240
2464 017336 000240
2465 017340 000240
2466 017342 052777 010000 162406
2467 017350 000240
2468 017352 000240
2469 017354 000240
2470 017356 013737 060000 001126
2471 017364 012737 000001 001124
2472 017372 023737 001124 001126
2473 017400 001401
2474 017402 104030
2475
2476

```

```

*****
*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 PROBLY WRONG
;IF DATA WAS NON ZERO, THE 'INCR' REGISTER IS STUCK TO A 1

```

```

2477
2478
2479
2480 017404 000004
2481 017406 012737 000040 001160
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491 017414 012737 017430 001110
2492 017422 012737 000002 002004
2493
2494 017430 012777 004000 162320 1$:
2495 017436 012777 060000 162304
2496 017444 012777 000024 162274
2497 017452 012777 000014 162276
2498 017460 052777 000001 162260
2499 017466 013737 002004 060000
2500 017474 005337 060000
2501 017500 052777 000002 162250
2502 017506 042777 000002 162242
2503 017514 052777 010000 162234
2504 017522 013737 002004 001124
2505 017530 013737 060000 001126
2506 017536 023737 001124 001126
2507 017544 001401
2508 017546 104030
2509 017550 017737 162172 001126 2$:
2510 017556 032737 020000 001126
2511 017564 001407
2512 017566 012737 000024 001124
2513 017574 104031
2514 017576 052777 020000 162152
2515 017604 006337 002004 3$:
2516 017610 001307

```

```

*****
*TEST 114 VERIFY EACH BIT OF THE INCREMENT REG. DATA PATH
*****
TST114: SCOPE
MOV #40,$TIMES ;:DO 40 ITERATIONS
;CHECK FOR A "CARRY" INTO EACH BIT
IE:
000002-000001
000004-000003
000010-000007
000020-000017
000040-000037
000100-000077
000200-000177
ETC
MOV #1,$, $LPERR ;:LOAD RETURN ADDRESS IF ERROR
MOV #2,$, $TEMP ;:LOAD INITIAL VALUE
;CLEAR THE DEVICE
;LOAD INITIAL OFFSET REGISTER
;ENABLE WORD MATRIX MODE
;SET TEST DMA AND CONTROL
;ENABLE THE DEVICE
;LOAD PRESET VALUE
;TO EXPECTED -1
;ENABLE "TEST Z" PULSES
;DISABLE "TEST Z" PULSES
;ALLOW 1 TRANSFER
;LOAD TYPEOUT EXPECTED
;READ THE BUFO LOCATION
;COMPARE VALUES
;:BR IF SAME
;INCORRECT DATA IN THE INCREMENT REGISTER
;READ STATUS
;TEST FOR UNEXPECTED CELL OVERFLOW
;BR IF NOT
;LOAD THE EXPECTED TYPEOUT
;UNEXPECTED "CELL OVERFLOW" STATUS
;TRY TO CLEAR THE FLAG
;CHANGE THE EXPCTED
;BR IF MORE DATA BITS

```

2517
 2518
 2519
 2520 017612 000004
 2521 017614 012737 000040 001160
 2522 017622 012777 004000 162126
 2523 017630 005077 162116
 2524 017634 005077 162114
 2525 017640 012777 060000 162102
 2526 017646 012777 000022 162072
 2527 017654 012777 000014 162074
 2528 017662 052777 000001 162056
 2529 017670 052777 000002 162060
 2530 017676 042777 000002 162052
 2531 017704 000240
 2532 017706 000240
 2533 017710 000240
 2534 017712 112737 000377 060000
 2535 017720 112737 000200 060001
 2536 017726 052777 010000 162022
 2537 017734 000240
 2538 017736 000240
 2539 017740 000240
 2540 017742 017737 162000 001126
 2541 017750 012737 020022 001124
 2542 017756 023737 001124 001126
 2543 017764 001401
 2544 017766 104031
 2545
 2546
 2547
 2548
 2549 017770 052777 020000 161760
 2550 017776 017737 161744 001126
 2551 020004 012737 000022 001124
 2552 020012 023737 001124 001126
 2553 020020 001401
 2554 020022 104031
 2555
 2556
 2557 020024 013737 060000 001126
 2558 020032 012737 100377 001124
 2559 020040 023737 001124 001126
 2560 020046 001401
 2561 020050 104031
 2562
 2563

```

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

;NOW GENERATE 'CLR CELL' TO CLEAR CELL OVERFLOW FLOP
1$: BIS #BIT13,@SFR ;GENERATE 'CLR CELL'
MOV @CSR,$BDDAT ;READ STATUS
MOV #BIT4!BIT1,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ 2$ ;;BR IF SAME
ERROR 31 ;'CLR CELL' FAILED TO CLEAR 'CELL OVFL' FLOP

;NOW VERIFY THE BYTE DATA CELL
2$: MOV BUFO,$BDDAT ;READ DATA
MOV #100377,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST116 ;;BR IF SAME
ERROR 31 ;OVERFLOW FROM INC. REG. FAILED TO INHIBIT
;'INC CNT' FROM CHANGING THE DATA OR
;'CSR BYTE CELLS' FAILED TO INHIBIT 'INC ENA HI'

```

2564
2565
2566
2567 020052 000004
2568 020054 012737 000040 001160
2569 020062 012777 004000 161666
2570 020070 005077 161660
2571 020074 005077 161652
2572 020100 012777 060000 161642
2573 020106 012777 000024 161632
2574 020114 012777 000014 161634
2575 020122 052777 000001 161616
2576 020130 052777 000002 161620
2577 020136 052777 000002 161612
2578 020144 000240
2579 020146 000240
2580 020150 000240
2581 020152 012737 177777 060000
2582 020160 052777 010000 161570
2583 020166 000240
2584 020170 000240
2585 020172 000240
2586 020174 017737 161546 001126
2587 020202 012737 020024 001124
2588 020210 023737 001124 001126
2589 020216 001401
2590 020220 104031
2591
2592 020222 052777 004000 161526
2593 020230 017737 161512 001126
2594 020236 012737 000200 001124
2595 020244 023737 001124 001126
2596 020252 001401
2597 020254 104031
2598
2599 020256 013737 060000 001126
2600 020264 012737 177777 001124
2601 020272 023737 001124 001126
2602 020300 001401
2603 020302 104031

```

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

```

```
2604 ::*****
2605 :*TEST 117 CHECK FOR "CELL OVERFLOW" INTERRUPT
2606 :*****
2607 TST117: SCOPE
2608 020304 000004 MOV #40,$TIMES ;;DO 40 ITERATIONS
2609 020306 012737 000040 001160 MOV #CLRALL,@SFR ;CLEAR THE DEVICE
2610 020314 012777 004000 161434 CLR @BAR
2611 020322 005077 161426 CLR @WCR
2612 020332 012777 060000 161410 MOV #BUFO,@OFF ;LOAD INITIAL OFFSET REGISTER
2613 020340 012777 000064 161400 MOV #BIT5!BIT4!BIT2,@CSR ;ENABLE INTR. AND MATRIX MODE
2614 020346 012777 000014 161402 MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
2615 020354 052777 000001 161364 BIS #BIT0,@CSR ;ENABLE THE DEVICE
2616 020362 052777 000002 161366 BIS #TESTZ,@SFR ;ENABLE "TEST Z" PULSES
2617 020370 042777 000002 161360 BIC #TESTZ,@SFR ;DISABLE "TEST Z" PULSES
2618 020376 000240 NOP
2619 020400 000240 NOP
2620 020402 000240 NOP
2621 020404 012737 177777 060000 MOV #-1,BUFO ;SET BUFO LOC. TO WORD -1
2622 020412 012746 000000 MOV #0,-(SP)
2623 020416 012746 020424 MOV #1$,-(SP)
2624 020422 000002 RTI
2625 020424 012777 020452 161344 1$: MOV #2$,@VECTB0 ;LOAD RETURN VECTOR
2626 020432 052777 010000 161316 BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
2627 020440 000240 NOP
2628 020442 000240 NOP
2629 020444 000240 NOP
2630 020446 104032 ERROR 32 ;"CELL OVERFLOW" FAILED TO CAUES AN INTERRUPT
2631 020450 000414 BR 3$ ;;BR TO CLEAN UP
2632 020452 022626 2$: CMP (SP)+,(SP)+
2633 020454 017737 161266 001126 MOV @CSR,$BDDAT ;READ STATUS
2634 020462 012737 020264 001124 MOV #BIT13!BIT7!BIT5!BIT4!BIT2,$GDDAT ;LOAD EXPECTED
2635 020470 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE VALUE
2636 020476 001401 BEQ 3$ ;;BR IF "ACTIVE" CLEARED
2637 020500 104031 ERROR 31 ;"ACTIVE" FAILED TO CLEAR
2638 020502 012777 004000 161246 3$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
2639 020510 013777 002000 161260 MOV VECTB1,@VECTB0 ;RESET THE VECTORS
2640 020516 005077 161256 CLR @VECTB1
```

```

2641
2642
2643
2644 020522 000004
2645 020524 012737 000040 001160
2646 020532 013746 002002
2647 020536 012746 020544
2648 020542 000002
2649 020544 012777 004000 161204 1$:
2650 020552 012777 020746 161216
2651 020560 012777 000340 161212
2652 020566 005077 161162
2653 020572 005077 161154
2654 020576 012777 060000 161144
2655 020604 012777 000064 161134
2656 020612 012777 000014 161136
2657 020620 012737 177777 060000
2658 020626 052777 000001 161112
2659 020634 052777 000002 161114
2660 020642 042777 000002 161106
2661 020650 000240
2662 020652 000240
2663 020654 000240
2664 020656 052777 010000 161072
2665 020664 000240
2666 020666 000240
2667 020670 000240
2668 020672 000240
2669 020674 000240
2670 020676 000240
2671 020700 013700 002002
2672 020704 162700 000040
2673 020710 005737 003354
2674 020714 001401
2675 020716 005000
2676 020720 012777 020754 161050 2$:
2677 020726 010046
2678 020730 012746 020736
2679 020734 000002
2680 020736 000240 3$:
2681 020740 000240
2682 020742 104042
2683 020744 000404
2684
2685
2686 020746 022626
2687 020750 104042
2688 020752 000401
2689
2690
2691 020754 022626
2692 020756 012777 004000 160772
2693 020764 013777 002000 161004
2694 020772 005077 161002

```

```

*****
*TEST 120   VERIFY CORRECT BR LEVEL THE NCV-11
*****
TST120: SCOPE
MOV #40,$TIMES      ;;DO 40 ITERATIONS
BRLEV,-(SP)        ;PUSH THE EXPECTED BR LEVEL ON STACK
MOV #1$,-(SP)      ;PUSH THE RETURN ADDRESS ON STACK
RTI                ;FAKE AN INTERRUPT TO BE LSI-11 AND PDP-11 COMPATABLE
MOV #CLRALL,@SFR   ;CLEAR THE DEVICE
MOV #10$,@VECTB0  ;LOAD UNEXPECTED INTERRUPT RETURN
MOV #340,@VECTB1
CLR @BAR           ;INIT THE LOW Z COUNT
CLR @WCR          ;INIT THE HIGH Z COUNT
MOV #BUFO,@OFF    ;PRIME THE OFFSET REGISTER
MOV #64,@CSR      ;SELECT INTR. AND MATRIX MODE
MOV #TSTDMA!TSTCON,@SFR ;MAINT. MODE
MOV #-1,BUFO      ;PRIME THE TARGET LOCATION
BIS #BIT0,@CSR    ;ENABLE THE NCV-11
BIS #TESTZ,@SFR   ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR   ;DISABLE 'TEST Z' PULSES
NOP
NOP
NOP
BIS #BIT12,@SFR   ;ENABLE 1 TRANSFER
NOP
NOP
NOP
NOP
NOP
MOV BRLEV,R0      ;GET CURRENT BR LEVEL
SUB #40,R0        ;AND MAKE IT 1 LEVEL LOWER
TST NLSI11       ;TEST IF LSI-11 CPU
BEQ 2$           ;BR IF NOT
CLR R0           ;LOAD BR LEVEL 0
MOV #20$,@VECTB0 ;RELOAD TO EXPECTED VECTOR
MOV R0,-(SP)     ;PUSH ADJUSTED BR LEVEL
MOV #3$,-(SP)    ;PUSH RETURN ADDRESS
RTI              ;LOWER CPU BR LEVEL
NOP
NOP
ERROR 42         ;NCV11 FAILED TO INTERRUPT - INCORRECT NCV11 BR LEVEL?
BR 21$          ;;BR TO CLEAN-UP

;UNEXPECTED INTERRUPT WITH BR LEVEL INDICATED
10$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
ERROR 42         ;NCV11 INTERRUPTED ON AN INCORRECT BR LEVEL
BR 21$          ;;BR TO CLEANUP

;EXPECTED INTERRUPT DID OCCUR
20$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
21$: MOV #CLRALL,@SFR ;CLEAN THE DEVICE
MOV VECTB1,@VECTB0 ;RESET THE VECTOR
CLR @VECTB1

```

2695
2696
2697
2698 020776 000004
2699 021000 012737 000100 001160
2700 021006 012777 004000 160742
2701 021014 012777 060000 160726
2702 021022 012777 000014 160726
2703 021030 012777 000024 160710
2704 021036 052777 000001 160702
2705 021044 052777 000002 160704
2706 021052 042777 000002 160676
2707 021060 005037 060000
2708 021064 052777 010000 160664
2709 021072 012737 060000 001122
2710 021100 012737 000001 001124
2711 021106 017737 160010 001126
2712 021114 023737 001124 001126
2713 021122 001401
2714 021124 104033
2715
2716
2717
2718
2719 021126 000004
2720 021130 012737 000100 001160
2721 021136 012777 004000 160612
2722 021144 012777 070000 160576
2723 021152 012777 000014 160576
2724 021160 012777 000024 160560
2725 021166 052777 000001 160552
2726 021174 052777 000002 160554
2727 021202 042777 000002 160546
2728 021210 005037 070000
2729 021214 052777 010000 160534
2730 021222 012737 070000 001122
2731 021230 012737 000001 001124
2732 021236 017737 157660 001126
2733 021244 023737 001124 001126
2734 021252 001401
2735 021254 104033
2736

```
::*****  
:*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  
:*****  
:*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
```


2737
2738
2739
2740 021256 000004
2741 021260 012737 000100 001160
2742 021266 012777 004000 160462
2743 021274 012777 064000 160446
2744 021302 012777 000014 160446
2745 021310 012777 000024 160430
2746 021316 052777 000001 160422
2747 021324 052777 000002 160424
2748 021332 042777 000002 160416
2749 021340 005037 064000
2750 021344 052777 010000 160404
2751 021352 012737 064000 001122
2752 021360 012737 000001 001124
2753 021366 017737 157530 001126
2754 021374 023737 001124 001126
2755 021402 001401
2756 021404 104033
2757
2758
2759
2760
2761 021406 000004
2762 021410 012737 000100 001160
2763 021416 012777 004000 160332
2764 021424 012777 062000 160316
2765 021432 012777 000014 160316
2766 021440 012777 000024 160300
2767 021446 052777 000001 160272
2768 021454 052777 000002 160274
2769 021462 042777 000002 160266
2770 021470 005037 062000
2771 021474 052777 010000 160254
2772 021502 012737 062000 001122
2773 021510 012737 000001 001124
2774 021516 017737 157400 001126
2775 021524 023737 001124 001126
2776 021532 001401
2777 021534 104033
2778

```
*****  
: *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  
*****  
: *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
```

```

2779
2780
2781
2782 021536 000004
2783 021540 012737 000100 001160
2784 021546 012777 004000 160202
2785 021554 012777 061000 160166
2786 021562 012777 000014 160166
2787 021570 012777 000024 160150
2788 021576 052777 000001 160142
2789 021604 052777 000002 160144
2790 021612 042777 000002 160136
2791 021620 005037 061000
2792 021624 052777 010000 160124
2793 021632 012737 061000 001122
2794 021640 012737 000001 001124
2795 021646 017737 157250 001126
2796 021654 023737 001124 001126
2797 021662 001401
2798 021664 104033
2799
2800
2801
2802
2803 021666 000004
2804 021670 012737 000100 001160
2805 021676 012777 004000 160052
2806 021704 012777 060400 160036
2807 021712 012777 000014 160036
2808 021720 012777 000024 160020
2809 021726 052777 000001 160012
2810 021734 052777 000002 160014
2811 021742 042777 000002 160006
2812 021750 005037 060400
2813 021754 052777 010000 157774
2814 021762 012737 060400 001122
2815 021770 012737 000001 001124
2816 021776 017737 157120 001126
2817 022004 023737 001124 001126
2818 022012 001401
2819 022014 104033
2820
  
```

```

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

```

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

```

2821
2822
2823
2824 022016 000004
2825 022020 012737 000100 001160
2826 022026 012777 004000 157722
2827 022034 012777 060200 157706
2828 022042 012777 000014 157706
2829 022050 012777 000024 157670
2830 022056 052777 000001 157662
2831 022064 052777 000002 157664
2832 022072 042777 000002 157656
2833 022100 005037 060200
2834 022104 052777 010000 157644
2835 022112 012737 060200 001122
2836 022120 012737 000001 001124
2837 022126 017737 156770 001126
2838 022134 023737 001124 001126
2839 022142 001401
2840 022144 104033
2841
2842
2843
2844
2845 022146 000004
2846 022150 012737 000100 001160
2847 022156 012777 004000 157572
2848 022164 012777 060100 157556
2849 022172 012777 000014 157556
2850 022200 012777 000024 157540
2851 022206 052777 000001 157532
2852 022214 052777 000002 157534
2853 022222 042777 000002 157526
2854 022230 005037 060100
2855 022234 052777 010000 157514
2856 022242 012737 060100 001122
2857 022250 012737 000001 001124
2858 022256 017737 156640 001126
2859 022264 023737 001124 001126
2860 022272 001401
2861 022274 104033
2862

```

```

*****
*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

```

```

*****
*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

```

```

2863
2864
2865
2866 022276 000004
2867 022300 012737 000100 001160
2868 022306 012777 004000 157442
2869 022314 012777 060040 157426
2870 022322 012777 000014 157426
2871 022330 012777 000024 157410
2872 022336 052777 000001 157402
2873 022344 052777 000002 157404
2874 022352 042777 000002 157376
2875 022360 005037 060040
2876 022364 052777 010000 157364
2877 022372 012737 060040 001122
2878 022400 012737 000001 001124
2879 022406 017737 156510 001126
2880 022414 023737 001124 001126
2881 022422 001401
2882 022424 104033
2883
2884
2885
2886
2887 022426 000004
2888 022430 012737 000100 001160
2889 022436 012777 004000 157312
2890 022444 012777 060020 157276
2891 022452 012777 000014 157276
2892 022460 012777 000024 157260
2893 022466 052777 000001 157252
2894 022474 052777 000002 157254
2895 022502 042777 000002 157246
2896 022510 005037 060020
2897 022514 052777 010000 157234
2898 022522 012737 060020 001122
2899 022530 012737 000001 001124
2900 022536 017737 156360 001126
2901 022544 023737 001124 001126
2902 022552 001401
2903 022554 104033
2904
  
```

```

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

2905
2906
2907
2908 022556 000004
2909 022560 012737 000100 001160
2910 022566 012777 004000 157162
2911 022574 012777 060010 157146
2912 022602 012777 000014 157146
2913 022610 012777 000024 157130
2914 022616 052777 000001 157122
2915 022624 052777 000002 157124
2916 022632 042777 000002 157116
2917 022640 005037 060010
2918 022644 052777 010000 157104
2919 022652 012737 060010 001122
2920 022660 012737 000001 001124
2921 022666 017737 156230 001126
2922 022674 023737 001124 001126
2923 022702 001401
2924 022704 104033
2925
2926
2927
2928
2929 022706 000004
2930 022710 012737 000100 001160
2931 022716 012777 004000 157032
2932 022724 012777 040004 157016
2933 022732 012777 000014 157016
2934 022740 012777 000024 157000
2935 022746 052777 000001 156772
2936 022754 052777 000002 156774
2937 022762 042777 000002 156766
2938 022770 005037 040004
2939 022774 052777 010000 156754
2940 023002 012737 040004 001122
2941 023010 012737 000001 001124
2942 023016 017737 156100 001126
2943 023024 023737 001124 001126
2944 023032 001401
2945 023034 104033
2946

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

```

2947
2948
2949
2950 023036 000004
2951 023040 012737 000010 001160
2952
2953
2954
2955 023046 012777 004000 156702
2956 023054 012777 062550 156666
2957 023062 012777 000014 156666
2958 023070 012777 002024 156650
2959 023076 052777 000001 156642
2960 023104 052777 000002 156644
2961 023112 042777 000002 156636
2962 023120 005037 065322
2963 023124 005037 062722
2964 023130 052777 010000 156620
2965 023136 000240
2966 023140 000240
2967 023142 000240
2968 023144 012737 065322 001122
2969 023152 012737 000001 001124
2970 023160 017737 155736 001126
2971 023166 023737 001124 001126
2972 023174 001413
2973 023176 005737 062722
2974 023202 001002
2975 023204 104033
2976
2977 023206 000406
2978 023210 013737 062722 001126 1$:
2979 023216 005037 001124
2980 023222 104033
2981
  
```

```

*****
:*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
  
```

CZ
CZ

```

2982
2983
2984
2985 023224 000004
2986 023226 012737 000010 001160
2987
2988
2989
2990 023234 012777 004000 156514
2991 023242 012777 065224 156500
2992 023250 012777 000014 156500
2993 023256 012777 004024 156462
2994 023264 052777 000002 156464
2995 023272 052777 000001 156446
2996 023300 042777 000001 156450
2997 023306 005037 072450
2998 023312 052777 010000 156436
2999 023320 012737 072450 001122
3000 023326 012737 000001 001124
3001 023334 017737 155562 001126
3002 023342 023737 001124 001126
3003 023350 001401
3004 023352 104033
3005
3006
3007
3008
3009 023354 000004
3010 023356 012737 000010 001160
3011 023364 012777 004000 156364
3012 023372 012777 060000 156350
3013 023400 012777 000014 156350
3014 023406 012777 002024 156332
3015 023414 052777 000001 156324
3016 023422 052777 000002 156326
3017 023430 042777 000002 156320
3018 023436 052777 000020 156312
3019 023444 005037 062552
3020 023450 005037 060152
3021 023454 052777 010000 156274
3022 023462 000240
3023 023464 000240
3024 023466 000240
3025 023470 012737 000001 001124
3026 023476 013737 060152 001126
3027 023504 023737 001124 001126
3028 023512 001413
3029 023514 005737 062552
3030 023520 001002
3031 023522 104034
3032 023524 000406
3033 023526 013737 062552 001126 1$:
3034 023534 005037 001124
3035 023540 104034

```

```

*****
*TEST 136 VERIFY THE ADM INPUT TO THE MATRIX MUX. ADDER USING ZB ENABLE
*****
TST136: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
;WITH ZB ENABLE SET IN TEST CONTROL MODE, THE SELF TEST DATA SHOULD BE BYTE 224
;WHEN PROCESSED THRU THE ADDRESS MAKER LOGIC THE NEW VALUE IS 12450
;VERIFY THAT THE MATRIX MUX CAN ADD CORRECTLY
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #BUF0+5224,@OFF ;LOAD THE OFFSET REGISTER
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #4024,@CSR ;ENABLE ZB AND WORD MATRIX MODE
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIS #BIT0,@CSR ;ENABLE NCV11
BIC #BIT0,@SFR ;DISABLE 'TEST Z' PULSES
CLR @#BUF0+12450 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 TRANSFER
MOV #BUF0+12450,$BDDADR ;LOAD THE EXPECTED ADDRESS
MOV #1,$GDDAT ;LOAD THE EXPECTED DATA
MOV @#BDDADR,$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

```

3036
3037
3038
3039 023542 000004
3040 023544 012737 000100 001160
3041 023552 012777 004000 156176
3042 023560 012777 050000 156162
3043 023566 012777 000014 156162
3044 023574 012777 002030 156144
3045 023602 052777 000001 156136
3046 023610 052777 000002 156140
3047 023616 042777 000002 156132
3048 023624 005037 062726
3049 023630 052777 010000 156120
3050 023636 012737 062726 001122
3051 023644 012737 000001 001124
3052 023652 017737 155244 001126
3053 023660 023737 001124 001126
3054 023666 001401
3055 023670 104033
3056
3057
3058
3059
3060 023672 000004
3061 023674 012737 000100 001160
3062 023702 012777 004000 156046
3063 023710 012777 037774 156032
3064 023716 012777 000014 156032
3065 023724 012777 002032 156014
3066 023732 052777 000001 156006
3067 023740 052777 000002 156010
3068 023746 042777 000002 156002
3069 023754 005037 065522
3070 023760 052777 010000 155770
3071 023766 012737 065522 001122
3072 023774 012737 000001 001124
3073 024002 017737 155114 001126
3074 024010 023737 001124 001126
3075 024016 001401
3076 024020 104033
3077

```
::*****  
:*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
```



```
3078
3079
3080
3081 024022 000004
3082 024024 012737 000100 001160
3083
3084
3085
3086 024032 012777 004000 155716
3087 024040 023727 036402 001340
3088 024046 103445
3089 024050 012777 050000 155672
3090 024056 012777 000014 155672
3091 024064 012777 002034 155654
3092 024072 052777 000001 155646
3093 024100 052777 000002 155650
3094 024106 042777 000002 155642
3095 024114 005037 123656
3096 024120 052777 010000 155630
3097 024126 012737 123656 001122
3098 024134 012737 000001 001124
3099 024142 017737 154754 001126
3100 024150 023737 001124 001126
3101 024156 001401
3102 024160 104033
3103

:*****
:*TEST 142 VERIFY BIT 14 ADDER INPUT TO MATRIX MODE MUX
:*****
TST142: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
:OFFSET = 50000
:ADM OUTPUT = 53656
:TARGET = 123656
MOV #CLRALL,@SFR ;CLEAR THE DEVICE
CMP $LSTBK,#1340 ;TEST IF ENOUGH MEMORY >20K
BLO TST143 ;;BR IF NO ROOM
MOV #50000,@OFF ;LOAD THE OFFSET REG.
MOV #TSTDMA!TSTCON,@SFR ;SET TEST DMA AND CONTROL
MOV #2034,@CSR ;SELECT GAIN MATRIX BYTE MODE
BIS #BIT0,@CSR ;ENABLE NCV11
BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
CLR @#BUF0+43656 ;CLEAR THE TARGET LOCATION
BIS #BIT12,@SFR ;ALLOW 1 WORD TRANSFER
MOV #BUF0+43656,$BDADR ;LOAD EXPECTED ADDRESS
MOV #1,$GDDAT ;LOAD EXPECTED DATA
MOV @$BDADR,$BDDAT ;READ THE ACTUAL DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST143 ;;BR IF SAME
ERROR 33 ;MATRIX ADDER FAILED TO SELECT CORRECT ADDRESS
;THE WRONG ADDRESS - EXPECTED ADR. WAS BUF0+43656
```

```

3104
3105
3106
3107 024162 000004
3108 024164 012737 000010 001160
3109 024172 012777 004000 155556
3110 024200 005077 155546
3111 024204 005077 155544
3112 024210 012777 060000 155532
3113 024216 012777 002022 155522
3114 024224 012777 000014 155524
3115 024232 052777 000001 155506
3116 024240 052777 000002 155510
3117 024246 042777 000002 155502
3118 024254 000240
3119 024256 000240
3120 024260 000240
3121 024262 112737 000100 061264
3122 024270 112737 000377 061265
3123 024276 052777 010000 155452
3124 024304 000240
3125 024306 000240
3126 024310 000240
3127 024312 012737 022022 001124
3128 024320 017737 155422 001126
3129 024326 023737 001124 001126
3130 024334 001401
3131 024336 104030
3132
3133 024340 013737 061264 001126 1$:
3134 024346 012737 177500 001124
3135 024354 023737 001124 001126
3136 024362 001401
3137 024364 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 INCREMENT IS THE HIGH BYTE
MOV BUFO+1264,$BDDAT ;READ 1 BUFO LOCATION
MOV #177500,$GDDAT ;LOAD EXPECTED VALUE
CMP $GDDAT,$BDDAT ;COMPARE VALUES
BEQ TST144 ;;BR IF SAME
ERROR 30 ;TARGET LOC. DATA WAS INCORRECT

```

```

3138
3139
3140
3141 024366 000004
3142 024370 012737 000400 001160
3143 024376 005737 050110
3144 024402 001454
3145 024404 012737 000001 001124
3146 024412 012737 024420 001110
3147
3148 024420 012777 004000 155330 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3149 024426 012777 000010 155322 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
3150 024434 012777 000000 155310 MOV #0,@WCR ;LOAD UPPER 16 BITS
3151 024442 013777 001124 155304 MOV $GDDAT,@BAR ;
3152 024450 005377 155300 DEC @BAR ;LOAD INITIAL COUNTER VALUE
3153 024454 012777 000022 155264 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
3154 024462 052777 000001 155256 BIS #BIT0,@CSR ;ENABLE THE DEVICE
3155 024470 052777 000001 155230 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
3156 024476 105777 155224 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
3157 024502 100375 BPL 2$
3158 024504 017737 155244 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE
3159 024512 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3160 024520 001402 BEQ 3$ ;BR IF SAME
3161 024522 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER IN ERROR
3162 024524 000403 BR TST145 ;
3163 024526 006337 001124 3$: ASL $GDDAT ;CHANGE THE DATA BIT
3164 024532 001332 BNE 1$ ;BR AND TRY NEXT BIT
3165
3166
3167
3168 024534 000004
3169 024536 012737 000400 001160
3170 024544 005737 050110
3171 024550 001454
3172 024552 012737 000001 001124
3173 024560 012737 024566 001110
3174
3175 024566 012777 004000 155162 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3176 024574 012777 000010 155154 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
3177 024602 013777 001124 155142 MOV $GDDAT,@WCR ;
3178 024610 005377 155136 DEC @WCR ;LOAD INITIAL COUNTER VALUE
3179 024614 012777 177777 155132 MOV #-1,@BAR ;LOAD LOWER 16 BITS
3180 024622 012777 000022 155116 MOV #BIT4!BIT1,@CSR ;ENTER MATRIX MODE
3181 024630 052777 000001 155110 BIS #BIT0,@CSR ;ENABLE THE DEVICE
3182 024636 052777 000001 155062 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
3183 024644 105777 155056 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
3184 024650 100375 BPL 2$
3185 024652 017737 155074 001126 MOV @WCR,$BDDAT ;READ EXPECTED REGISTER VALUE
3186 024660 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3187 024666 001402 BEQ 3$ ;BR IF SAME
3188 024670 104011 ERROR 11 ;HIGHER 16 BITS OF THE Z COUNTER IN ERROR
3189 024672 000403 BR TST146 ;
3190 024674 006337 001124 3$: ASL $GDDAT ;CHANGE THE DATA BIT
3191 024700 001332 BNE 1$ ;BR AND TRY NEXT BIT

```

```

3192
3193
3194
3195 024702 000004
3196 024704 012737 000400 001160
3197 024712 005737 050110
3198 024716 001443
3199 024720 012737 000001 001124
3200
3201 024726 012777 004000 155022 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3202 024734 012777 000010 155014 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
3203 024742 012777 000000 155002 MOV #0,@WCR ;LOAD UPPER 16 BITS
3204 024750 012777 000000 154776 MOV #0,@BAR ;LOAD LOWER 16 BITS
3205 024756 012777 000422 154762 MOV #BIT8!BIT4!BIT1,@CSR ;ENTER MATRIX MODE ON CAMERA 01
3206 024764 052777 000001 154754 BIS #BIT0,@CSR ;ENABLE THE DEVICE
3207 024772 052777 000001 154726 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
3208 025000 105777 154722 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
3209 025004 100375 BPL 2$
3210 025006 017737 154742 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE
3211 025014 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3212 025022 001401 BEQ TST147 ;:BR IF SAME
3213 025024 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER
;IN ERROR WHEN USING CAMERA 01

```

```

3214
3215
3216
3217
3218
3219 025026 000004
3220 025030 012737 000100 001160
3221 025036 005737 050110
3222 025042 001443
3223 025044 012737 000001 001124
3224
3225 025052 012777 004000 154676 1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
3226 025060 012777 000010 154670 MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
3227 025066 012777 000000 154656 MOV #0,@WCR ;LOAD UPPER 16 BITS
3228 025074 012777 000000 154652 MOV #0,@BAR ;LOAD LOWER 16 BITS
3229 025102 012777 001022 154636 MOV #BIT9!BIT4!BIT1,@CSR ;ENTER MATRIX MODE
3230 025110 052777 000001 154630 BIS #BIT0,@CSR ;ENABLE THE DEVICE
3231 025116 052777 000001 154602 BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
3232 025124 105777 154576 2$: TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
3233 025130 100375 BPL 2$
3234 025132 017737 154616 001126 MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE
3235 025140 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3236 025146 001401 BEQ TST150 ;:BR IF SAME
3237 025150 104010 ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER
;IN ERROR WHEN USING CAMERA 10

```

```

3239
3240
3241
3242 025152 000004
3243 025154 012737 000100 001160
3244 025162 005737 050110
3245 025166 001443
3246 025170 012737 000001 001124
3247
3248 025176 012777 004000 154552 1$:
3249 025204 012777 000010 154544
3250 025212 012777 000000 154532
3251 025220 012777 000000 154526
3252 025226 012777 001422 154512
3253 025234 052777 000001 154504
3254 025242 052777 000001 154456
3255 025250 105777 154452 2$:
3256 025254 100375
3257 025256 017737 154472 001126
3258 025264 023737 001124 001126
3259 025272 001401
3260 025274 104010
3261

*****
*TEST 150 VERIFY THAT CAMERA 11 CHANNEL IS OPERATIONAL (TESTER ONLY)
*****
TST150: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
TST WFMODE ;TEST IF TESTER MODE
BEQ TST151 ;;BR IF NOT
MOV #1,$GDDAT ;LOAD EXPECTED VALUE

1$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
MOV #TSTDMA,@SFR ;SET TEST DMA FLOP
MOV #0,@WCR ;LOAD UPPER 16 BITS
MOV #0,@BAR ;LOAD LOWER 16 BITS
MOV #BIT9!BIT8!BIT4!BIT1,@CSR ;ENTER MATRIX MODE
BIS #BIT0,@CSR ;ENABLE THE DEVICE
BIS #BIT0,@DACSR ;GENERATE 'Z' PULSE
TSTB @DACSR ;WAIT FOR Z PULSE COMPLETION
BPL 2$
MOV @BAR,$BDDAT ;READ EXPECTED REGISTER VALUE
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST151 ;;BR IF SAME
ERROR 10 ;LOWER 16 BITS OF THE Z COUNTER
;IN ERROR WHEN USING CAMERA 11
  
```

```

3262
3263
3264
3265 025276 000004
3266 025300 012737 000002 001160
3267
3268
3269
3270
3271
3272
3273 025306 012737 060000 050114
3274 025314 012737 020000 025662
3275 025322 012737 025330 001110
3276
3277 025330 012700 060000
3278 025334 012701 125252
3279 025340 013702 103352
3280 025344 010120
3281 025346 020002
3282 025350 001375
3283 025352 012777 177770 022534
3284
3285 025360 012777 004000 154370
3286 025366 013777 050114 154354
3287 025374 012777 177777 154350
3288 025402 005077 154346
3289 025406 012777 000004 154342
3290 025414 012777 000024 154324
3291 025422 052777 000002 154326
3292 025430 052777 000001 154310
3293
3294 025436 012700 000004
3295 025442 005001
3296 025444 032777 060000 154274 4$:
3297 025452 001014
3298 025454 005301
3299 025456 001372
3300 025460 005300
3301 025462 001370
3302 025464 017737 154256 001126
3303 025472 012737 060224 001124
3304 025500 104002
3305 025502 000470
3306 025504 012700 060000 5$:
3307 025510 023700 050114 6$:
3308 025514 001415
3309 025516 010037 001122
3310 025522 011037 001126
3311 025526 012737 125252 001124
3312 025534 023737 001124 001126
3313 025542 001417
3314 025544 104033
3315 025546 000446

:*****
:*TEST 151 DYNAMIC MATRIX MODE ADDRESS
:*****
TST151: SCOPE
MOV #2,$TIMES ;;DO 2 ITERATIONS
:CELL INCREMENT OF A FLOATING LOCATION (60000,70000,64000,62000,61000)
:FILL THE MEMORY BUFFER WITH A KNOWN PATTERN (125252)
:COLLECT DATA OF 0 AND INCREMENT A TARGET LOCATION
:VERIFY THAT NO OTHER ADDRESS IS CHANGED
:AFTER VERIFYING THE WHOLE BUFFER - UPDATE THE OFFSET REGISTER
AND INCREMENT ANOTHER LOCATION
MOV #BUFO,CURRENT ;PRIME THE CURRENT TARGET LOC.
MOV #BIT13,100$ ;LOAD FORCE BIT
MOV #2$,$LPERR ;LOAD RETURN ADDRESS
:PRIME THE BUFFER WITH A 125252 PATTERN
2$: MOV #BUFO,R0 ;LOAD POINTER TO BUFFER
MOV #125252,R1 ;LOAD VALUE
MOV ADNOKT,R2 ;LOAD BUFFER END ADDRESS
3$: MOV R1,(R0)+ ;LOAD BUFFER WITH DATA
CMP R0,R2 ;TEST FOR END
BNE 3$
MOV #-10,@CURRENT ;PRIME THE TARGET LOCATION
MOV #CLRALL,@SFR ;INIT THE DEVICE
MOV CURRENT,@OFF ;LOAD OFFSET TO TARGET
MOV #-1,@WCR ;PRIME THE HIGH 16 BIT Z COUNTER
CLR @BAR ;CLEAR LOW 16 BIT COUNTER
MOV #TSTCON,@SFR ;ENABLE TEST CONNECTOR
MOV #24,@CSR ;LOAD MATRIX WORD MODE
BIS #TESTZ,@SFR ;SET TEST Z FLOP
BIS #BIT0,@CSR ;ENABLE THE NCV11
MOV #4,R0 ;PRIME THE DELAY
CLR R1
3296 025444 032777 060000 154274 4$: BIT #BIT14!BIT13,@CSR ;TEST FOR CELL OR Z OVERFLOW
BNE 5$ ;;BR IF EITHER IS SET
DEC R1 ;DELAY
BNE 4$
DEC R0 ;DELAY
BNE 4$
MOV @CSR,$BDDAT ;READ BAD STATUS
MOV #BIT14!BIT13+224,$GDDAT ;LOAD EXPECTED STATUS
ERROR 2 ;DYNAMIC MATRIX MODE STATUS ERROR
BR TST152
3306 025504 012700 060000 5$: MOV #BUFO,R0 ;GET BUFFER POINER
3307 025510 023700 050114 6$: CMP CURRENT,R0 ;TEST IF TARGET ADDRESS
BEQ 7$ ;;BR IF YES
MOV R0,$BADDR ;GET BAD ADDRESS FOR TYPE-OUT
MOV (R0),$BDDAT ;GET BAD DATA
MOV #125252,$GDDAT ;LOAD EXPECTED DATA
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 10$ ;;BR IF SAME
ERROR 33 ;CHANGED AN INCORRECT TARGET LOCATION
BR TST152

```

```

CZNRCA NCV11  DIAGNOSTIC  MACY11 27(654) 8-AUG-78 15:05  PAGE 74  I 7
CZNRCA.P11  T151  DYNAMIC  MATRIX MODE ADDRESS  SEQ 0086

3316 025550 013737 050114 001122 7$:  MOV  CURENT,$BDADR  ;LOAD EXPECTED ADDRESS
3317 025556 017737 022332 001126  MOV  @CURENT,$BDDAT ;LOAD ACTUAL DATA
3318 025564 012737 003407 001124  MOV  #3407,$GDDAT  ;LOAD EXPECTED DATA
3319 025572 023737 001124 001126  CMP  $GDDAT,$BDDAT ;COMPARE
3320 025600 001400  BEQ  10$  ;:BR IF SAME
3321
3322 025602 005720 10$:  TST  (R0)+  ;BUMP THE POINTER INTO THE BUFFER
3323 025604 023700 003352  CMP  ADNOKT,R0 ;TEST IF FINISHED THE BUFFER
3324 025610 001337  BNE  6$  ;BR AND RETEST THE REST OF THE BUFFER
3325
3326 025612 005737 001202  TST  $PASS  ;TEST IF FIRST PASS
3327 025616 001422  BEQ  TST152  ;:BR IF YES
3328 025620 032777 004000 153312  BIT  #SW11,@SWR ;TEST INHIBIT INTER.
3329 025626 001016  BNE  TST152  ;:BR IF SET
3330 025630 006237 025662  ASR  100$  ;CHANGE THE FORCED ADDR. BIT
3331 025634 022737 000002 025662  CMP  #2,100$  ;TEST IF FINISHED
3332 025642 001410  BEQ  TST152  ;:BR IF FINISHED
3333 025644 012737 060000 050114  MOV  #BUFO,CURENT ;MAKE UP NEW ADDRESS
3334 025652 053737 025662 050114  BIS  100$,CURENT ;
3335 025660 000623  BR  2$
3336 025662 020000 100$:  BIT13
3337  ;:*****
3338  ;:*TEST 152 DYNAMIC LIST MODE ADDRESS
3339  ;:*****
3340 025664 000004  TST152: SCOPE
3341 025666 012737 000002 001160  MOV  #2,$TIMES  ;:DO 2 ITERATIONS
3342  ;:FILL THE MEMORY BUFFER WITH A KNOWN PATTERN (125252)
3343  ;:DO A 1 WORD TRANSFER TO A TAgET LOCATION
3344  ;:VERIFY THAT NO OTHER LOCATION IN THE BUFFER IS CHANGED
3345  ;:AFTER VERIFYING BUMP THE BUS ADDRESS AND DO SUB-TEST AGAIN
3346 025674 012737 060000 050114  MOV  #BUFO,CURENT ;PRIME THE CURRENT TARGET LOC.
3347 025702 012737 020000 026242  MOV  #BIT13,100$ ;LOAD FORCE BIT
3348 025710 012737 025716 001110  MOV  #2$, $LPERR ;LOAD RETURN ADDRESS
3349  ;:PRIME THE BUFFER WITH A 125252 PATTERN
3350 025716 012700 060000 2$:  MOV  #BUFO,R0 ;LOAD POINTER TO BUFFER
3351 025722 012701 125252  MOV  #125252,R1 ;LOAD VALUE
3352 025726 013702 003352  MOV  ADNOKT,R2 ;LOAD BUFFER END ADDRESS
3353 025732 010120 3$:  MOV  R1,(R0)+ ;LOAD BUFFER WITH DATA
3354 025734 020002  CMP  R0,R2 ;TEST FOR END
3355 025736 001375  BNE  3$
3356
3357 025740 012777 004000 154010  MOV  #CLRALL,@SFR ;INIT THE DEVICE
3358 025746 012777 177777 153776  MOV  #-1,@WCR ;PRIME THE WORD COUNT
3359 025754 013777 050114 153772  MOV  CURENT,@JAR ;LOAD TARGET LOCATION POINTER
3360 025762 012777 000004 153766  MOV  #TSTCON,@SFR ;ENABLE MAINT. MODE
3361 025770 052777 000001 153750  BIS  #BIT0,@CSR ;ENABLE THE NCV11
3362 025776 052777 000002 153752  BIS  #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES
3363 026004 042777 000002 153744  BIC  #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES
3364
3365 026012 012700 000004  MOV  #4,R0 ;PRIME THE DELAY
3366 026016 005001  CLR  R1
3367 026020 032777 060000 153720 4$:  BIT  #BIT14!BIT13,@CSR ;TEST FOR CELL OR Z OVERFLOW
3368 026026 001014  BNE  5$  ;:BR IF EITHER IS SET
3369 026030 005301  DEC  R1 ;DELAY

```

```

3370 026032 001372      BNE      4$
3371 026034 005300      DEC      RO          ;DELAY
3372 026036 001370      BNE      4$
3373 026040 017737 153702 001126      MOV      @CSR,$BDDAT ;READ BAD STATUS
3374 026046 012737 060224 001124      MOV      BIT14!BIT13+224,$GDDAT ;LOAD EXPECTED STATUS
3375 026054 104036      ERROR    36          ;DYNAMIC LIST MODE STATUS ERROR
3376 026056 000472      BR       TST153      ;:
3377 026060 012700 060000 5$:      MOV      #BUFO,RO    ;GET BUFFER POINER
3378 026064 023700 050114 6$:      CMP      CURENT,RO   ;TEST IF TARGET ADDRESS
3379 026070 001415      BEQ      7$          ;:BR IF YES
3380 026072 010037 001122      MOV      RO,$BDADR   ;GET BAD ADDRESS FOR TYPE-OUT
3381 026076 011037 001126      MOV      (RO),$BDDAT ;GET BAD DATA
3382 026102 012737 125252 001124      MOV      #125252,$GDDAT ;LOAD EXPECTED DATA
3383 026110 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE
3384 026116 001421      BEQ      10$         ;:BR IF SAME
3385 026120 104021      ERROR    21          ;CHANGED AN INCORRECT TARGET LOCATION
3386 026122 000450      BR       TST153      ;:
3387 026124 013737 050114 001122 7$:      MOV      CURENT,$BDADR ;LOAD EXPECTED ADDRESS
3388 026132 017737 021756 001126      MOV      @CURENT,$BDDAT ;GET ACTUAL DATA
3389 026140 012737 003407 001124      MOV      #3407,$GDDAT  ;LOAD EXPECTED DATA
3390 026146 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE
3391 026154 001402      BEQ      10$         ;:BR IF SAME
3392 026156 104037      ERROR    37          ;LIST MODE DATA ERROR
3393 026160 000431      BR       TST153      ;:
3394 026162 005720 10$:      TST      (RO)+       ;BUMP THE POINTER INTO THE BUFFER
3395 026164 023700 003352      CMP      ADNOKT,RO   ;TEST IF FINISHED THE BUFFER
3396 026170 001335      BNE      6$          ;BR AND RETEST THE REST OF THE BUFFER
3397
3398 026172 005737 001202      TST      $PASS       ;TEST IF FIRST PASS
3399 026176 001422      BEQ      TST153      ;:BR IF FIRST PASS
3400 026200 032777 004000 152732      BIT      #SW11,@SWR  ;TEST INHIBIT INTER.
3401 026206 001016      BNE      TST153      ;:BR IF SET
3402 026210 006237 026242      ASR      100$        ;CHANGE THE FORCE ADDRESS
3403 026214 022737 000002 026242      CMP      #2,100$     ;TEST IF END
3404 026222 001410      BEQ      TST153      ;:BR IF FINISHED
3405 026224 012737 060000 050114      MOV      #BUFO,CURENT ;MAKE NEW ADDRESS
3406 026232 053737 026242 050114      BIS      100$,CURENT ;
3407 026240 000626      BR       2$          ;
3408 026242 020000 100$:      BIT13

```



```
3409 ::*****
3410 :*TEST 153 DYNAMIC LIST MODE TRANSFER - MAXIMUM BUFFER LENGTH IN LOWER 28K
3411 :*****
3412 026244 000004 TST153: SCOPE
3413 026246 012737 000040 001160 MOV #40,$TIMES ;:DO 40 ITERATIONS
3414 026254 013700 003352 MOV ADNOKT,R0 ;:GET LAST ADDRESS
3415 026260 162700 060000 SUB #BUFO,R0 ;:DETERMINE THE WORD COUNT VALUE
3416 026264 005400 NEG R0
3417 026266 006200 ASR R0
3418 ;PRIME THE BUFFER WITH A 125252 PATTERN
3419 026270 012703 060000 2$: MOV #BUFO,R3 ;:LOAD POINTER TO BUFFER
3420 026274 012701 125252 MOV #125252,R1 ;:LOAD VALUE
3421 026300 013702 003352 MOV ADNOKT,R2 ;:LOAD BUFFER END ADDRESS
3422 026304 010123 3$: MOV R1,(R3)+ ;:LOAD BUFFER WITH DATA
3423 026306 020302 CMP R3,R2 ;:TEST FOR END
3424 026310 001375 BNE 3$
3425 ;NOW COLLET THE LIST MODE DATA
3426 026312 012777 004000 153436 MOV #CLRALL,@SFR ;:INIT THE DEVICE
3427 026320 010077 153426 MOV R0,@WCR ;:PRIME THE WORD COUNT
3428 026324 012777 060000 153422 MOV #BUFO,@BAR ;:LOAD TARGET LOCATION POINTER
3429 026332 012777 000004 153416 MOV #TSTCON,@SFR ;:ENABLE MAINT. MODE
3430 026340 052777 000001 153400 BIS #BIT0,@CSR ;:ENABLE THE NCV11
3431 026346 052777 000002 153402 BIS #TESTZ,@SFR ;:ENABLE MAINT. Z
3432 026354 012700 000004 MOV #4,R0 ;:PRIME THE DELAY
3433 026360 032777 060000 153360 4$: BIT #BIT14!BIT13,@CSR ;:TEST FOR CELL OR Z OVERFLOW
3434 026366 001014 BNE 5$ ;:BR IF EITHER IS SET
3435 026370 005301 DEC R1 ;:DELAY
3436 026372 001372 BNE 4$
3437 026374 005300 DEC R0 ;:DELAY
3438 026376 001370 BNE 4$
3439 026400 017737 153342 001126 MOV @CSR,$BDDAT ;:READ BAD STATUS
3440 026406 012737 060200 001124 MOV #BIT14!BIT13+200,$GDDAT ;:LOAD EXPECTED STATUS
3441 026414 104036 ERROR 36 ;:DYNAMIC LIST MODE STATUS ERROR - NO WORD COUNT OVERFLOW
3442 026416 000423 BR TST154 ;:
3443 026420 012700 060000 5$: MOV #BUFO,R0 ;:GET BUFFER POINER
3444 026424 010037 001122 6$: MOV R0,$BDADR ;:GET BAD ADDRESS FOR TYPE-OUT
3445 026430 011037 001126 MOV (R0),$BDDAT ;:GET BAD DATA
3446 026434 012737 003407 001124 MOV #3407,$GDDAT ;:LOAD EXPECTED DATA
3447 026442 023737 001124 001126 CMP $GDDAT,$BDDAT ;:COMPARE
3448 026450 001402 BEQ 10$ ;:BR IF SAME
3449 026452 104037 ERROR 37 ;:LIST MODE DATA ERROR
3450 026454 000404 BR TST154 ;:
3451 026456 005720 10$: TST (R0)+ ;:BUMP THE POINTER INTO THE BUFFER
3452 026460 023700 003352 CMP ADNOKT,R0 ;:TEST IF FINISHED THE BUFFER
3453 026464 001357 BNE 6$ ;:BR AND RETEST THE REST OF THE BUFFER
```

```

3454      ;:*****
3455      ;*TEST 154      ONE MATRIX DATA TRANSFER TO EACH 4K EXTENDED MEMORY
3456      ;:*****
3457 026466 000004      TST154: SCOPE
3458 026470 012737 000002 001160      MOV #2,$TIMES      ;;DO 2 ITERATIONS
3459 026476 005737 036120      TST $KT11      ;:TEST IF KT-11 INSTALLED
3460 026502 100127      BPL TST155      ;:BR IF NONE
3461      ;DO A BYTE MATRIX MODE TRANSFER TO A 1 BYTE LOCATION IN EACH 4K EXTENDED
3462      ; MEMORY BANK
3463 026504 012737 000600 041772      MOV #600,OUT      ;:LOAD INITIAL BANK VALUE
3464 026512 013737 041772 172346 1$: MOV OUT,@#KIPAR3      ;:LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
3465 026520 012700 060000      MOV #BUF0,R0      ;:LOAD BUFFER POINTER.
3466 026524 012710 125252      MOV #125252,(R0)      ;:PRIME BAD TARGET LOC. IF EXT ADD FAILS
3467 026530 052737 001400 177572      BIS #1400,@#SRO      ;:ENABLE MAINT. MODE KT-11
3468 026536 012710 000370      MOV #370,(R0)      ;:LOAD TARGET LOCATION DATA VALUE
3469 026542 042737 001400 177572      BIC #1400,@#SRO      ;:DISABLE MAINT. MODE KT-11
3470 026550 013701 041772      MOV OUT,R1      ;:GET BANK VALUE
3471 026554 005002      CLR R2      ;:CLEAR EXT. ADD. TEMPORARY
3472 026556 006301      ASL R1      ;:MOVE LEFT
3473 026560 006301      ASL R1      ;:MOVE LEFT
3474 026562 006301      ASL R1      ;:MOVE LEFT
3475 026564 006301      ASL R1      ;:MOVE LEFT
3476 026566 006301      ASL R1      ;:MOVE LEFT
3477 026570 006102      ROL R2      ;:SAVE EA BITS
3478 026572 006301      ASL R1      ;:MOVE LEFT
3479 026574 006102      ROL R2      ;:SAVE EA BITS
3480 026576 010237 041756      MOV R2,NARROW      ;:SAVE EA BITS FOR TYPEOUT
3481 026602 010137 001122      MOV R1,$BDADR      ;:SAVE ADDRESS BITS FOR TYPEOUT
3482 026606 060201      ADD R2,R1      ;:MAKE COMPLETE ADDRESS
3483
3484      ;NOW GET READY TO DO THE TRANSFER
3485 026610 012777 004000 153140      MOV #CLRALL,@SFR      ;:INIT THE NCV11
3486 026616 010177 153126      MOV R1,@OFF      ;:LOAD COMBINED BUFFER ADDRESS
3487 026622 012777 000022 153116      MOV #BIT4!BIT1,@CSR      ;:ENABLE THE NCV11
3488 026630 012777 000014 153120      MOV #TSTDMA!TSTCON,@SFR      ;:ENABLE MAINT NCV11 MODE
3489 026636 052777 000001 153102      BIS #BIT0,@CSR      ;:ENABLE THE NCV11
3490 026644 052777 000002 153104      BIS #TESTZ,@SFR      ;:ENABLE 'TEST Z' PULSES
3491 026652 042777 000002 153076      BIC #TESTZ,@SFR      ;:DISABLE 'TEST Z' PULSES
3492 026660 012737 000371 001124      MOV #371,$GDDAT      ;:LOAD EXPECTED DATA
3493 026666 052777 010000 153062      BIS #BIT12,@SFR      ;:ENABLE 1 BYTE TRANSFER
3494 026674 052737 000001 177572      BIS #BIT0,@#SRO      ;:ENABLE KT-11
3495 026702 011037 001126      MOV (R0),$BDDAT      ;:GET ACTUAL DATA
3496 026706 042737 000001 177572      BIC #BIT0,@#SRO      ;:DISABLE KT-11
3497 026714 023737 001124 001126      CMP $GDDAT,$BDDAT      ;:COMPARE DATA
3498 026722 001401      BEQ 2$      ;:BR IF SAME
3499 026724 104035      ERROR 35      ;:DATA TRANSFER ERROR TO EXTENDED MEMORY
3500 026726 005737 001202      TST $PASS      ;:TEST PASS COUNTER
3501 026732 001413      BEQ TST155      ;:BR IF FIRST PASS
3502 026734 032777 004000 152176      BIT #SW11,@SWR      ;:TEST IF INHIBIT INTER.
3503 026742 001007      BNE TST155      ;:BR IF SET
3504 026744 062737 000200 041772      ADD #200,OUT      ;:UPDATE BANK VALUE
3505 026752 023737 036402 041772      CMP $LSTBK,OUT      ;:TEST IF DONE
3506 026760 101254      BHI 1$      ;:BR IF NOT

```

```

3507
3508
3509
3510 026762 000004
3511 026764 012737 000002 001160
3512 026772 005737 036120
3513 026776 100130
3514
3515 027000 012737 000600 041772
3516 027006 013737 041772 172346
3517 027014 012700 060000
3518 027020 012710 052525
3519 027024 052737 001400 177572
3520 027032 012710 125252
3521 027036 042737 001400 177572
3522 027044 013701 041772
3523 027050 005002
3524 027052 006301
3525 027054 006301
3526 027056 006301
3527 027060 006301
3528 027062 006301
3529 027064 006102
3530 027066 006301
3531 027070 006102
3532
3533
3534 027072 012777 004000 152656
3535 027100 010277 152644
3536 027104 012777 177777 152640
3537 027112 010177 152636
3538 027116 012777 000014 152632
3539 027124 052777 000001 152614
3540 027132 052777 000002 152616
3541 027140 042777 000002 152610
3542 027146 010137 001122
3543 027152 010237 041756
3544 027156 052777 010000 152572
3545 027164 012737 003407 001124
3546 027172 052737 000001 177572
3547 027200 011037 001126
3548 027204 042737 000001 177572
3549 027212 023737 001124 001126
3550 027220 001401
3551 027222 104035
3552 027224 005737 001202
3553 027230 001413
3554 027232 032777 004000 151700
3555 027240 001007
3556 027242 062737 000200 041772
3557 027250 023737 036402 041772
3558 027256 101253

*****
*TEST 155 ONE LIST DATA TRANSFER TO EACH 4K EXTENDED MEMORY
*****
TST155: SCOPE
MOV #2,$TIMES ;DO 2 ITERATIONS
TST $KT11 ;TEST IF KT-11 INSTALLED
BPL TST156 ;BR IF NONE
;DO A 1 WORD MODE TRANSFER TO A 1 WORD BUFFER IN EACH 4K EXTENDED MEMORY BANK
MOV #600,OUT ;LOAD INITIAL BANK VALUE <60000>
1$: MOV OUT,@#KIPAR3 ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
MOV #BUF0,R0 ;LOAD BUFFER POINTER
MOV #52525,(R0) ;LOAD BAD TARGET LOC. IF EXT ADD FAILS
BIS #1400,@#SRO ;ENABLE MAINT. MODE KT-11
MOV #125252,(R0) ;PRESET THE TARGET LOCATION
BIC #1400,@#SRO ;DISABLE MAINT. MODE KT-11
MOV OUT,R1 ;GET BANK VALUE
CLR R2 ;CLEAR EXT. ADD TEMPORARY
ASL R1
ASL R1
ASL R1
ASL R1
ASL R1
ROL R2 ;SAVE EXT. ADDRESS BITS
ASL R1
ROL R2 ;SAVE EXT. ADDRESS BITS

;NOW GET READY TO DO THE TRANSFER
MOV #CLRALL,@SFR ;INIT THE NCV11
MOV R2,@OFF ;LOAD THE EXTENDED ADDRESS BITS
MOV #-1,@WCR ;LOAD WORD COUNT
MOV R1,@BAR ;LOAD BUS ADDRESS
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
MOV R1,$BDADR ;SAVE TARGET ADDRESS
MOV R2,NARROW ;SAVE EA BITS FOR ERROR TYPEOUT
BIS #BIT12,@SFR ;ENABLE 1 BYTE TRANSFER
MOV #3407,$GDDAT ;LOAD EXPECTED DATA
BIS #BIT0,@#SRO ;ENABLE KT-11
MOV (R0),$BDDAT ;GET ACTUAL DATA
BIC #BIT0,@#SRO ;DISABLE KT-11
CMP $GDDAT,$BDDAT ;COMPARE DATA
BEQ 2$ ;BR IF SAME
ERROR 35 ;LIST MODE DATA TRANSFER ERROR TO EXTENDED MEMORY
2$: TST $PASS ;TEST PASS COUNTER
BEQ TST156 ;BR IF FIRST PASS
BIT #SW11,@SWR ;TEST INHIBIT INTER.
BNE TST156 ;BR IF SET
ADD #200,OUT ;UPDATE BANK VALUE
CMP $LSTBK,OUT ;TEST IF DONE
BHI 1$ ;BR IF NOT
  
```

```

3559      ::*****
3560      :*TEST 156      VERIFY BIT 15 MATRIX ADDER INPUT
3561      :*****
3562 027260 000004      TST156: SCOPE
3563 027262 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
3564      :ADM OUTPUT      =      127657
3565      :OFFSET          =      140000
3566      :TARGET          =      267657
3567 027270 023727 036402 003140      CMP      $LSTBK,#3140      ;TEST IF >52K IS AVAILABLE
3568 027276 103476      BLO      TST157          ;;BR IF NO MORE MEMORY
3569 027300 012737 002676 172346      MOV      #2676,@#KIPAR3    ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
3570 027306 012700 060056      MOV      #BUF0+56,R0      ;LOAD BUFFER POINTER
3571 027312 012710 125252      MOV      #125252,(R0)     ;LOAD BAD TARGET LOC. IF EXT ADD FAILS
3572 027316 052737 001400 177572      BIS      #1400,@#SRO      ;ENABLE MAINT. MODE KT-11
3573 027324 005010      CLR      (R0)            ;LOAD TARGET LOCATION DATA VALUE
3574 027326 042737 001400 177572      BIC      #1400,@#SRO      ;DISABLE MAINT. MODE KT-11
3575
3576      ;NOW GET READY TO DO THE TRANSFER
3577 027334 012777 004000 152414      MOV      #CLRALL,@SFR      ;INIT THE NCV11
3578 027342 012777 140000 152400      MOV      #140000,@OFF     ;LOAD COMBINED BUFFER ADDRESS
3579 027350 012777 002036 152370      MOV      #2036,@CSR      ;ENABLE THE NCV11
3580 027356 012777 000014 152372      MOV      #TSTDMA!TSTCON,@SFR ;ENABLE MAINT NCV11 MODE
3581 027364 052777 000001 152354      BIS      #BIT0,@CSR      ;ENABLE THE NCV11
3582 027372 052777 000002 152356      BIS      #TESTZ,@SFR      ;ENABLE 'TEST Z' PULSES
3583 027400 042777 000002 152350      BIC      #TESTZ,@SFR      ;DISABLE 'TEST Z' PULSES
3584 027406 000240      NOP
3585 027410 000240      NOP
3586 027412 052777 010000 152336      BIS      #BIT12,@SFR      ;ENABLE 1 BYTE TRANSFER
3587 027420 000240      NOP
3588 027422 000240      NOP
3589 027424 000240      NOP
3590 027426 012737 000001 041756      MOV      #1,NARROW        ;SAVE EA BITS FOR ERROR TYPEOUT
3591 027434 012737 000400 001124      MOV      #400,$GDDAT      ;LOAD EXPECTED DATA
3592 027442 052737 000001 177572      BIS      #BIT0,@#SRO      ;ENABLE KT-11
3593 027450 011037 001126      MOV      (R0),$BDDAT      ;SAVE ACTUAL DATA
3594 027454 042737 000001 177572      BIC      #BIT0,@#SRO      ;DISABLE KT-11
3595 027462 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE DATA
3596 027470 001401      BEQ      TST157          ;;BR IF SAME
3597 027472 104020      ERROR      20            ;BIT 15 INPUT TO MATRIX MODE ADDER FAILED

```

```
3598 ::*****  
3599 :*TEST 157 VERIFY HIGH BYTE OPERATION OF THE 'TEST X'  
3600 :*****  
3601 027474 000004 TST157: SCOPE  
3602 027476 012737 000040 001160 MOV #40,$TIMES ;:DO 40 ITERATIONS  
3603 :TEST X = 1  
3604 :OFFSET =100000  
3605 :ADM OUTPUT =127657  
3606 :TARGET =100257  
3607 027504 012777 004000 152244 MOV #CLRALL,@SFR ;CLEAR THE DEVICE  
3608 027512 023727 036402 001140 CMP $LSTBK,#1140 ;TEST IF LEAST 20K  
3609 027520 103445 BLO TST160 ;:BR IF NOT  
3610 027522 012777 100000 152220 MOV #100000,@OFF ;SET BIT 15 OF OFFSET  
3611 027530 012777 000034 152220 MOV #TSTDMA!TSTCON!BIT4,@SFR ;SET TEST DMA, CONTROL AND TESTX  
3612 027536 012777 002036 152202 MOV #2036,@CSR ;ENABLE THE NCV11  
3613 027544 052777 000001 152174 BIS #BIT0,@CSR ;ENABLE THE NCV11  
3614 027552 052777 000002 152176 BIS #TESTZ,@SFR ;ENABLE 'TEST Z' PULSES  
3615 027560 042777 000002 152170 BIC #TESTZ,@SFR ;DISABLE 'TEST Z' PULSES  
3616 027566 005037 100256 CLR @#BUF0+20256 ;CLEAR THE TARGET LOC.  
3617 027572 052777 010000 152156 BIS #BIT12,@SFR ;ALLOW 1 DMA TRANSFER  
3618 027600 012737 100256 001122 MOV #BUF0+20256,$BDADR ;LOAD EXPECTED ADDRESSES VALUE  
3619 027606 012737 000400 001124 MOV #400,$GDDAT ;LOAD EXPECTED VALUE READ  
3620 027614 017737 151302 001126 MOV @$BDADR,$BDDAT ;READ ACTUAL DATA  
3621 027622 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
3622 027630 001401 BEQ TST160 ;:BR IF SAME  
3623 027632 104034 ERROR 34 ;MATRIX ADDER INPUT OF ADM15 AND TESTX L  
3624 ;FAILED TO BE INHIBITED
```

```

3625
3626
3627
3628 027634 000004
3629 027636 012737 000100 001160
3630
3631
3632
3633 027644 023727 036402 005140
3634 027652 103500
3635 027654 012737 004635 172346
3636 027662 012700 060026
3637 027666 012710 125252
3638 027672 052737 001400 177572
3639 027700 005010
3640 027702 042737 001400 177572
3641
3642
3643 027710 012777 004000 152040
3644 027716 012777 010001 152024
3645 027724 012777 000014 152024
3646 027732 012777 004036 152006
3647 027740 052777 000001 152000
3648 027746 052777 000002 152002
3649 027754 042777 000002 151774
3650 027762 000240
3651 027764 000240
3652 027766 052777 010000 151762
3653 027774 000240
3654 027776 000240
3655 030000 000240
3656 030002 012737 000002 041756
3657 030010 012737 000400 001124
3658 030016 052737 000001 177572
3659 030024 111037 001126
3660 030030 042737 000001 177572
3661 030036 105037 001126
3662 030042 023737 001124 001126
3663 030050 001401
3664 030052 104020

```

```

*****
*TEST 160 VERIFY BIT 16 INPUT TO THE MATRIX MODE ADDER
*****
TST160: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
;ADM OUTPUT = 253527
;OFFSET = 210000
;TARGET = 463527
CMP $LSTBK,#5140 ;TEST IF ENOUGH MEMORY >84K
BLO TST161 ;;BR IF NOT
MOV #4635,@#KIPAR3 ;LOAD KT-PAR REG #3 <BUFFER IS AT LOC 60000>
MOV #BUF0+26,R0 ;LOAD BUFFER POINTER
MOV #125252,(R0) ;LOAD BAD TARGET LOC.
BIS #1400,@#SRO ;ENABLE MAINT. MODE KT-11
CLR (R0) ;PRESET THE TARGET LOCATION
BIC #1400,@#SRO ;DISABLE MAINT. MODE KT-11

;NOW GET READY TO DO THE TRANSFER
MOV #CLRALL,@SFR ;INIT THE NCV11
MOV #10001,@OFF ;LOAD THE EXTENDED ADDRESS BITS
MOV #TSTDMA!TSTCON,@SFR ;ENABLE MAINT NCV11 MODE
MOV #4036,@CSR ;LOAD MATRIX MODE AND ZB ENABLE
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
NOP
MOV #2,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
CLRB $BDDAT ;MASK OFF LOW BYTE
CMP $GDDAT,$BDDAT ;COMPARE DATA
BEQ TST161 ;;BR IF SAME
ERROR 20 ;BIT 16 INPUT TO MATRIX MODE ADDER FAILED

```

```

3665
3666
3667
3668 030054 000004
3669 030056 012737 000001 001160
3670 030064 005737 050106
3671 030070 001402
3672 030072 000137 043712
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682 030076
3683 030076 000004
3684 030100 005037 001102
3685 030104 005037 001160
3686 030110 005237 001202
3687 030114 042737 100000 001202
3688 030122 005327
3689 030124 000001
3690 030126 003022
3691 030130 012737
3692 030132 000001
3693 030134 030124
3694 030136 104401 030203
3695 030142 013746 001202
3696 030146 104405
3697 030150 104401 030200
3698 030154 013700 000042
3699 030160 001405
3700 030162 000005
3701 030164 004710
3702 030166 000240
3703 030170 000240
3704 030172 000240
3705 030174
3706 030174 000137
3707 030176 003356
3708 030200 377 377 000
3709 030203 015 042412 042116
3710 030210 050040 051501 020123
3711 030216 000043

*****
*TEST 161 DETERMINE IF DIFLIN IS TO BE RUN (F)
*****
TST161: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST RUNDIF ;;TEST IF DIFLIN IS TO BE RUN
BEQ $EOP ;;BR IF NOT TO BE RUN
JMP DIFLIN ;;JUMP AND RUN DIFLIN
.SBTTL END OF PASS ROUTINE

*****
*INCREMENT THE PASS NUMBER ($PASS)
*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO LOGIC

$EOP:
SCOPE
CLR $TSTNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT
TYPE ,$ENDMG ;;TYPE 'END PASS #'
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE ,$ENULL ;;TYPE A NULL CHARACTER
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
$DOAGN: JMP @ (PC)+ ;;RETURN
$RTNAD: .WORD LOGIC
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12>/END PASS #/

```

```
3712 .SBTTL ERROR ASCII MESSAGES
3713
3714 030220 034115 031060 004466 EM1: .ASCIZ /M8026 NCV11 BUS ADDRESS TIMEOUT/
3715 030226 041516 030526 020061
3716 030234 052502 020123 042101
3717 030242 051104 051505 020123
3718 030250 044524 042515 052517
3719 030256 000124
3720 030260 034115 031060 004466 EM2: .ASCIZ /M8026 COMMAND-STATUS REGISTER ERROR/
3721 030266 047503 046515 047101
3722 030274 026504 052123 052101
3723 030302 051525 051040 043505
3724 030310 051511 042524 020122
3725 030316 051105 047522 000122
3726 030324 034115 031060 004466 EM3: .ASCIZ /M8026 SPECIAL FUNCTION REGISTER ERROR/
3727 030332 050123 041505 040511
3728 030340 020114 052506 041516
3729 030346 044524 047117 051040
3730 030354 043505 051511 042524
3731 030362 020122 051105 047522
3732 030370 000122
3733 030372 034115 031060 004466 EM4: .ASCIZ /M8026 WORD COUNT REGISTER ERROR/
3734 030400 047527 042122 041440
3735 030406 052517 052116 051040
3736 030414 043505 051511 042524
3737 030422 020122 051105 047522
3738 030430 000122
3739 030432 034115 031060 004466 EM5: .ASCIZ /M8026 BUS ADDRESS REGISTER ERROR/
3740 030440 052502 020123 042101
3741 030446 051104 051505 020123
3742 030454 042522 044507 052123
3743 030462 051105 042440 051122
3744 030470 051117 000
3745 030473 115 030070 033062 EM6: .ASCIZ /M8026 OFFSET REGISTER ERROR/
3746 030500 047411 043106 042523
3747 030506 020124 042522 044507
3748 030514 052123 051105 042440
3749 030522 051122 051117 000
3750 030527 115 030070 033062 EM7: .ASCIZ /M8026 DUAL REGISTER SELECTION ERROR/
3751 030534 042011 040525 020114
3752 030542 042522 044507 052123
3753 030550 051105 051440 046105
3754 030556 041505 044524 047117
3755 030564 042440 051122 051117
3756 030572 000
3757 030573 115 030070 033062 EM10: .ASCIZ /M8026-M8036 LOW 16 BIT Z COUNT ERROR/
3758 030600 046455 030070 033063
3759 030606 046011 053517 030440
3760 030614 020066 044502 020124
3761 030622 020132 047503 047125
3762 030630 020124 051105 047522
3763 030636 000122
3764 030640 034115 031060 026466 EM11: .ASCIZ /M8026-M8036 HIGH 16 BIT Z COUNT ERROR/
3765 030646 034115 031460 004466
```


3766	030654	044510	044107	030440		
3767	030662	020066	044502	020124		
3768	030670	020132	047503	047125		
3769	030676	020124	051105	047522		
3770	030704	000122				
3771	030706	034115	031060	004466	EM12:	.ASCIZ /M8026 Z COUNT STATUS ERROR/
3772	030714	020132	047503	047125		
3773	030722	020124	052123	052101		
3774	030730	051525	042440	051122		
3775	030736	051117	000			
3776	030741	115	030070	033062	EM13:	.ASCIZ /M8026 Z COUNT INTERRUPT ERROR/
3777	030746	055011	041440	052517		
3778	030754	052116	044440	052116		
3779	030762	051105	052522	052120		
3780	030770	042440	051122	051117		
3781	030776	000				
3782	030777	115	030070	033063	EM14:	.ASCIZ /M8036 JOYSTICK STATUS ERROR/
3783	031004	045011	054517	052123		
3784	031012	041511	020113	052123		
3785	031020	052101	051525	042440		
3786	031026	051122	051117	000		
3787	031033	115	030070	033063	EM15:	.ASCIZ /M8036 JOYSTICK DATA ERROR/
3788	031040	045011	054517	052123		
3789	031046	041511	020113	040504		
3790	031054	040524	042440	051122		
3791	031062	051117	000			
3792	031065	115	030070	033063	EM16:	.ASCIZ /M8036 DATA INCREMENT ERROR/
3793	031072	042011	052101	020101		
3794	031100	047111	051103	046505		
3795	031106	047105	020124	051105		
3796	031114	047522	000122			
3797	031120	034115	031460	004466	EM17:	.ASCIZ /M8036 DATA DECREMENT ERROR/
3798	031126	040504	040524	042040		
3799	031134	041505	042522	042515		
3800	031142	052116	042440	051122		
3801	031150	051117	000			
3802	031153	115	030070	033062	EM20:	.ASCIZ /M8026-M8036 MATRIX MODE ADDRESS MAKER DATA ERROR/
3803	031160	046455	030070	033063		
3804	031166	046411	052101	044522		
3805	031174	020130	047515	042504		
3806	031202	040440	042104	042522		
3807	031210	051523	046440	045501		
3808	031216	051105	042040	052101		
3809	031224	020101	051105	047522		
3810	031232	000122				
3811	031234	034115	031060	004466	EM21:	.ASCIZ /M8026 LIST MODE ADDRESS MAKER DATA ERROR/
3812	031242	044514	052123	046440		
3813	031250	042117	020105	042101		
3814	031256	051104	051505	020123		
3815	031264	040515	042513	020122		
3816	031272	040504	040524	042440		
3817	031300	051122	051117	000		
3818	031305	115	030070	033062	EM22:	.ASCIZ /M8026 LIST MODE TRANSFER BUS ADDRESS ERROR/
3819	031312	046011	051511	020124		

3820	031320	047515	042504	052040		
3821	031326	040522	051516	042506		
3822	031334	020122	052502	020123		
3823	031342	042101	051104	051505		
3824	031350	020123	051105	047522		
3825	031356	000122				
3826	031360	034115	031060	004466	EM23:	.ASCIZ /M8026 LIST MODE TRANSFER WORD COUNT ERROR/
3827	031366	044514	052123	046440		
3828	031374	042117	020105	051124		
3829	031402	047101	043123	051105		
3830	031410	053440	051117	020104		
3831	031416	047503	047125	020124		
3832	031424	051105	047522	000122		
3833	031432	034115	031060	004466	EM24:	.ASCIZ /M8026 LIST MODE TRANSFER OFFSET ERROR/
3834	031440	044514	052123	046440		
3835	031446	042117	020105	051124		
3836	031454	047101	043123	051105		
3837	031462	047440	043106	042523		
3838	031470	020124	051105	047522		
3839	031476	000122				
3840	031500	034115	031060	004466	EM25:	.ASCIZ /M8026 TIMEOUT STATUS ERROR/
3841	031506	044524	042515	052517		
3842	031514	020124	052123	052101		
3843	031522	051525	042440	051122		
3844	031530	051117	000			
3845	031533	115	030070	033062	EM26:	.ASCIZ /M8026 TIMEOUT INTERRUPT ERROR/
3846	031540	052011	046511	047505		
3847	031546	052125	044440	052116		
3848	031554	051105	052522	052120		
3849	031562	042440	051122	051117		
3850	031570	000				
3851	031571	115	030070	033062	EM27:	.ASCIZ /M8026 SET 'EVENT' OR 'TIME' DATA ERROR/
3852	031576	051411	052105	021040		
3853	031604	053105	047105	021124		
3854	031612	047440	020122	052042		
3855	031620	046511	021105	042040		
3856	031626	052101	020101	051105		
3857	031634	047522	000122			
3858	031640	034115	031060	004466	EM30:	.ASCIZ /M8026 CELL INCREMENT DATA ERROR/
3859	031646	042503	046114	044440		
3860	031654	041516	042522	042515		
3861	031662	052116	042040	052101		
3862	031670	020101	051105	047522		
3863	031676	000122				
3864	031700	034115	031060	004466	EM31:	.ASCIZ /M8026 CELL OVERFLOW STATUS ERROR/
3865	031706	042503	046114	047440		
3866	031714	042526	043122	047514		
3867	031722	020127	052123	052101		
3868	031730	051525	042440	051122		
3869	031736	051117	000			
3870	031741	115	030070	033062	EM32:	.ASCIZ /M8026 CELL OVERFLOW INTERRUPT ERROR/
3871	031746	041411	046105	020114		
3872	031754	053117	051105	046106		
3873	031762	053517	044440	052116		

3874	031770	051105	052522	052120	
3875	031776	042440	051122	051117	
3876	032004	000			
3877	032005	115	030070	033062	EM33: .ASCIZ /M8026 MATRIX MODE ADDRESS MUX ERROR/
3878	032012	046411	052101	044522	
3879	032020	020130	047515	042504	
3880	032026	040440	042104	042522	
3881	032034	051523	046440	054125	
3882	032042	042440	051122	051117	
3883	032050	000			
3884	032051	115	030070	033062	EM34: .ASCIZ /M8026 "TESTX" FUNCTION ERROR/
3885	032056	021011	042524	052123	
3886	032064	021130	043040	047125	
3887	032072	052103	047511	020116	
3888	032100	051105	047522	000122	
3889	032106	034115	031060	004466	EM35: .ASCIZ /M8026 DATA ERROR WHEN TRANSFERING TO EXTENDED MEMORY/
3890	032114	040504	040524	042440	
3891	032122	051122	051117	053440	
3892	032130	042510	020116	051124	
3893	032136	047101	043123	051105	
3894	032144	047111	020107	047524	
3895	032152	042440	052130	047105	
3896	032160	042504	020104	042515	
3897	032166	047515	054522	000	
3898	032173	115	030070	033062	EM36: .ASCIZ /M8026 LIST MODE TRANSFER STATUS ERROR/
3899	032200	046011	051511	020124	
3900	032206	047515	042504	052040	
3901	032214	040522	051516	042506	
3902	032222	020122	052123	052101	
3903	032230	051525	042440	051122	
3904	032236	051117	000		
3905	032241	115	030070	033062	EM37: .ASCIZ /M8026 LIST MODE TRANSFER DATA ERROR/
3906	032246	046011	051511	020124	
3907	032254	047515	042504	052040	
3908	032262	040522	051516	042506	
3909	032270	020122	040504	040524	
3910	032276	042440	051122	051117	
3911	032304	000			
3912	032305	112	046525	042520	EM40: .ASCIZ /JUMPER-M8026-M7952 "EVENT" OR "TIME" MARK ERROR/
3913	032312	026522	034115	031060	
3914	032320	026466	033515	032471	
3915	032326	004462	042442	042526	
3916	032334	052116	020042	051117	
3917	032342	021040	044524	042515	
3918	032350	020042	040515	045522	
3919	032356	042440	051122	051117	
3920	032364	000			
3921	032365	115	034467	031065	EM41: .ASCIZ /M7952 CLOCK BUS ADDRESS TIMEOUT/
3922	032372	041411	047514	045503	
3923	032400	041040	051525	040440	
3924	032406	042104	042522	051523	
3925	032414	052040	046511	047505	
3926	032422	052125	000		
3927	032425	115	031070	033461	EM42: .ASCIZ /M8217 INCORRECT INTERRUPT LEVEL/


```

3975          .CBTTL  TTY INPUT ROUTINE
3976
3977          ;:*****
3978          .ENABL  LSB
3979
3980          ;:*****
3981          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3982          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3983          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3984          ;*WHEN OPERATING IN TTY FLAG MODE.
3985 033004 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
3986 033012 001074                BNE      15$                ;; BRANCH IF NO
3987 033014 105777 146124                TSTB    @TKS                ;; CHAR THERE?
3988 033020 100071                BPL      15$                ;; IF NO, DON'T WAIT AROUND
3989 033022 117746 146120                MOVB    @TKB,-(SP)          ;; SAVE THE CHAR
3990 033026 042716 177600                BIC     #^C177,(SP)        ;; STRIP-OFF THE ASCII
3991 033032 022726 000007                CMP     #7,(SP)+          ;; IS IT A CONTROL G?
3992 033036 001062                BNE     15$                ;; NO, RETURN TO USER
3993 033040 123727 001134 000001        CMPB    $AUTOB,#1          ;; ARE WE RUNNING IN AUTO-MODE?
3994 033046 001456                BEQ     15$                ;; BRANCH IF YES
3995
3996 033050 104401 033531                TYPE   , $CNTLG           ;; ECHO THE CONTROL-G (^G)
3997 033054 104401 033536        $GTSWR: TYPE   , $MSWR           ;; TYPE CURRENT CONTENTS
3998 033060 013746 000176                MOV     SWREG,-(SP)        ;; SAVE SWREG FOR TYPEOUT
3999 033064 104402                TYPOC  ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
4000 033066 104401 033547                TYPE   , $MNEW           ;; PROMPT FOR NEW SWR
4001 033072 005046                19$:   CLR     -(SP)        ;; CLEAR COUNTER
4002 033074 005046                CLR     -(SP)        ;; THE NEW SWR
4003 033076 105777 146042                7$:   TSTB    @TKS                ;; CHAR THERE?
4004 033102 100375                BPL     7$                ;; IF NOT TRY AGAIN
4005
4006 033104 117746 146036                MOVB    @TKB,-(SP)          ;; PICK UP CHAR
4007 033110 042716 177600                BIC     #^C177,(SP)        ;; MAKE IT 7-BIT ASCII
4008
4009
4010
4011 033114 021627 000025                9$:   CMP     (SP),#25        ;; IS IT A CONTROL-U?
4012 033120 001005                BNE     10$                ;; BRANCH IF NOT
4013 033122 104401 033524                TYPE   , $CNTLU          ;; YES, ECHO CONTROL-U (^U)
4014 033126 062706 000006                20$:  ADD     #6,SP          ;; IGNORE PREVIOUS INPUT
4015 033132 000757                BR      19$                ;; LET'S TRY IT AGAIN
4016
4017
4018 033134 021627 000015                10$:  CMP     (SP),#15        ;; IS IT A <CR>?
4019 033140 001022                BNE     16$                ;; BRANCH IF NO
4020 033142 005766 000004                TST     4(SP)              ;; YES, IS IT THE FIRST CHAR?
4021 033146 001403                BEQ     11$                ;; BRANCH IF YES
4022 033150 016677 000002 145762        MOV     2(SP),@SWR          ;; SAVE NEW SWR
4023 033156 062706 000006                11$:  ADD     #6,SP          ;; CLEAR UP STACK
4024 033162 104401 001171                14$:  TYPE   , $CRLF           ;; ECHO <CR> AND <LF>
4025 033166 123727 001135 000001        CMPB    $INTAG,#1          ;; RE-ENABLE TTY KBD INTERRUPTS?
4026 033174 001003                BNE     15$                ;; BRANCH IF NOT
4027 033176 012777 000100 145740        MOV     #100,@TKS          ;; RE-ENABLE TTY KBD INTERRUPTS
4028 033204 000002                15$:  RTI                    ;; RETURN

```

```

4029 033206 004737 034604      16$: JSR    PC,$TYPEC      ::ECHO CHAR
4030 033212 021627 000060      CMP    (SP),#60          ::CHAR < 0?
4031 033216 002420                BLT    18$              ::BRANCH IF YES
4032 033220 021627 000067      CMP    (SP),#67          ::CHAR > 7?
4033 033224 003015                BGT    18$              ::BRANCH IF YES
4034 033226 042726 000060      BIC    #60,(SP)+        ::STRIP-OFF ASCII
4035 033232 005766 000002      TST    2(SP)            ::IS THIS THE FIRST CHAR
4036 033236 001403                BEQ    17$              ::BRANCH IF YES
4037 033240 006316                ASL    (SP)              ::NO, SHIFT PRESENT
4038 033242 006316                ASL    (SP)              ::CHAR OVER TO MAKE
4039 033244 006316                ASL    (SP)              ::ROOM FOR NEW ONE.
4040 033246 005266 000002      17$: INC    2(SP)          ::KEEP COUNT OF CHAR
4041 033252 056616 177776      BIS    -2(SP),(SP)      ::SET IN NEW CHAR
4042 033256 000707                BR     7$                ::GET THE NEXT ONE
4043 033260 104401 001170      18$: TYPE  $QUES        ::TYPE ?<CR><LF>
4044 033264 000720                BR     20$              ::SIMULATE CONTROL-U
4045
4046      .DSABL  LSB
4047
4048      ::*****
4049      ::*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4050      ::*CALL:
4051      ::*      RDCHR          ::INPUT A SINGLE CHARACTER FROM THE TTY
4052      ::*      RETURN HERE   ::CHARACTER IS ON THE STACK
4053      ::*                               ::WITH PARITY BIT STRIPPED OFF
4054      ::*
4055      ::
4056 033266 011646      $RDCHR: MOV    (SP),-(SP)  ::PUSH DOWN THE PC
4057 033270 016666 000004 000002  MOV    4(SP),2(SP)      ::SAVE THE PS
4058 033276 105777 145642      1$:  TSTB   @TKS         ::WAIT FOR
4059 033302 100375                BPL    1$                ::A CHARACTER
4060 033304 117766 145636 000004  MOVB   @TKB,4(SP)      ::READ THE TTY
4061 033312 042766 177600 000004  BIC    #^C<177>,4(SP)  ::GET RID OF JUNK IF ANY
4062 033320 026627 000004 000023  CMP    4(SP),#23       ::IS IT A CONTROL-S?
4063 033326 001013                BNE    3$                ::BRANCH IF NO
4064 033330 105777 145610      2$:  TSTB   @TKS         ::WAIT FOR A CHARACTER
4065 033334 100375                BPL    2$                ::LOOP UNTIL ITS THERE
4066 033336 117746 145604      MOVB   @TKB,-(SP)      ::GET CHARACTER
4067 033342 042716 177600      BIC    #^C177,(SP)    ::MAKE IT 7-BIT ASCII
4068 033346 022627 000021      CMP    (SP)+,#21      ::IS IT A CONTROL-Q?
4069 033352 001366                BNE    2$                ::IF NOT DISCARD IT
4070 033354 000750                BR     1$                ::YES, RESUME
4071 033356 026627 000004 000140  3$:  CMP    4(SP),#140    ::IS IT UPPER CASE?
4072 033364 002407                BLT    4$                ::BRANCH IF YES
4073 033366 026627 000004 000175  CMP    4(SP),#175     ::IS IT A SPECIAL CHAR?
4074 033374 003003                BGT    4$                ::BRANCH IF YES
4075 033376 042766 000040 000004  BIC    #40,4(SP)      ::MAKE IT UPPER CASE
4076 033404 000002      4$:  RTI                    ::GO BACK TO USER
4077      ::*****
4078      ::*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4079      ::*CALL:
4080      ::*      RDLIN         ::INPUT A STRING FROM THE TTY
4081      ::*      RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4082      ::*                               ::TERMINATOR WILL BE A BYTE OF ALL 0'S

```

```

4083
4084 033406 010346          $RDLIN: MOV    R3,-(SP)      ;;SAVE R3
4085 033410 012703 033514  1$:    MOV    #$TTYIN,R3    ;;GET ADDRESS
4086 033414 022703 033524  2$:    CMP    #$TTYIN+8.,R3  ;;BUFFER FULL?
4087 033420 101405          BLOS   4$                    ;;BR IF YES
4088 033422 104410          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
4089 033424 112613          MOVB   (SP)+,(R3)          ;;GET CHARACTER
4090 033426 122713 000177  10$:   CMPB   #177,(R3)        ;;IS IT A RUBOUT
4091 033432 001003          BNE    3$                    ;;SKIP IF NOT
4092 033434 104401 001170  4$:    TYPE   ,SQUES          ;;TYPE A '?'
4093 033440 000763          BR     1$                    ;;CLEAR THE BUFFER AND LOOP
4094 033442 111337 033512  3$:    MOVB   (R3),9$          ;;ECHO THE CHARACTER
4095 033446 104401 033512          TYPE   ,9$
4096 033452 122723 000015          CMPB   #15,(R3)+          ;;CHECK FOR RETURN
4097 033456 001356          BNE    2$                    ;;LOOP IF NOT RETURN
4098 033460 105063 177777          CLRB   -1(R3)             ;;CLEAR RETURN (THE 15)
4099 033464 104401 001172          TYPE   ,SLF              ;;TYPE A LINE FEED
4100 033470 012603          MOV    (SP)+,R3          ;;RESTORE R3
4101 033472 011646          MOV    (SP),-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4102 033474 016666 000004 000002  MOV    4(SP),2(SP)        ;;FIRST ASCII CHARACTER ON IT
4103 033502 012766 033514 000004  MOV    #$TTYIN,4(SP)
4104 033510 000002          RTI                      ;;RETURN
4105 033512 000          9$:    .BYTE   0              ;;STORAGE FOR ASCII CHAR. TO TYPE
4106 033513 000          .BYTE   0              ;;TERMINATOR
4107 033514 000010          $TTYIN: .BLKB  8.        ;;RESERVE 8 BYTES FOR TTY INPUT
4108 033524 052536 005015 000  $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
4109 033531 0136 006507 000012  $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
4110 033536 005015 053523 020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
4111 033544 020075 000
4112 033547 040 047040 053505  $MNEW:  .ASCIZ  / NEW = /
4113 033554 036440 000040

```

```

4114          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
4115
4116          ::*****
4117          ::*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4118          ::*CHANGE IT TO BINARY.
4119          ::*CALL:
4120          ::*      RDOCT          ::READ AN OCTAL NUMBER
4121          ::*      RETURN HERE   ::LOW ORDER BITS ARE ON TOP OF THE STACK
4122          ::*                  ::HIGH ORDER BITS ARE IN $HIOCT
4123
4124 033560 011646          $RDOCT: MOV      (SP),-(SP)      ::PROVIDE SPACE FOR THE
4125 033562 016666 000004 000002      MOV      4(SP),2(SP)      ::INPUT NUMBER
4126 033570 010046          MOV      R0,-(SP)      ::PUSH R0 ON STACK
4127 033572 010146          MOV      R1,-(SP)      ::PUSH R1 ON STACK
4128 033574 010246          MOV      R2,-(SP)      ::PUSH R2 ON STACK
4129 033576 104411          1$:  RDLIN          ::READ AN ASCII LINE
4130 033600 012600          MOV      (SP)+,R0      ::GET ADDRESS OF 1ST CHARACTER
4131 033602 005001          CLR      R1          ::CLEAR DATA WORD
4132 033604 005002          CLR      R2
4133 033606 112046          2$:  MOVB      (R0)+,-(SP)      ::PICKUP THIS CHARACTER
4134 033610 001412          BEQ      3$          ::IF ZERO GET OUT
4135 033612 006301          ASL      R1          ::*2
4136 033614 006102          ROL      R2
4137 033616 006301          ASL      R1          ::*4
4138 033620 006102          ROL      R2
4139 033622 006301          ASL      R1          ::*8
4140 033624 006102          ROL      R2
4141 033626 042716 177770          BIC      #^C7,(SP)      ::STRIP THE ASCII JUNK
4142 033632 062601          ADD      (SP)+,R1      ::ADD IN THIS DIGIT
4143 033634 000764          BR      2$          ::LOOP
4144 033636 005726          3$:  TST      (SP)+      ::CLEAN TERMINATOR FROM STACK
4145 033640 010166 000012          MOV      R1,12(SP)      ::SAVE THE RESULT
4146 033644 010237 033660          MOV      R2,$HIOCT
4147 033650 012602          MOV      (SP)+,R2      ::POP STACK INTO R2
4148 033652 012601          MOV      (SP)+,R1      ::POP STACK INTO R1
4149 033654 012600          MOV      (SP)+,R0      ::POP STACK INTO R0
4150 033656 000002          RTI          ::RETURN
4151 033660 000000          $HIOCT: .WORD 0      ::HIGH ORDER BITS GO HERE

```



```

4152          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4153
4154          ;:*****
4155          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4156          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4157          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
4158          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4159          ;*REPLACED WITH SPACES.
4160          ;*CALL:
4161          ;*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
4162          ;*      TYPDS          ;;GO TO THE ROUTINE
4163
4164          $TYPDS:
4165          033662 010046          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
4166          033664 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
4167          033666 010246          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
4168          033670 010346          MOV      R3,-(SP)          ;;PUSH R3 ON STACK
4169          033672 010546          MOV      R5,-(SP)          ;;PUSH R5 ON STACK
4170          033674 012746 020200  MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
4171          033700 016605 000020  MOV      20(SP),R5        ;;GET THE INPUT NUMBER
4172          033704 100004          BPL      1$                ;;BR IF INPUT IS POS.
4173          033706 005405          NEG      R5                ;;MAKE THE BINARY NUMBER POS.
4174          033710 112766 000055 000001  MOVB     #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
4175          033716 005000          CLR      R0                ;;ZERO THE CONSTANTS INDEX
4176          033720 012703 034076          MOV      #$DBLK,R3        ;;SETUP THE OUTPUT POINTER
4177          033724 112723 000040          MOVB     #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
4178          033730 005002          CLR      R2                ;;CLEAR THE BCD NUMBER
4179          033732 016001 034066          MOV      $DTBL(R0),R1     ;;GET THE CONSTANT
4180          033736 160105          SUB      R1,R5            ;;FORM THIS BCD DIGIT
4181          033740 002402          BLT      4$                ;;BR IF DONE
4182          033742 005202          INC      R2                ;;INCREASE THE BCD DIGIT BY 1
4183          033744 000774          BR       3$
4184          033746 060105          4$:      ADD      R1,R5        ;;ADD BACK THE CONSTANT
4185          033750 005702          TST      R2                ;;CHECK IF BCD DIGIT=0
4186          033752 001002          BNE      5$                ;;FALL THROUGH IF 0
4187          033754 105716          TSTB    (SP)              ;;STILL DOING LEADING 0'S?
4188          033756 100407          BMI      7$                ;;BR IF YES
4189          033760 106316          5$:      ASLB     (SP)        ;;MSD?
4190          033762 103003          BCC      6$                ;;BR IF NO
4191          033764 116663 000001 177777  MOVB     1(SP),-1(R3)     ;;YES--SET THE SIGN
4192          033772 052702 000060          BIS      #'0,R2           ;;MAKE THE BCD DIGIT ASCII
4193          033776 052702 000040          7$:      BIS      #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4194          034002 110223          MOVB     R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4195          034004 005720          TST      (R0)+           ;;JUST INCREMENTING
4196          034006 020027 000010          CMP      R0,#10          ;;CHECK THE TABLE INDEX
4197          034012 002746          BLT      2$                ;;GO DO THE NEXT DIGIT
4198          034014 003002          BGT      8$                ;;GO TO EXIT
4199          034016 010502          MOV      R5,R2           ;;GET THE LSD
4200          034020 000764          BR       6$                ;;GO CHANGE TO ASCII
4201          034022 105726          8$:      TSTB    (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
4202          034024 100003          BPL      9$                ;;BR IF NO
4203          034026 116663 177777 177776  MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
4204          034034 105013          9$:      CLRB    (R3)         ;;SET THE TERMINATOR
4205          034036 012605          MOV      (SP)+,R5        ;;POP STACK INTO R5

```

```

4206 034040 012603          MOV      (SP)+,R3          ;;POP STACK INTO R3
4207 034042 012602          MOV      (SP)+,R2          ;;POP STACK INTO R2
4208 034044 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
4209 034046 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
4210 034050 104401 034076    TYPE      $DBLK           ;;NOW TYPE THE NUMBER
4211 034054 016666 000002 000004    MOV      2(SP),4(SP)      ;;ADJUST THE STACK
4212 034062 012616          MOV      (SP)+,(SP)
4213 034064 000002          RTI                          ;;RETURN TO USER
4214 034066 023420          $DTBL: 10000.
4215 034070 001750          1000.
4216 034072 000144          100.
4217 034074 000012          10.
4218 034076 000004          $DBLK: .BLKW 4
4219                                     .SBTTL SCOPE HANDLER ROUTINE
4220
4221                                     ;:*****
4222                                     ;:THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4223                                     ;:AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4224                                     ;:AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4225                                     ;:THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4226                                     ;:*SW14=1      LOOP ON TEST
4227                                     ;:*SW11=1      INHIBIT ITERATIONS
4228                                     ;:*SW09=1      LOOP ON ERROR
4229                                     ;:*SW08=1      LOOP ON TEST IN SWR<7:0>
4230                                     ;:*CALL
4231                                     ;:*      SCOPE      ;;SCOPE=IOT
4232
4233                                     $SCOPE:
4234 034106 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
4235 034110 004737 046042    JSR      PC,CTRLCG          ;:TEST FOR CTRL C OR G
4236 034114 032777 040000 145016 1$: BIT      #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
4237 034122 001114          BNE      $OVER          ;;YES IF SW14=1
4238                                     ;:#####START OF CODE FOR THE XOR TESTER#####
4239 034124 000416          $XTSTR: BR      6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
4240                                     ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
4241 034126 013746 000004    MOV      @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4242 034132 012737 034152 000004    MOV      #5$,@#ERRVEC    ;;SET FOR TIMEOUT
4243 034140 005737 177060    TST      @#177060          ;;TIME OUT ON XOR?
4244 034144 012637 000004    MOV      (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
4245 034150 000463          BR      $SVLAD          ;;GO TO THE NEXT TEST
4246 034152 022626          5$: CMP      (SP)+,(SP)+    ;;CLEAR THE STACK AFTER A TIME OUT
4247 034154 012637 000004    MOV      (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
4248 034160 000423          BR      7$          ;;LOOP ON THE PRESENT TEST
4249 034162          6$:;#####END OF CODE FOR THE XOR TESTER#####
4250 034162 032777 000400 144750    BIT      #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
4251 034170 001404          BEQ      2$          ;;BR IF NO
4252 034172 127737 144742 001102    CMPB    @SWR,$TSTNM        ;;ON THE RIGHT TEST? SWR<7:0>
4253 034200 001465          BEQ      $OVER          ;;BR IF YES
4254 034202 105737 001103 2$: TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
4255 034206 001421          BEQ      3$          ;;BR IF NO
4256 034210 123737 001115 001103    CMPB    $ERMAX,$ERFLG      ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4257 034216 101015          BHI      3$          ;;BR IF NO
4258 034220 032777 001000 144712    BIT      #BIT09,@SWR          ;;LOOP ON ERROR?
4259 034226 001404          BEQ      4$          ;;BR IF NO

```

```

4260 034230 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
4261 034236 000446 BR $OVER
4262 034240 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
4263 034244 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4264 034250 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
4265 034252 032777 004000 144660 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
4266 034260 001011 BNE 1$ ;;BR IF YES
4267 034262 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
4268 034266 001406 BEQ 1$ ;; INHIBIT ITERATIONS
4269 034270 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
4270 034274 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
4271 034302 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
4272 034304 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
4273 034312 013737 034370 001160 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
4274 034320 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
4275 034324 113737 001102 001200 MOV $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
4276 034332 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
4277 034336 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
4278 034342 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4279 034346 112737 000001 001115 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4280 034354 013777 001102 144560 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
4281 034362 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
4282 034366 000002 RTI ;;FIXES PS
4283 034370 003720 $MXCNT: 2000. ;;MAX. NUMBER OF ITERATIONS
4284 .SBTTL TYPE ROUTINE
4285
4286 *****
4287 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4288 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4289 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4290 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4291 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4292 *
4293 *CALL:
4294 *1) USING A TRAP INSTRUCTION
4295 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4296 *OR
4297 * TYPE
4298 * MESADR
4299 *
4300
4301 034372 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
4302 034376 100002 BPL 1$ ;;BR IF YES
4303 034400 000000 HALT ;;HALT HERE IF NO TERMINAL
4304 034402 000430 BR 3$ ;;LEAVE
4305 034404 010046 1$: MOV R0,-(SP) ;;SAVE R0
4306 034406 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
4307 034412 122737 000001 001214 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4308 034420 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
4309 034422 132737 000100 001215 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
4310 034430 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
4311 034432 010037 034442 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
4312 034436 004737 035444 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
4313 034442 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
  
```

```
4314 034444 132737 000040 001215 62$: BITB #APTC SUP,$ENVM ;;APT CONSOLE SUPPRESSED
4315 034452 001003 BNE 60$ ;;YES,SKIP TYPE OUT
4316 034454 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4317 034456 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
4318 034460 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
4319 034462 012600 60$: MOV (SP)+,RO ;;RESTORE RO
4320 034464 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
4321 034470 000002 RTI ;;RETURN
4322 034472 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
4323 034476 001430 BEQ 8$
4324 034500 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
4325 034504 001006 BNE 5$
4326 034506 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
4327 034510 104401 TYPE ;;TYPE A CR AND LF
4328 034512 001171 $CRLF
4329 034514 105037 034650 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
4330 034520 000755 BR 2$ ;;GET NEXT CHARACTER
4331 034522 004737 034604 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
4332 034526 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
4333 034532 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
4334 034534 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
4335 ;;AND THE NULL CHAR.
4336 034540 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
4337 034544 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
4338 034546 004737 034604 JSR PC,$TYPEC ;;GO TYPE A NULL
4339 034552 105337 034650 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
4340 034556 000770 BR 7$ ;;LOOP
4341
4342 ;HORIZONTAL TAB PROCESSOR
4343
4344 034560 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
4345 034564 004737 034604 9$: JSR PC,$TYPEC ;;TYPE A SPACE
4346 034570 132737 000007 034650 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
4347 034576 001372 BNE 9$ ;;TAB STOP
4348 034600 005726 TST (SP)+ ;;POP SPACE OFF STACK
4349 034602 000724 BR 2$ ;;GET NEXT CHARACTER
4350 034604 105777 144340 $TYPEC: TSTB @$TPS ;;WAIT UNTIL PRINTER IS READY
4351 034610 100375 BPL $TYPEC
4352 034612 116677 000002 144332 MOVB 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4353 034620 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
4354 034626 001003 BNE 1$ ;;BRANCH IF NO
4355 034630 105037 034650 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
4356 034634 000406 BR $TYPEX ;;EXIT
4357 034636 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
4358 034644 001402 BEQ $TYPEX ;;BRANCH IF YES
4359 034646 105227 INCB (PC)+ ;;COUNT THE CHARACTER
4360 034650 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
4361 034652 000207 $TYPEX: RTS PC
4362
```

```

4363          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
4364
4365          ::*****
4366          ::*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4367          ::*OCTAL (ASCII) NUMBER AND TYPE IT.
4368          ::*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4369          ::*CALL:
4370          ::*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
4371          ::*      TYPOS      ::CALL FOR TYPEOUT
4372          ::*      .BYTE   N              ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4373          ::*      .BYTE   M              ::M=1 OR 0
4374          ::*                                          ::1=TYPE LEADING ZEROS
4375          ::*                                          ::0=SUPPRESS LEADING ZEROS
4376
4377          ::*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4378          ::*$TYPOS OR $TYPOC
4379          ::*CALL:
4380          ::*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
4381          ::*      TYPON      ::CALL FOR TYPEOUT
4382
4383          ::*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
4384          ::*CALL:
4385          ::*      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
4386          ::*      TYPOC      ::CALL FOR TYPEOUT
4387
4388          034654 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ::PICKUP THE MODE
4389          034660 116637 000001 035077  MOVVB    1(SP), $OFILL      ::LOAD ZERO FILL SWITCH
4390          034666 112637 035101          MOVVB    (SP)+, $OMODE+1  ::NUMBER OF DIGITS TO TYPE
4391          034672 062716 000002          ADD      #2, (SP)      ::ADJUST RETURN ADDRESS
4392          034676 000406          BR      $TYPON
4393          034700 112737 000001 035077  $TYPOC: MOVVB    #1, $OFILL      ::SET THE ZERO FILL SWITCH
4394          034706 112737 000006 035101  MOVVB    #6, $OMODE+1  ::SET FOR SIX(6) DIGITS
4395          034714 112737 000005 035076  $TYPON: MOVVB    #5, $OCNT      ::SET THE ITERATION COUNT
4396          034722 010346          MOV      R3, -(SP)      ::SAVE R3
4397          034724 010446          MOV      R4, -(SP)      ::SAVE R4
4398          034726 010546          MOV      R5, -(SP)      ::SAVE R5
4399          034730 113704 035101          MOVVB    $OMODE+1, R4  ::GET THE NUMBER OF DIGITS TO TYPE
4400          034734 005404          NEG      R4
4401          034736 062704 000006          ADD      #6, R4        ::SUBTRACT IT FOR MAX. ALLOWED
4402          034742 110437 035100          MOVVB    R4, $OMODE    ::SAVE IT FOR USE
4403          034746 113704 035077          MOVVB    $OFILL, R4    ::GET THE ZERO FILL SWITCH
4404          034752 016605 000012          MOV      12(SP), R5    ::PICKUP THE INPUT NUMBER
4405          034756 005003          CLR      R3            ::CLEAR THE OUTPUT WORD
4406          034760 006105          1$:     ROL      R5      ::ROTATE MSB INTO 'C'
4407          034762 000404          BR      3$
4408          034764 006105          2$:     ROL      R5      ::FORM THIS DIGIT
4409          034766 006105          ROL      R5
4410          034770 006105          ROL      R5
4411          034772 010503          MOV      R5, R3
4412          034774 006103          3$:     ROL      R3      ::GET LSB OF THIS DIGIT
4413          034776 105337 035100          DECB    $OMODE        ::TYPE THIS DIGIT?
4414          035002 100016          BPL     7$            ::BR IF NO
4415          035004 042703 177770          BIC     #177770, R3   ::GET RID OF JUNK
4416          035010 001002          BNE     4$            ::TEST FOR 0

```

```

4417 035012 005704          TST      R4          ;;SUPPRESS THIS 0?
4418 035014 001403          BEQ      5$          ;;BR IF YES
4419 035016 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
4420 035020 052703 000060     BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
4421 035024 052703 000040     5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
4422 035030 110337 035074     MOV      R3,8$       ;;SAVE FOR TYPING
4423 035034 104401 035074     TYPE     ,8$        ;;GO TYPE THIS DIGIT
4424 035040 105337 035076     7$: DECB    $OCNT     ;;COUNT BY 1
4425 035044 003347          BGT      2$          ;;BR IF MORE TO DO
4426 035046 002402          BLT      6$          ;;BR IF DONE
4427 035050 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
4428 035052 000744          BR       2$          ;;GO DO THE LAST DIGIT
4429 035054 012605     6$: MOV      (SP)+,R5    ;;RESTORE R5
4430 035056 012604          MOV      (SP)+,R4    ;;RESTORE R4
4431 035060 012603          MOV      (SP)+,R3    ;;RESTORE R3
4432 035062 016666 000002 000004  MOV      2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
4433 035070 012616          MOV      (SP)+,(SP)
4434 035072 000002          RTI                    ;;RETURN
4435 035074 000          8$: .BYTE   0          ;;STORAGE FOR ASCII DIGIT
4436 035075 000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
4437 035076 000     $OCNT: .BYTE  0      ;;OCTAL DIGIT COUNTER
4438 035077 000     $OFILL: .BYTE  0    ;;ZERO FILL SWITCH
4439 035100 000000     $OMODE: .WORD  0    ;;NUMBER OF DIGITS TO TYPE
4440
4441          .SBTTL  ERROR HANDLER ROUTINE
4442
4443          ;;*****
4444          ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4445          ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4446          ;;AND GO TO $ERRTYP ON ERROR
4447          ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4448          ;;*SW15=1      HALT ON ERROR
4449          ;;*SW13=1      INHIBIT ERROR TYPEOUTS
4450          ;;*SW10=1     BELL ON ERROR
4451          ;;*SW09=1     LOOP ON ERROR
4452          ;;*CALL
4453          ;;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4454
4455          $ERROR:
4456 035102 104407          7$: CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
4457 035104 105237 001103     INCB    $ERFLG      ;;SET THE ERROR FLAG
4458 035110 001775          BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
4459 035112 013777 001102 144022  MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4460 035120 032777 002000 144012  BIT      #BIT10,@SWR  ;;BELL ON ERROR?
4461 035126 001402          BEQ      1$          ;;NO - SKIP
4462 035130 104401 001164          TYPE     , $BELL     ;;RING BELL
4463 035134 005237 001112     1$: INC      $ERTTL   ;;COUNT THE NUMBER OF ERRORS
4464 035140 011637 001116     MOV      (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
4465 035144 162737 000002 001116  SUB      #2,$ERRPC
4466 035152 117737 143740 001114  MOV      @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
4467 035160 032777 020000 143752  BIT      #BIT13,@SWR  ;;SKIP TYPEOUT IF SET
4468 035166 001004          BNE      20$         ;;SKIP TYPEOUTS
4469 035170 004737 035302     JSR     PC,$ERRTYP   ;;GO TO USER ERROR ROUTINE
4470 035174 104401 001171     TYPE     , $CRLF

```

```

4471 035200
4472 035200 122737 000001 001214 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4473 035206 001007 BNE 2$ ;;NO,SKIP APT ERROR REPORT
4474 035210 113737 001114 035222 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4475 035216 004737 035454 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
4476 035222 000 21$: .BYTE 0
4477 035223 000 .BYTE 0
4478 035224 000777 22$: BR 22$ ;;APT ERROR LOOP
4479 035226 005777 143706 2$: TST @SWR ;;HALT ON ERROR
4480 035232 100002 BPL 3$ ;;SKIP IF CONTINUE
4481 035234 000000 HALT ;;HALT ON ERROR!
4482 035236 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4483 035240 032777 001000 143672 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
4484 035246 001402 BEQ 4$ ;;BR IF NO
4485 035250 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4486 035254 005737 001162 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
4487 035260 001402 BEQ 5$ ;;BR IF NONE
4488 035262 013716 001162 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4489 035266
4490 035266 022737 030164 000042 5$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
4491 035274 001001 BNE 6$ ;;BRANCH IF NO
4492 035276 000000 HALT ;;YES
4493 035300
4494 035300 000002 6$: RTI ;;RETURN
  
```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
  
```

 *THIS ROUTINE USES THE 'ITEM CONTROL BYTE' (\$ITEMB) TO DETERMINE WHICH
 *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' (\$ERRTB),
 *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

$ERRTYP:
TYPE , $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
MOV RO,-(SP) ;;SAVE RO
CLR RO ;;PICKUP THE ITEM INDEX
BISB @#$ITEMB,RO
BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
TYPE THE PC OF THE ERROR
MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
ERROR ADDRESS
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 6$ ;;GET OUT
1$: DEC RO ;;ADJUST THE INDEX SO THAT IT WILL
ASL RO ;; WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD #ERRTB,RO ;;FORM TABLE POINTER
MOV (RO)+,2$ ;;PICKUP 'ERROR MESSAGE' POINTER
BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER
TYPE ;;TYPE THE 'ERROR MESSAGE'
2$: .WORD 0 ;;'ERROR MESSAGE' POINTER GOES HERE
TYPE , $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
3$: MOV (RO)+,4$ ;;PICKUP 'DATA HEADER' POINTER
  
```

```

4525 035366 001404          BEQ      5$          ::SKIP TYPEOUT IF 0
4526 035370 104401          TYPE          ::TYPE THE 'DATA HEADER'
4527 035372 000000          4$: .WORD 0      ::'DATA HEADER' POINTER GOES HERE
4528 035374 104401 001171  TYPE      , $CRLF  ::'CARRIAGE RETURN' & 'LINE FEED'
4529 035400 011000          5$: MOV      (R0),R0  ::PICKUP 'DATA TABLE' POINTER
4530 035402 001004          BNE      7$          ::GO TYPE THE DATA
4531 035404 012600          6$: MOV      (SP)+,R0  ::RESTORE R0
4532 035406 104401 001171  TYPE      , $CRLF  ::'CARRIAGE RETURN' & 'LINE FEED'
4533 035412 000207          RTS      PC          ::RETURN
4534 035414
4535 035414 013046          MOV      @(R0)+,-(SP)  ::SAVE @(R0)+ FOR TYPEOUT
4536 035416 104402          TYPOC          ::GO TYPE--OCTAL ASCII(ALL DIGITS)
4537 035420 005710          TST      (R0)       ::IS THERE ANOTHER NUMBER?
4538 035422 001770          BEQ      6$          ::BR IF NO
4539 035424 104401 035432  TYPE      , 8$       ::TYPE TWO(2) SPACES
4540 035430 000771          BR      7$          ::LOOP
4541 035432 020040 000      8$: .ASCIZ  / /      ::TWO(2) SPACES
4542 035436          .EVEN
  
```



```

4543          .SBTTL  APT COMMUNICATIONS ROUTINE
4544
4545          ::*****
4546 035436 112737 000001 035702 $ATY1:  MOVB  #1,$FFLG      ::TO REPORT FATAL ERROR
4547 035444 112737 000001 035700 $ATY3:  MOVB  #1,$MFLG      ::TO TYPE A MESSAGE
4548 035452 000403          BR          $ATYC
4549 035454 112737 000001 035702 $ATY4:  MOVB  #1,$FFLG      ::TO ONLY REPORT FATAL ERROR
4550 035462          $ATYC:
4551 035462 010046          MOV    R0,-(SP)      ::PUSH R0 ON STACK
4552 035464 010146          MOV    R1,-(SP)      ::PUSH R1 ON STACK
4553 035466 105737 035700          TSTB  $MFLG      ::SHOULD TYPE A MESSAGE?
4554 035472 001450          BEQ   5$          ::IF NOT: BR
4555 035474 122737 000001 001214          CMPB  #APTENV,$ENV      ::OPERATING UNDER APT?
4556 035502 001031          BNE   3$          ::IF NOT: BR
4557 035504 132737 000100 001215          BITB  #APTSPOOL,$ENVM  ::SHOULD SPOOL MESSAGES?
4558 035512 001425          BEQ   3$          ::IF NOT: BR
4559 035514 017600 000004          MOV    @4(SP),R0      ::GET MESSAGE ADDR.
4560 035520 062766 000002 000004          ADD    #2,4(SP)      ::BUMP RETURN ADDR.
4561 035526 005737 001174          1$:  TST   $MSGTYPE      ::SEE IF DONE W/ LAST XMISSION?
4562 035532 001375          BNE   1$          ::IF NOT: WAIT
4563 035534 010037 001210          MOV    R0,$MSGAD      ::PUT ADDR IN MAILBOX
4564 035540 105720          2$:  TSTB  (R0)+        ::FIND END OF MESSAGE
4565 035542 001376          BNE   2$
4566 035544 163700 001210          SUB    $MSGAD,R0      ::SUB START OF MESSAGE
4567 035550 006200          ASR   R0              ::GET MESSAGE LNTH IN WORDS
4568 035552 010037 001212          MOV    R0,$MSGGLT     ::PUT LENGTH IN MAILBOX
4569 035556 012737 000004 001174          MOV    #4,$MSGTYPE    ::TELL APT TO TAKE MSG.
4570 035564 000413          BR    5$
4571 035566 017637 000004 035612 3$:  MOV    @4(SP),4$      ::PUT MSG ADDR IN JSR LINKAGE
4572 035574 062766 000002 000004          ADD    #2,4(SP)      ::BUMP RETURN ADDRESS
4573 035602 013746 177776          MOV    177776,-(SP)  ::PUSH 177776 ON STACK
4574 035606 004737 034372          JSR   PC,$TYPE      ::CALL TYPE MACRO
4575 035612 000000          4$:  .WORD  0
4576 035614          5$:
4577 035614 105737 035702          10$: TSTB  $FFLG          ::SHOULD REPORT FATAL ERROR?
4578 035620 001416          BEQ   12$          ::IF NOT: BR
4579 035622 005737 001214          TST   $ENV          ::RUNNING UNDER APT?
4580 035626 001413          BEQ   12$          ::IF NOT: BR
4581 035630 005737 001174          11$: TST   $MSGTYPE      ::FINISHED LAST MESSAGE?
4582 035634 001375          BNE   11$          ::IF NOT: WAIT
4583 035636 017637 000004 001176          MOV    @4(SP),$FATAL  ::GET ERROR #
4584 035644 062766 000002 000004          ADD    #2,4(SP)      ::BUMP RETURN ADDR.
4585 035652 005237 001174          INC   $MSGTYPE      ::TELL APT TO TAKE ERROR
4586 035656 105037 035702          12$: CLRB  $FFLG          ::CLEAR FATAL FLAG
4587 035662 105037 035701          CLRB  $LFLG         ::CLEAR LOG FLAG
4588 035666 105037 035700          CLRB  $MFLG         ::CLEAR MESSAGE FLAG
4589 035672 012601          MOV    (SP)+,R1      ::POP STACK INTO R1
4590 035674 012600          MOV    (SP)+,R0      ::POP STACK INTO R0
4591 035676 000207          RTS   PC            ::RETURN
4592 035700          000          $MFLG: .BYTE 0      ::MESSG. FLAG
4593 035701          000          $LFLG: .BYTE 0      ::LOG FLAG
4594 035702          000          $FFLG: .BYTE 0      ::FATAL FLAG
4595          035704          .EVEN
4596          000200          APTSIZE=200

```

```

4597      000001      APTENV=001
4598      000100      APTSPOOL=100
4599      000040      APTCSUP=040
4600      .SBTTL      POWER DOWN AND UP ROUTINES
4601
4602      ::*****
4603      :POWER DOWN ROUTINE
4604 035704 012737 036044 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST UP
4605 035712 012737 000340 000026      MOV    #340,@#PWRVEC+2  ;;PRIO:7
4606 035720 010046      MOV    R0,-(SP)        ;;PUSH R0 ON STACK
4607 035722 010146      MOV    R1,-(SP)        ;;PUSH R1 ON STACK
4608 035724 010246      MOV    R2,-(SP)        ;;PUSH R2 ON STACK
4609 035726 010346      MOV    R3,-(SP)        ;;PUSH R3 ON STACK
4610 035730 010446      MOV    R4,-(SP)        ;;PUSH R4 ON STACK
4611 035732 010546      MOV    R5,-(SP)        ;;PUSH R5 ON STACK
4612 035734 017746 143200      MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
4613 035740 010637 036050      MOV    SP,$SAVR6      ;;SAVE SP
4614 035744 012737 035756 000024      MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
4615 035752 000000      HALT
4616 035754 000776      BR     .-2            ;;HANG UP
4617
4618      ::*****
4619      :POWER UP ROUTINE
4620 035756 012737 036044 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
4621 035764 013706 036050      MOV    $SAVR6,SP      ;;GET SP
4622 035770 005037 036050      CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
4623 035774 005237 036050      1$:  INC    $SAVR6      ;;WAIT FOR THE INC
4624 036000 001375      BNE    1$            ;;OF WORD
4625 036002 012677 143132      MOV    (SP)+,@SWR     ;;POP STACK INTO @SWR
4626 036006 012605      MOV    (SP)+,R5      ;;POP STACK INTO R5
4627 036010 012604      MOV    (SP)+,R4      ;;POP STACK INTO R4
4628 036012 012603      MOV    (SP)+,R3      ;;POP STACK INTO R3
4629 036014 012602      MOV    (SP)+,R2      ;;POP STACK INTO R2
4630 036016 012601      MOV    (SP)+,R1      ;;POP STACK INTO R1
4631 036020 012600      MOV    (SP)+,R0      ;;POP STACK INTO R0
4632 036022 012737 035704 000024      MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
4633 036030 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
4634 036036 104401      TYPE      ;;REPORT THE POWER FAILURE
4635 036040 036052      $PWRMG: .WORD  $POWER  ;;POWER FAIL MESSAGE POINTER
4636 036042 000002      RTI
4637 036044 000000      $ILLUP: HALT
4638 036046 000776      BR     .-2            ;;THE POWER UP SEQUENCE WAS STARTED
4639 036050 000000      $SAVR6: 0            ;;BEFORE THE POWER DOWN WAS COMPLETE
4640 036052 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
4641 036060 000122
4642      .EVEN
    
```

```

4643      .SBTTL  ROUTINE TO SIZE MEMORY
4644
4645      ;*****
4646      ;*CALL:
4647      ;*   JSR    PC,$SIZE
4648      ;*   RETURN
4649      ;*$LSTAD WILL CONTAIN:
4650      ;*   WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
4651      ;*   WITHOUT KT11 OPTION   -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
4652      ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
4653      ;*$KT11 IS THE MEMORY MANAGEMENT KEY
4654      ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
4655      ;*   MUST BE SETUP BEFORE THE CALL
4656      ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
4657      ;*   DETERMINED BY ROUTINE
4658
4659      $SIZE:  MOV    R0,-(SP)      ;;SAVE R0 ON THE STACK
4660      MOV    R1,-(SP)      ;;SAVE R1 ON THE STACK
4661      MOV    R2,-(SP)      ;;SAVE R2 ON THE STACK
4662      MOV    R3,-(SP)      ;;SAVE R3 ON THE STACK
4663      MOV    @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
4664      MOV    @#ERRVEC+2,-(SP)
4665      MOV    SP,R0          ;;SAVE THE STACK POINTER
4666      ;;SET THE ERRVEC PS TO THE PRESENT PS
4667      TRAP
4668      MOV    (SP)+,@#ERRVEC+2  ;;PUSH OLD PSW AND PC ON STACK
4669      MOV    #3776,R1        ;;SAVE THE PSW IN @#ERRVEC+2
4670      TSTB  (PC)+          ;;SETUP ADDRESS
4671      TSTB  (PC)+          ;;USE MEMORY MANAGEMENT?
4672      $KT11: .WORD 200      ;;SET TO USE MEMORY MANAGEMENT
4673      BPL   $SCORE         ;;BR IF NO
4674      MOV   #KTNEX,@#ERRVEC  ;;SET FOR TIMEOUT
4675      TST  @#SRO           ;;KT11 ARE YOU THERE?
4676      BIS  #100000,$KT11    ;;YES--SET KT11 KEY
4677      MOV  #KTERR,@#MMVEC   ;;SET IN CASE OF ERROR
4678      MOV  #340,@#MMVEC+2
4679      CLR  -(SP)           ;;INITIALIZE FOR 'PAR' LOADING
4680      MOV  #KIPAR0,R2       ;;ADDRESS OF FIRST 'PAR'
4681      MOV  #^D8,R3         ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
4682      MOV  #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
4683      MOV  (SP),(R2)+      ;;LOAD 'PAR'
4684      ADD  #200,(SP)       ;;UPDATE FOR NEXT 'PAR'
4685      SOB  R3,1$          ;;LOOP UNTIL ALL EIGHT ARE LOADED
4686      MOV  #177600,-(R2)   ;;SETUP KIPAR7 FOR I/O
4687      CLR  -(R2)          ;;SETUP KIPAR6 FOR TESTING
4688      MOV  #2$,@#ERRVEC   ;;CATCH TIMEOUT IF NO SR3
4689      MOV  #20,@#SR3      ;;ENABLE 22 BIT MODE
4690      BR   3$            ;;THIS PDP-11 HAS A SR3 REGISTER
4691      2$:  CMP  (SP)+,(SP)+  ;;CLEAN OFF THE STACK--NO SR3
4692      3$:  INC  @#SRO       ;;TURN ON MEMORY MANAGEMENT
4693      MOV  #KTOUR,@#ERRVEC  ;;SET FOR TIME OUT
4694      4$:  TST  @#143776    ;;TRAP ON NON-EX-MEM
4695      ADD  #40,(R2)        ;;MAKE A 1K STEP
4696      CMP  @#KIPAR7,(R2)   ;;LAST ONE?
4697      BHI  4$            ;;NO--TRY IT
    
```

```

4697 036266 011202          $KTOUT: MOV      (R2),R2          ;;GET LAST BANK+1
4698 036270 005037 177572          CLR      @#SR0          ;;TURN OFF MEMORY MANAGEMENT
4699 036274 000421          BR        $SIZEX
4700 036276 042737 100000 036120 $KTNEX: BIC      #100000,$KT11 ;;KT11 NON-EXISTENT
4701 036304 012737 036334 000004 $CORE:  MOV      # $CROUT,@#ERRVEC ;;SET FOR TIMEOUT
4702 036312 005002          CLR      R2            ;;SET UP BANK
4703 036314 062701 004000          1$:      ADD      #4000,R1      ;;INCREMENT BY 1K
4704 036320 062702 000040          ADD      #40,R2         ;;1K STEP
4705 036324 005711          TST      (R1)          ;;TRAP ON TIME OUT
4706 036326 022701 177776          CMP      #177776,R1    ;;LAST ONE
4707 036332 001370          BNE      1$           ;;NO--TRY AGAIN
4708 036334 162701 004000          $CROUT: SUB      #4000,R1
4709 036340 162702 000040          $SIZEX: SUB      #40,R2          ;;DROP BACK
4710 036344 010006          MOV      R0,SP         ;;RESTORE THE STACK
4711 036346 012637 000006          MOV      (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
4712 036352 012637 000004          MOV      (SP)+,@#ERRVEC
4713 036356 010137 036400          MOV      R1,$LSTAD     ;;LAST ADDRESS
4714 036362 010237 036402          MOV      R2,$LSTBK     ;;LAST BANK
4715 036366 012603          MOV      (SP)+,R3      ;;RESTORE R3
4716 036370 012602          MOV      (SP)+,R2      ;;RESTORE R2
4717 036372 012601          MOV      (SP)+,R1      ;;RESTORE R1
4718 036374 012600          MOV      (SP)+,R0      ;;RESTORE R0
4719 036376 000207          RTS      PC
4720 036400 000000          $LSTAD: .WORD      0          ;;CONTAINS THE LAST ADDRESS
4721 036402 000000          $LSTBK: .WORD      0          ;;CONTAINS THE LAST BANK
4722 036404 000000          KTERR:  HALT
4723 036406 000776          BR        KTERR
  
```

```

4724      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
4725
4726      ::*****
4727      :*SAVE R0-R5
4728      :*CALL:
4729      :*      SAVREG
4730      :*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
4731      :*
4732      :*TOP---(+16)
4733      :* +2---(+18)
4734      :* +4---R5
4735      :* +6---R4
4736      :* +8---R3
4737      :*+10---R2
4738      :*+12---R1
4739      :*+14---R0
4740
4741      $SAVREG:
4742      036410 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
4743      036412 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
4744      036414 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
4745      036416 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
4746      036420 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
4747      036422 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
4748      036424 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
4749      036430 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
4750      036434 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
4751      036440 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
4752      036444 000002      RTI
4753
4754      :*RESTORE R0-R5
4755      :*CALL:
4756      :*      RESREG
4757      $RESREG:
4758      036446 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
4759      036452 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
4760      036456 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
4761      036462 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
4762      036466 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
4763      036470 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
4764      036472 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
4765      036474 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
4766      036476 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
4767      036500 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
4768      036502 000002      RTI

```

```

4769          .SBTTL  INTEGER MULTIPLY ROUTINE
4770
4771          :*****
4772          :*CALL
4773          :*   MOV    MULTIPLER,-(SP)
4774          :*   MOV    MULTIPLICAND,-(SP)
4775          :*   JSR    PC,@#$MULT
4776          :*   RETURN  ;;PRODUCT IS ON THE STACK
4777          :*
4778          :*   STACK  PRODUCT
4779          :*   -----
4780          :*   TOP    LSB'S
4781          :*   +2     MSB'S
4782
4783          $MULT:
4784          036504 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
4785          036506 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
4786          036510 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
4787          036512 005046      CLR    -(SP)         ;;CLEAR THE SIGN KEY
4788          036514 016601 000012  MOV    12(SP),R1     ;;GET THE MULTIPLICAND
4789          036520 100002      BPL    1$            ;;BR IF PLUS
4790          036522 005216      INC    (SP)         ;;SET THE SIGN KEY
4791          036524 005401      NEG    R1           ;;MAKE THE MULTIPLICAND POSTIVE
4792          036526 016602 000014  1$:  MOV    14(SP),R2     ;;GET THE MULTIPLIER
4793          036532 100002      BPL    2$            ;;BR IF PLUS
4794          036534 005316      DEC    (SP)         ;;UPDATE THE SIGN KEY
4795          036536 005402      NEG    R2           ;;MAKE THE MULTIPLIER POSTIVE
4796          036540 012746 000021  2$:  MOV    #17,-(SP)    ;;SET THE LOOP COUNT
4797          036544 005000      CLR    R0           ;;SETUP FOR THE MULTIPLY LOOP
4798          036546 103001  3$:  BCC    4$            ;;DON'T ADD IF MULTIPLICAND = 0
4799          036550 060200      ADD    R2,R0
4800          036552 006000  4$:  ROR    R0           ;;POSITION THE PARITIAL PRODUCT AND
4801          036554 006001      ROR    R1           ;;THE MULTIPLICAND
4802          036556 005316      DEC    (SP)         ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
4803          036560 001372      BNE    3$           ;;BR IF NO
4804          036562 022616      CMP    (SP)+,(SP)   ;;SHOULD PRODUCT BE NEGATIVE?
4805          036564 001403      BEQ    5$           ;;GO TO EXIT IF NO
4806          036566 005400      NEG    R0           ;;YES--SO MAKE IT SO
4807          036570 005401      NEG    R1
4808          036572 005600      SBC    R0
4809          036574 005726  5$:  TST    (SP)+        ;;CLEAR SIGN INFO. OFF OF STACK
4810          036576 010066 000012  MOV    R0,12(SP)    ;;PUT THE PRODUCT ON THE STACK (MSB'S)
4811          036602 010166 000010  MOV    R1,10(SP)    ;;LSB'S
4812          036606 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
4813          036610 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
4814          036612 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
4815          036614 000207      RTS    PC
  
```

4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869

036616
036616 104400
036620 042716 000017
036624 010046
036626 010146
036630 010246
036632 010346
036634 005046
036636 012746 000021
036642 016601 000024
036646 016600 000022
036652 100005
036654 105366 000003
036660 005400
036662 005401
036664 005600
036666 016602 000020
036672 002407
036674 003011
036676 052766 000003 000014
036704 012700 177777
036710 000424
036712 005266 000002
036716 000401
036720 005402
036722 000241
036724 000405
036726 006100
036730 010003

.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
*SAVE 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
    
```

```

$DIV:
TRAP                                ;;PUSH OLD PSW AND PC ON STACK
BIC #17,(SP)                        ;;STRIP AWAY CONDITION CODES
MOV R0,-(SP)                        ;;PUSH R0 ON STACK
MOV R1,-(SP)                        ;;PUSH R1 ON STACK
MOV R2,-(SP)                        ;;PUSH R2 ON STACK
MOV R3,-(SP)                        ;;PUSH R3 ON STACK
CLR -(SP)                           ;;SAVE A PLACE FOR SIGNS
MOV #17,-(SP)                       ;;SETUP THE ITERATION COUNTER
MOV 24(SP),R1                       ;;PICKUP THE DIVIDEND
MOV 22(SP),R0
BPL 1$                               ;;CHECK THE SIGN
DECB 3(SP)                          ;;KEEP TRACK OF THE SIGN
NEG R0                               ;;AND NEGATE THE ORIGINAL
NEG R1                               ;;NUMBER
SBC R0
1$: MOV 20(SP),R2                    ;;PICKUP THE DIVISOR
BLT 2$                               ;;CHECK THE SIGN
BGT 3$                               ;;DIVISOR OF 0 IS A NO-NO
BIS #3,14(SP)                       ;;SET 'V' & 'C'
MOV #-1,R0                          ;;SET REMAINDER TO ALL ONES
BR 7$                               ;;EXIT
2$: INC 2(SP)                       ;;KEEP TRACK OF DIVISORS SIGN
BR 4$
3$: NEG R2                          ;;NEGATE THE ORIGINAL NUMBER
4$: CLC                             ;;CLEAR 'C'
BR 6$                               ;;START FORMING QUOTIENT
5$: ROL R0                          ;;POSITION MSB'S
MOV R0,R3                          ;;COPY
    
```

4870	036732	060203			ADD	R2,R3	::COMPARE DIVIDEND & DIVISOR
4871	036734	103001			BCC	6\$::BR IF DIVIDEND > DIVISOR
4872	036736	010300			MOV	R3,R0	::REMAINDER AFTER THIS LOOP
4873	036740	006101		6\$:	ROL	R1	::QUOTIENT BIT ENTERS HERE
4874	036742	005316			DEC	(SP)	::DONE?
4875	036744	001370			BNE	5\$::BR IF NO
4876	036746	005701			TST	R1	::OVERFLOW?
4877	036750	100005			BPL	8\$::BR IF NO
4878	036752	052766	000002	000014	BIS	#2,14(SP)	::SET 'V' IN RETURN STATUS WORD
4879	036760	005000			CLR	R0	::SET REMAINDER TO ALL ZEROS
4880	036762	010001		7\$:	MOV	R0,R1	::COPY REMAINDER INTO QUOTIENT
4881	036764	005726		8\$:	TST	(SP)+	::CLEAR COUNTER FROM STACK
4882	036766	005716			TST	(SP)	::REMAINDER SIGN CORRECTION NEEDED?
4883	036770	002004			BGE	9\$::BR IF NO
4884	036772	005400			NEG	R0	::NEGATE REMAINDER
4885	036774	105066	000001		CLRB	1(SP)	::CLEAR SIGN
4886	037000	005316			DEC	(SP)	::BUT DON'T FORGET QUOTIENT
4887	037002	005726		9\$:	TST	(SP)+	::QUOTIENT SIGN CORRECTION NEEDED?
4888	037004	001401			BEQ	10\$::BR IF NO
4889	037006	005401			NEG	R1	::NEGATE QUOTIENT
4890	037010	010166	000020	10\$:	MOV	R1,20(SP)	::RETURN QUOTIENT AND
4891	037014	010066	000016		MOV	R0,16(SP)	::REMAINDER TO USER
4892	037020	012603			MOV	(SP)+,R3	::POP STACK INTO R3
4893	037022	012602			MOV	(SP)+,R2	::POP STACK INTO R2
4894	037024	012601			MOV	(SP)+,R1	::POP STACK INTO R1
4895	037026	012600			MOV	(SP)+,R0	::POP STACK INTO R0
4896	037030	012666	000002		MOV	(SP)+,2(SP)	::SETUP TO RETURN CONDITION CODES
4897	037034	000002			RTI		::RETURN

4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951

037036 104413
037040 016602 000002
037044 012700 037216
037050 010066 000002
037054 012201
037056 012202
037060 012737 000012 037134
037066 012704 037146
037072 012705 037150
037076 005003
037100 161401
037102 005602
037104 161502
037106 002402
037110 005203
037112 000772
037114 062401
037116 005502
037120 062402
037122 022525
037124 052703 000060
037130 110320
037132 005327
037134 000000
037136 001357
037140 105020
037142 104414
037144 000207
037146 145000
037150 035632
037152 160400
037154 002765
037156 113200
037160 000230
037162 041100
037164 000017
037166 103240
037170 000001
037172 023420
037174 000000
037176 001750

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

:*****
:*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
:*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
:*POSITIVE.
:*CALL
:*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
:*      JSR      PC,@#$DB2D      ;; THE FIRST ADDRESS OF ASCII
:*      RETURN                      ;; IS ON THE STACK

$DB2D: SAVREG                      ;; SAVE REGISTERS
      MOV      2(SP),R2          ;; PICKUP THE DATA POINTER
      MOV      #$DECVL,R0       ;; GET ADDRESS OF '$DECVL' STRING
      MOV      R0,2(SP)         ;; PUT ADDRESS OF ASCII STRING ON STACK
      MOV      (R2)+,R1         ;; PICKUP THE BINARY NUMBER
      MOV      (R2)+,R2
      MOV      #10.,4$          ;; SET UP TO DO 10 CONVERSIONS
      MOV      #$TNPWR,R4       ;; ADDRESS OF TEN POWER
      MOV      #$TNPWR+2,R5

1$:   CLR      R3                ;; CLEAR PARTIAL
2$:   SUB      (R4),R1           ;; SUBTRACT TEN POWER
      SBC      R2
      SUB      (R5),R2
      BLT      3$               ;; BR IF TEN POWER TOO LARGE
      INC      R3                ;; ADD 1 TO PARTIAL
      BR      2$                ;; LOOP
3$:   ADD      (R4)+,R1         ;; RESTORE SUBTRACTED VALUE
      ADC      R2
      ADD      (R4)+,R2
      CMP      (R5)+,(R5)+      ;; MOVE TO NEXT TEN POWER
      BIS      #'0,R3           ;; CHANGE PARTIAL TO ASCII
      MOV      R3,(R0)+         ;; SAVE IT
      DEC      (PC)+           ;; DONE?
4$:   .WORD    0
      BNE     1$                ;; BR IF NO
      CLRB   (R0)+             ;; TERMINATOR
      RESREG                      ;; RESTORE REGISTERS
      RTS    PC                ;; RETURN
$TNPWR: 145000                  ;; 1.0E09
        35632
        160400                  ;; 1.0E08
        2765
        113200                  ;; 1.0E07
        230
        041100                  ;; 1.0E06
        17
        103240                  ;; 1.0E05
        1
        23420                   ;; 1.0E04
        0
        1750                    ;; 1.0E03

```

```

4952 037200 000000          0
4953 037202 000144          144      ::1.0E02
4954 037204 000000          0
4955 037206 000012          12      ::1.0E01
4956 037210 000000          0
4957 037212 000001          1      ::1.0E00
4958 037214 000000          0
4959 037216 000014          0
$DECVL: .BLKB 12.          ::RESERVE STORAGE FOR ASCIZ STRING
.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971 037232 016637 000002 037262 $SB2D: MOV 2(SP),1$      ::SAVE BINARY NUMBER
4972 037240 012746 037262      MOV #1$,-(SP)      ::SET POINTER
4973 037244 004737 037036      JSR PC,@#$DB2D    ::CALL DOUBLE LENGTH CONVERT
4974 037250 062716 000005      ADD #5,(SP)      ::ONLY ALLOW FIVE CHARACTERS
4975 037254 012666 000002      MOV (SP)+,2(SP)  ::PICKUP POINTER
4976 037260 000207              RTS PC           ::RETURN
4977 037262 000000 000000      1$: .WORD 0,0

```

```

:*****
:*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
:*UNSIGNED DECIMAL ASCII NUMBER.

```

```

:*CALL
:*      MOV      NUMBER,-(SP)      ::PUT BINARY NUMBER ON THE STACK
:*      JSR      PC,@#$SB2D      ::CALL
:*      RETURN     ::ADDRESS OF THE 1ST ASCII CHAR.IS ON THE STACK

```

4978
4979
4980
4981
4982
4983
4984
4985
4986 037266 010046
4987 037270 016600 000002
4988 037274 005740
4989 037276 111000
4990 037300 006300
4991 037302 016000 037322
4992 037306 000200
4993
4994
4995
4996
4997 037310 011646
4998 037312 016666 000004 000002
4999 037320 000002
5000
5001
5002
5003
5004
5005
5006
5007
5008 037322 037310
5009 037324 034372
5010 037326 034700
5011 037330 034654
5012 037332 034714
5013 037334 033662
5014
5015 037336 033054
5016
5017 037340 033004
5018 037342 033266
5019 037344 033406
5020 037346 033560
5021 037350 036410
5022 037352 036446
5023

.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
```

```

5024 .SBTTL A TO D FIELD SITE AND ADJUSTMENT LOOP
5025 037354 004737 046042 FSITE: JSR PC,CTRLCG ;TEST FOR CTRL C OR G
5026 :CAMERA #0
5027 037360 032777 000100 141552 BIT #SW06,@SWR ;TEST IF CAM 0 G=1
5028 037366 001004 BNE 1$
5029 037370 004537 040102 JSR R5,CONVRT ;CONVERT
5030 037374 000000 0 ;CAMERA 0
5031 037376 040474 CAMOG1 ;STORE RESULTS
5032 037400 032777 000001 141532 1$: BIT #SW00,@SWR ;TEST IF CAM 0 G=2
5033 037406 001004 BNE 2$
5034 037410 004537 040102 JSR R5,CONVRT ;CONVERT
5035 037414 002000 BIT10 ;GAIN = 2
5036 037416 040476 CAMOG2
5037 :CAMERA #1
5038 037420 032777 000200 141512 2$: BIT #SW7,@SWR ;TEST IF CAM 1 G=1
5039 037426 001004 BNE 3$
5040 037430 004537 040102 JSR R5,CONVRT ;CONVERT
5041 037434 000400 BIT8 ;CAMERA 1
5042 037436 040500 CAM1G1 ;STORE RESULTS
5043 037440 032777 000002 141472 3$: BIT #SW01,@SWR ;TEST IF CAM 1 G=2
5044 037446 001004 BNE 4$
5045 037450 004537 040102 JSR R5,CONVRT ;CONVERT
5046 037454 002400 BIT10!BIT8 ;CAMERA 1 GAIN = 2
5047 037456 040502 CAM1G2 ;RESULTS
5048 :CAMERA #2
5049 037460 032777 000400 141452 4$: BIT #SW8,@SWR ;TEST IF CAM 2 G=1
5050 037466 001004 BNE 5$
5051 037470 004537 040102 JSR R5,CONVRT ;CONVERT
5052 037474 001000 BIT9 ;CAMERA #2
5053 037476 040504 CAM2G1 ;RESULTS
5054 037500 032777 000004 141432 5$: BIT #SW2,@SWR ;TEST IF CAM 2 G=2
5055 037506 001004 BNE 6$
5056 037510 004537 040102 JSR R5,CONVRT ;CONVERT
5057 037514 003000 BIT10!BIT9 ;GAIN = 2
5058 037516 040506 CAM2G2 ;RESULTS
5059 :CAMERA #3
5060 037520 032777 001000 141412 6$: BIT #SW9,@SWR ;TEST IF CAM 3 G=1
5061 037526 001004 BNE 7$
5062 037530 004537 040102 JSR R5,CONVRT ;CONVERT
5063 037534 001400 BIT9!BIT8 ;CAMERA #3
5064 037536 040510 CAM3G1 ;RESULTS
5065 037540 032777 000010 141372 7$: BIT #SW3,@SWR ;TEST IF CAM 3 G=2
5066 037546 001004 BNE 10$
5067 037550 004537 040102 JSR R5,CONVRT ;CONVERT
5068 037554 003400 BIT10!BIT9!BIT8 ;GAIN = 2
5069 037556 040512 CAM3G2 ;RESULTS
5070 :JOYSTICK
5071 037560 032777 000020 141352 10$: BIT #SW4,@SWR ;TEST IF JOYSTICK
5072 037566 001011 BNE CALRPT
5073 037570 052777 000001 142160 BIS #BIT0,@SFR ;ASK FOR JOYSTICK
5074 037576 105777 142154 11$: TSTB @SFR ;WAIT FOR JOY DONE
5075 037602 100375 BPL 11$
5076 037604 017737 142152 040514 MOV @JOY,JOYG1 ;SAVE THE RESULT

```

```

5077                                     ;NOW TEST IS TYPEOUT IS ENABLED
5078
5079 037612 004537 040060 CALRPT: JSR      R5,CKTSWR      ;TEST BIT 6
5080 037616 000100          BIT6
5081 037620 000405          BR        1$          ;BR IF SET
5082 037622 104401 040516 TYPE,    CAM0TX      ;REPORT CAMERA #0
5083 037626 004537 040412 JSR      R5,CAMUNP   ;REPORT VALUES
5084 037632 040474          CAM0G1
5085 037634 004537 040060 1$:      JSR      R5,CKTSWR      ;TEST BIT 0
5086 037640 000001          BIT0
5087 037642 000405          BR        2$
5088 037644 104401 040535 TYPE,    CAM0TW      ;REPORT G=2
5089 037650 004537 040412 JSR      R5,CAMUNP
5090 037654 040476          CAM0G2
5091
5092 037656 004537 040060 2$:      JSR      R5,CKTSWR      ;TEST BIT 7
5093 037662 000200          BIT7
5094 037664 000405          BR        3$          ;BR IF SET
5095 037666 104401 040554 TYPE,    CAM1TX      ;REPORT CAMERA #1
5096 037672 004537 040412 JSR      R5,CAMUNP   ;REPORT VALUES
5097 037676 040500          CAM1G1
5098 037700 004537 040060 3$:      JSR      R5,CKTSWR      ;TEST BIT 1
5099 037704 000002          BIT1
5100 037706 000405          BR        4$
5101 037710 104401 040573 TYPE,    CAM1TW      ;REPORT CAM 1 G=2
5102 037714 004537 040412 JSR      R5,CAMUNP
5103 037720 040502          CAM1G2
5104
5105 037722 004537 040060 4$:      JSR      R5,CKTSWR      ;TEST BIT 8
5106 037726 000400          BIT8
5107 037730 000405          BR        5$          ;BR IF SET
5108 037732 104401 040612 TYPE,    CAM2TX      ;REPORT CAMERA #2
5109 037736 004537 040412 JSR      R5,CAMUNP   ;REPORT VALUES
5110 037742 040504          CAM2G1
5111 037744 004537 040060 5$:      JSR      R5,CKTSWR      ;TEST BIT 2
5112 037750 000004          BIT2
5113 037752 000405          BR        6$
5114 037754 104401 040631 TYPE,    CAM2TW      ;REPORT CAMERA #2 G=2
5115 037760 004537 040412 JSR      R5,CAMUNP
5116 037764 040506          CAM2G2
5117
5118 037766 004537 040060 6$:      JSR      R5,CKTSWR      ;TEST BIT 9
5119 037772 001000          BIT9
5120 037774 000405          BR        7$          ;BR IF SET
5121 037776 104401 040650 TYPE,    CAM3TX      ;REPORT CAMERA #3
5122 040002 004537 040412 JSR      R5,CAMUNP   ;REPORT VLAUES
5123 040006 040510          CAM3G1
5124 040010 004537 040060 7$:      JSR      R5,CKTSWR      ;TEST BIT 3
5125 040014 000010          BIT3
5126 040016 000405          BR        10$
5127 040020 104401 040667 TYPE,    CAM3TW      ;REPORT CAMERA #3 G=2
5128 040024 004537 040412 JSR      R5,CAMUNP
5129 040030 040512          CAM3G2
5130

```

```
5131 040032 004537 040060      10$: JSR R5,CKTSWR      ;TEST BIT 4
5132 040036 000020                BIT4
5133 040040 000405                BR 11$
5134 040042 104401 040706        TYPE, JOYTW
5135 040046 004537 040412        JSR R5,CAMUNP      ;REPORT JOYSTICK
5136 040052 040514                JOYG1
5137 040054 000137 037354      11$: JMP FSITE
5138
5139 040060 032577 141054      CKTSWR: BIT (R5)+,@SWR      ;TEST IF INHIBIT THIS CAMERA
5140 040064 001005                BNE 1$              ;BR IF YES
5141 040066 032777 020000 141044 BIT #SW13,@SWR      ;TEST INHIBIT TYPE-OUT
5142 040074 001001                BNE 1$              ;BR IF YES
5143 040076 005725                TST (R5)+           ;BUMP EXIT POINTER
5144 040100 000205      1$: RTS R5          ;EXIT
```

```
5145 ;SUBROUTINE TO TAKE 8 CONVERSIONS AND AVERAGE FOUR OF THEM
5146 040102 012537 040406 CONVRT: MOV (R5)+,1$ ;SAVE CAMERA CHANNEL
5147 040106 012777 004000 141642 4$: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
5148 040114 013777 040406 141624 MOV 11$,@CSR ;SELECT CAMERA AND GAIN
5149 040122 012777 060000 141624 MOV #BUFO,@BAR ;LOAD BUS ADDRESS
5150 040130 012777 177770 141614 MOV #-10,@WCR ;LOAD WORD COUNT
5151 040136 000240 NOP
5152 040140 000240 NOP
5153 040142 000240 NOP
5154 040144 000240 NOP
5155 040146 013777 040410 141602 MOV INOUTZ,@SFR ;SET Z INPUTS IF ENABLED
5156 040154 000240 NOP
5157 040156 000240 NOP
5158 040160 012700 000004 MOV #4.,R0 ;LOAD COUNTER
5159 040164 005001 CLR R1 ;FOR OUT OF RANGE VOLTAGE
5160 040166 000240 NOP
5161 040170 052777 000001 141550 BIS #BIT0,@CSR ;ENABLE THE DEVICE
5162 040176 105777 141544 1$: TSTB @CSR ;WAIT FOR IDLE
5163 040202 100430 BMI 3$
5164 040204 005301 DEC R1 ;DELAY
5165 040206 001373 BNE 1$
5166 040210 004737 046042 JSR PC,CTRLCG ;TEST FOR CTRL G OR C
5167 040214 005300 DEC R0 ;FINISHED OUT OF RANGE TIMER
5168 040216 001367 BNE 1$
5169 040220 032777 020000 140712 BIT #SW13,@SWR ;TEST IF INHIBIT TYPEOUT
5170 040226 001010 BNE 5$ ;BR IF INHIBIT
5171 040230 113700 040407 MOVB 11$+1,R0 ;GET CAMERA #
5172 040234 062700 000060 ADD #60,R0 ;MAKE INTO ASCII
5173 040240 110037 032714 MOVB R0,OUTCHN ;INSERT INTO ASCII ERROR MESSAGE
5174 040244 104401 032621 TYPE, OUTRNG ;TELL OPERATOR
5175 040250 032777 100000 140662 5$: BIT #SW15,@SWR ;TEST IF HALT ON NO CONVERSION ERROR
5176 040256 001401 BEQ 6$ ;BR IF CLEARED
5177 040260 000000 HALT ;NO CONVERT FLAG - INPUT VOLTAGE WAS OUT OF RANGE OR NO 'Z' PULSES
5178 040262 000711 BR 4$
5179 040264 012700 060000 3$: MOV #BUFO,R0 ;LOAD POINTER
5180 040270 005003 CLR R3 ;CLEAR RESULT
5181 040272 005004 CLR R4
5182 040274 012737 000004 040404 2$: MOV #4,10$ ;LOAD COUNTER
5183 040302 112002 MOVB (R0)+,R2 ;GET LOW BYTE
5184 040304 112001 MOVB (R0)+,R1 ;GET HIGH BYTE
5185 040306 042702 177400 BIC #177400,R2
5186 040312 042701 177400 BIC #177400,R1
5187 040316 060104 ADD R1,R4 ;UPDATE RESULT
5188 040320 060203 ADD R2,R3 ;UPDATE RESULT
5189 040322 005337 040404 DEC 10$ ;FINISHED ?
5190 040326 001365 BNE 2$
5191 040330 006203 ASR R3
5192 040332 006203 ASR R3
5193 040334 000240 NOP
5194 040336 000240 NOP
5195 040340 006204 ASR R4
5196 040342 006204 ASR R4
5197 040344 000240 NOP
5198 040346 000240 NOP
```

```

5199 040350 000240      NOP
5200 040352 010437 040404  MOV      R4,10$      ;SAVE IT
5201 040356 105037 040405  CLR      10$+1      ;CLEAR HIGH BYTE
5202 040362 110337 040405  MOV      R3,10$+1   ;LOAD HIGH BYTE
5203 040366 013735 040404  MOV      10$,@(R5)+ ;SAVE THE RESULT
5204 040372 000240      NOP
5205 040374 000240      NOP
5206 040376 000240      NOP
5207 040400 000240      NOP
5208 040402 000205      RTS      R5
5209 040404 000000      10$: 0
5210 040406 000000      11$: 0
5211 040410 000002      INOUTZ: TESTZ      ;0= USE CUSTOMER Z PULSES, 1= MAINT Z PULSES
5212
5213      ;SUBROUTINE TO TYPE THE CAMERA MESSAGE
5214 040412 013537 040470      CAMUNP: MOV      @(R5)+,10$      ;GET RESULT
5215 040416 113737 040471 040472  MOV      10$+1,11$
5216 040424 105037 040471      CLR      10$+1
5217 040430 105037 040473      CLR      11$+1
5218 040434 013746 040470      MOV      10$,-(SP)
5219 040440 104403      TYPOS
5220 040442      003      001      .BYTE      3,1      ;TYPE 3 DIGIT NUMBER
5221 040444 104401 040724      TYPE,      G1B      ;AND TRALING ASCII
5222 040450 013746 040472      MOV      11$,-(SP)
5223 040454 104403      TYPOS
5224 040456      003      001      .BYTE      3,1      ;TYPE 3 DIGIT NUMBER
5225 040460 000240      NOP
5226 040462 000240      NOP
5227 040464 000240      NOP
5228 040466 000205      RTS      R5      ;EXIT
5229 040470 000000      10$: 0
5230 040472 000000      11$: 0
5231 040474 000000      CAMOG1: 0
5232 040476 000000      CAMOG2: 0
5233 040500 000000      CAM1G1: 0
5234 040502 000000      CAM1G2: 0
5235 040504 000000      CAM2G1: 0
5236 040506 000000      CAM2G2: 0
5237 040510 000000      CAM3G1: 0
5238 040512 000000      CAM3G2: 0
5239 040514 000000      JOYG1: 0
5240
5241 040516      015      012      CAMOTX: .BYTE      15,12
5242 040520 040503 030115 020060      .ASCIZ      /CAM00 G=1 X=/
5243 040526 036507 020061 036530
5244 040534      000
5245 040535      015      012      CAMOTW: .BYTE      15,12
5246 040537      103      046501 030060      .ASCIZ      /CAM00 G=2 X=/
5247 040544 043440 031075 054040
5248 040552 000075
5249 040554      015      012      CAM1TX: .BYTE      15,12
5250 040556 040503 030115 020061      .ASCIZ      /CAM01 G=1 X=/
5251 040564 036507 020061 036530
5252 040572      000
  
```


5253	040573	015	012			CAM1TW: .BYTE 15,12
5254	040575	103	046501	030460		.ASCIZ /CAM01 G=2 X=/
5255	040602	043440	031075	054040		
5256	040610	000075				
5257	040612	015	012			CAM2TX: .BYTE 15,12
5258	040614	040503	030115	020062		.ASCIZ /CAM02 G=1 X=/
5259	040622	036507	020061	036530		
5260	040630	000				
5261	040631	015	012			CAM2TW: .BYTE 15,12
5262	040633	103	046501	031060		.ASCIZ /CAM02 G=2 X=/
5263	040640	043440	031075	054040		
5264	040646	000075				
5265	040650	015	012			CAM3TX: .BYTE 15,12
5266	040652	040503	030115	020063		.ASCIZ /CAM03 G=1 X=/
5267	040660	036507	020061	036530		
5268	040666	000				
5269	040667	015	012			CAM3TW: .BYTE 15,12
5270	040671	103	046501	031460		.ASCIZ /CAM03 G=2 X=/
5271	040676	043440	031075	054040		
5272	040704	000075				
5273	040706					JOYTX:
5274	040706	015	012			JOYTW: .BYTE 15,12
5275	040710	047512	051531	044524		.ASCIZ /JOYSTICK X=/
5276	040716	045503	054040	000075		
5277	040724	054440	000075			G1B: .ASCIZ / Y=/
5278	040730	005015	044504	043106		GAIN: .ASCII <15><12>/DIFFERENTIAL LINEARITY:/
5279	040736	051105	047105	044524		
5280	040744	046101	046040	047111		
5281	040752	040505	044522	054524		
5282	040760	072				
5283	040761	040	020040	040507		GAINX: .ASCII / GAIN = /
5284	040766	047111	036440	040		
5285	040773	061	040	040		GAIN1: .BYTE 61,40,40,0
5286	040776	000				
5287	040777	040	026455	000040		DASH: .ASCIZ / -- /
5288	041004	046040	041123	005015		LSBMSG: .ASCIZ / LSB/<15><12>
5289	041012	000				
5290	041013	040	045523	050111		SKPMSG: .ASCIZ / SKIPPED STATE(S)/<15><12>
5291	041020	042520	020104	052123		
5292	041026	052101	024105	024523		
5293	041034	005015	000			
5294	041037	040	025052	051105		ERMSG: .ASCIZ / **ERROR**/<15><12>
5295	041044	047522	025122	006452		
5296	041052	000012				
5297	041054	020040	020040	047440		OKMSG: .ASCIZ / OK/<15><12>
5298	041062	006513	000012			
5299	041066	047040	051101	047522		NARMSG: .ASCIZ # NARROW STATE(S)#<15><12>
5300	041074	020127	052123	052101		
5301	041102	024105	024523	005015		
5302	041110	000				
5303	041111	040	044527	042504		WIDMSG: .ASCIZ # WIDE STATE(S)#<15><12>
5304	041116	051440	040524	042524		
5305	041124	051450	006451	000012		
5306	041132	052123	052101	026505		STATE: .ASCIZ /STATE-- WIDTH/<15><12>

5307	041140	020055	044527	052104	
5308	041146	006510	000012		
5309	041152	046040	041123	046440	LINEA: .ASCIZ / LSB MAXIMUS AT /
5310	041160	054101	046511	051525	
5311	041166	040440	020124	000	
5312	041173	057	000		SLASH: .ASCIZ #/#
5313	041175	122	046105	052101	MSG21: .ASCIZ /RELATIVE ACCURACY: /<15><12>
5314	041202	053111	020105	041501	
5315	041210	052503	040522	054503	
5316	041216	006472	000012		
5317	041222	005015	047522	020115	COMP: .ASCIZ <15><12>/ROM BLASTING COMPLETED/
5318	041230	046102	051501	044524	
5319	041236	043516	041440	046517	
5320	041244	046120	052105	042105	
5321	041252	000			
5322	041253	015	052012	046511	TIMEO: .ASCII <15><12>/TIMEOUT FROM BLASTER CHECK SWITCHES ON BLASTER/
5323	041260	047505	052125	043040	
5324	041266	047522	020115	046102	
5325	041274	051501	042524	020122	
5326	041302	020040	044103	041505	
5327	041310	020113	053523	052111	
5328	041316	044103	051505	047440	
5329	041324	020116	046102	051501	
5330	041332	042524	122		
5331	041335	015	020012	051117	.ASCIZ <15><12>/ OR REPLACE ROM IN BLASTER/<15><12>
5332	041342	051040	050105	040514	
5333	041350	042503	051040	046517	
5334	041356	044440	020116	046102	
5335	041364	051501	042524	006522	
5336	041372	000012			
5337	041374	015	012		FIELDI: .BYTE 15,12
5338	041376	047111	042524	047122	.ASCIZ /INTERNAL MAINT. Z PULSES <Y FOR YES> ? /
5339	041404	046101	046440	044501	
5340	041412	052116	020056	020132	
5341	041420	052520	051514	051505	
5342	041426	036040	020131	047506	
5343	041434	020122	042531	037123	
5344	041442	037440	000040		
5345	041446	005015	053101	051105	AVRGO: .ASCIZ <15><12>/AVERAGE OF 128 STATES = /
5346	041454	043501	020105	043117	
5347	041462	030440	034062	051440	
5348	041470	040524	042524	020123	
5349	041476	020075	000		
5350	041501	015	012	007	UNFIX: .BYTE 15,12,7
5351	041504	054105	042503	042105	.ASCIZ /EXCEEDED CORRECTION COUNTER FOR /
5352	041512	042105	041440	051117	
5353	041520	042522	052103	047511	
5354	041526	020116	047503	047125	
5355	041534	042524	020122	047506	
5356	041542	020122	000		
5357	041545	015	012	007	ABORTB: .BYTE 15,12,7
5358	041550	047522	020115	046102	.ASCII /ROM BLASTING ABORTED/
5359	041556	051501	044524	043516	
5360	041564	040440	047502	052122	

```

5361 041572 042105
5362 041574 015 012 007 .BYTE 15,12,7,0
5363 041577 000
5364 041600 020040 026455 020040 SGNVAL: .ASCII / -- /
5365 041606 053 040 SGNVL1: .BYTE 53,40
5366 041610 060 060 052 LSBTXT: .BYTE 60,60,52,60,15,12,0
5367 041613 060 015 012
5368 041616 000
5369 041617 015 012 000 RTN: .BYTE 15,12,0,0
5370 041622 000
5371 041623 040 042502 047514 BELMSG: .ASCIZ / BELOW LIMIT/<15><12>
5372 041630 020127 044514 044515
5373 041636 006524 000012
5374 041642 047040 051117 040515 NORMMSG: .ASCIZ / NORMAL STATES/<15><12>
5375 041650 020114 052123 052101
5376 041656 051505 005015 000
5377 041663 040 041101 053117 ABOMSG: .ASCIZ / ABOVE LIMIT/<15><12>
5378 041670 020105 044514 044515
5379 041676 006524 000012
5380 041702 050040 051501 042523 PASMSG: .ASCIZ / PASSED/<15><12>
5381 041710 006504 000012
5382 041714 043040 044501 042514 FAIMSG: .ASCIZ / FAILED/<15><12>
5383 041722 006504 000012
5384
5385 041726 000000 .EVEN
5386 041730 000000 LSBAVG: 0
5387 041732 000000 LSBSVQ: 0
5388 041734 000000 LSBSVR: 0
5389 041736 000000 LSBSVW: 0
5390 041740 042162 DIFEX1: 0
5391 041742 027340 ROMPNT: ROMVAL
NOMIAL: 12000. ;AVERAGE COUNT
5392
5393 ;THESE LOC.'S CLEARED BY DIFLIN
5394 041744 000000 AVGVAL: 0
5395 041746 000000 DIFERR: 0
5396 041750 000000 BELOW: 0
5397 041752 000000 ABOVE: 0
5398 041754 000000 NORMAL: 0
5399 041756 000000 NARROW: 0
5400 041760 000000 NARA: 0
5401 041762 000000 NARB: 0
5402 041764 000000 WIDA: 0
5403 041766 000000 WIDB: 0
5404 041770 000000 DIF: 0
5405 041772 000000 OUT: 0
5406 041774 000000 FIRST: 0
  
```

```

5407
5408 :KEYBOARD CONVERSATION TO THE BLASTER SELECT TEST 'T'
5409 :APPLY POWER TO BLASTER AND DEPRESS I/O AND EXECUTE BUTTONS
5410 :ALL INDICATORS SHOULD LIGHT THE BLASTER SHOULD SEND A '>' CHARACTER
5411 :TYPE THE COMMANDS FOLLOWED BY A 'CR', SOME WILL ECHO THE CR AND OTHERS
5412 :WILL NOT TYPE 'ESC'/'ALT' KEYS TO ABORT CURRNET COMMAND
5413 :TYPE KF2/2 :THIS SETS OCTAL COMMAND MODE
5414 :TYPE FM30 :THIS SET OCTAL INPUT/OUTPUT
5415 :TYPE SP[0037/8] :THIS SIZES THE PROM -- 0037/8 IS REPORTED BY THE BLASTER
5416 :TYPE LD :THIS READS THE PROM INTO THE RAM
5417 :TYPE D00/37 :THIS OUTPUTS THE RAM CONTENTS TO THE TTY
5418 :TYPE ZP :THIS EXITS COMPUTER CONTROL TO THE BUTTON MODE
5419
5420 041776 105777 137674 LOOPC: TSTB @DLICSR ;BLASTER INPUT ?
5421 042002 100007 BPL LOOPD ;NO
5422 042004 000240 NOP
5423 042006 000240 NOP
5424 042010 000240 NOP
5425 042012 017720 137662 MOV @DLIBD,(R0)+ ;SAVE THE CHARACTER
5426 042016 005237 042160 INC BLICNT ;UPDATE COUNTER
5427
5428 042022 105777 137122 LOOPD: TSTB @$TPS ;PRINTER READY ?
5429 042026 100363 BPL LOOPC ;BR IF NOT
5430 042030 005737 042160 TST BLICNT ;ANY CHARACTERS
5431 042034 001413 BEQ LOOPE ;BR IF NOT
5432 042036 012177 137110 MOV (R1)+,@$TPB ;PRINT THE CHAR.
5433 042042 005337 042160 DEC BLICNT
5434 042046 001353 BNE LOOPC ;BR IF MORE DATA
5435 042050 012700 060000 BTALK: MOV #BUFO,R0
5436 042054 010001 MOV R0,R1
5437 042056 005037 042160 CLR BLICNT
5438 042062 000745 BR LOOPC
5439
5440 042064 105777 137054 ;COME HERE IS NO BLASTER DATA TO BE TYPED
5441 042070 100342 LOOPE: TSTB @$TKS ;KEYBOARD INPUT
5442 042072 017702 137050 BPL LOOPC ;BR IF NOT
5443 042076 042702 177600 MOV @$TKB,R2 ;READ CHAR
5444 042102 022702 000003 BIC #177600,R2 ;MASK THE DATA
5445 042106 001413 CMP #3,R2 ;TEST FOR CTRL C
5446 042110 022702 000177 BEQ 2$ ;BR IF CTRL C
5447 042114 001412 CMP #177,R2 ;TEST FOR RUBOUT
5448 042116 020227 000140 BEQ LOOPF ;BR IF ESC
5449 042122 103402 CMP R2,#140 ;TEST IF LOWER CASE
5450 042124 042702 000040 BLO 1$ ;BR IF NOT
5451 042130 010277 137550 BIC #40,R2 ;MAKE UPPER CASE
5452 042134 000720 1$: MOV R2,@DL0DB ;LOAD BLASTER DATA OUT
5453 042136 000137 002644 BR LOOPC
5454 042142 012777 000033 137534 2$: JMP RBEGO ;JUMP TO RESTART
5455 042150 012777 000134 136774 LOOPF: MOV #33,@DL0DB ;LOAD ESC <CANCLE>
5456 042156 000707 MOV #' \,@$TPB
5457 042160 000000 BR LOOPC
BLICNT: 0

```

```

5458
5459           ;A017 CORRECTION PROM VALUES   <MEMORY RAM>
5460
5461 042162      200      200      200 ROMVAL: .BYTE 200,200,200,200,200,200,200,200 ;GAIN = 1 VALUE
5462 042165      200      200      200
5463 042170      200      200
5464 042172      200      200      200 .BYTE 200,200,200,200,200,200,200,200
5465 042175      200      200      200
5466 042200      200      200
5467 042202      200      200      200 .BYTE 200,200,200,200,200,200,200,200 ;GAIN = 2 VALUE
5468 042205      200      200      200
5469 042210      200      200
5470 042212      200      200      200 .BYTE 200,200,200,200,200,200,200,200
5471 042215      200      200      200
5472 042220      200      200
5473
5474           ;A017 CORRECTION PRAM VALUES   <RAM MEMORY>
5475           ;ROMVAL MUST BE IN COMPLEMENTED AND REVERSED ORDER <LSB=MSB>
5476
5477 042222      376      376      376 RAMVAL: .BYTE 376,376,376,376,376,376,376,376 ;GAIN = 1 VALUE
5478 042225      376      376      376
5479 042230      376      376
5480 042232      376      376      376 .BYTE 376,376,376,376,376,376,376,376
5481 042235      376      376      376
5482 042240      376      376
5483 042242      376      376      376 .BYTE 376,376,376,376,376,376,376,376 ;GAIN = 2 VALUE
5484 042245      376      376      376
5485 042250      376      376
5486 042252      376      376      376 .BYTE 376,376,376,376,376,376,376,376
5487 042255      376      376      376
5488 042260      376      376
5489
5490           ;M8036 PROGRAM PROM VALUES
5491
5492
5493 042262      215      244      116 PROROM: .BYTE 215,244,116,144,215,244,116,144 ;M8036 PROGRAM PROM
5494 042265      144      215      244
5495 042270      116      144
5496 042272      215      244      116 .BYTE 215,244,116,144,215,244,116,144
5497 042275      144      215      244
5498 042300      116      144
5499 042302      116      144      216 .BYTE 116,144,216,244,377,377,377,377
5500 042305      244      377      377
5501 042310      377      377
5502 042312      377      377      377 .BYTE 377,377,377,377,377,377,377,377
5503 042315      377      377      377
5504 042320      377      377
  
```

```

5505          ;BLASTER SECTION
5506 042322 104401 051046      BLAST: TYPE, PRIMO      ;TELL OPERATOR ABOUT THE CABLE
5507 042326 104401 051460      TYPE, PRIM5          ;AND SWITCHES
5508 042332 042777 000001 137336 BIC #BIT0,@DLICSR    ;ENSURE INIT. ROM CONTROL LINE
5509 042340 004537 043112      JSR R5,RDBLST        ;TELL THE BLASTER TO INITILIZE
5510 042344 052644              ESCP                  ;REALLY WE ARE WAITING FOR THE OPERATOR
5511 042346 004537 043112      JSR R5,RDBLST        ;TELL THE BLASTER THE KEYBOARD INPUT MODE
5512 042352 052646              KFO
5513 042354 004537 043112      JSR R5,RDBLST        ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
5514 042360 052655              FMO
5515          ;PRIME THE MEMORY RAM BUFFER
5516 042362 012700 042162      MOV #ROMVAL,RO      ;GET POINTER
5517 042366 112720 000200      1$: MOV #200,(R0)+   ;LOAD THE BUFFER
5518 042372 022700 042222      CMP #ROMVAL+40,RO  ;TEST IF DONE
5519 042376 001373              BNE 1$              ;BR IF BUFFER NOT PRIMED
5520 042400 012737 000036 046146 MOV #36,PRIME1      ;LOAD GAIN AND RESOL. VALUES
5521 042406 004537 046152      JSR R5,LIMITS      ;LOAD DIFLIN ERROR LIMIT VALUES
5522 042412 046216              G1LIM0             ;G1 LIMIT - USERS
5523 042414 046226              G1LIM1             ;G1 LIMIT - OPTION CHECKOUT
5524 042416 112737 000061 040773 MOV #1,GAIN1        ;LOAD GAIN TYPEOUT VALUE
5525 042424 012737 042163 041740 MOV #ROMVAL+1,ROMPNT ;LOAD RAM POINTER
5526 042432 004737 042476      JSR PC,4$          ;TEST THAT GAIN
5527 042436 012737 002036 046146 MOV #2036,PRIME1   ;LOAD GAIN AND RESOL. VALUES
5528 042444 004537 046152      JSR R5,LIMITS      ;LOAD DIFLIN ERROR LIMIT VALUES
5529 042450 046236              G2LIM0             ;G2 LIMIT - USERS
5530 042452 046246              G2LIM1             ;G2 LIMIT - OPTION CHECKOUT
5531 042454 112737 000062 040773 MOV #2,GAIN1        ;LOAD GAIN TYPEOUT VALUE
5532 042462 012737 042203 041740 MOV #ROMVAL+21,ROMPNT ;LOAD RAM POINTER
5533 042470 004737 042476      JSR PC,4$          ;TEST GAIN OF 2
5534 042474 000436              BR 10$
5535 042476 012737 000003 042646 4$: MOV #3,100$        ;LOAD LOOP COUNTER
5536 042504 012737 000001 041736 5$: MOV #1,DIFEX1      ;LOAD DIFLIN EXIT FLAG
5537 042512 012737 042530 046150 MOV #6$,DIFEXO     ;LOAD DIFLIN EXIT RETURN
5538 042520 004737 042650      JSR PC,ZAPRAM      ;ENSURE BLASTER RAM HAS BEEN LOADED WITH
5539          ;THE 'RAMVAL' VALUE
5540 042524 000137 044064      JMP DIFLNO         ;RUN 'DIFLIN'
5541 042530 004737 043422      6$: JSR PC,ADJFIX   ;ADJUST AND FIX THE CORRECTION WEIGHTS
5542 042534 005337 042646      DEC 100$          ;HAS THIS SECTION BEEN EXECITED N TIMES
5543 042540 001361              BNE 5$            ;BR IF NOT
5544 042542 005737 041746      TST DIFERR        ;TEST IF ERROR OCCURRED?
5545 042546 001410              BEQ 8$            ;BR IF NOT
5546 042550 104401 041501      7$: TYPE, UNFIX    ;TELL THE OPERATOR
5547 042554 104401 040761      TYPE, GAINX       ;ABOUT IT
5548 042560 104401 041545      TYPE, ABORTB      ;TELL OPER. THE BLASTING IS ABORTED
5549 042564 000137 002644      JMP RBEGO         ;RESTART
5550 042570 000207      8$: RTS PC       ;EXIT

```

```

5551          ;THE RAM DATA IS NOW CORRECT - BLAST THE BLANK ROM
5552 042572 004537 043112 10$: JSR R5,RDBLST ;TELL THE BLASTER TO COMP. THE DATA
5553 042576 053075      CMO
5554 042600 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER TO CHECK-SUM DATA
5555 042604 053105      CCO
5556 042606 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER TO BLAST THE ROM
5557 042612 053110      PGO
5558 042614 005237 043274 INC NOEXIT ;SET FLAG TO EXIT WITHOUT RECPT OF '>' CHAR FROM BLASTER
5559 042620 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
5560 042624 053120      ZPO
5561          ;INFORM THE OPERATOR ABOUT THE BLASTER ROM
5562 042626 104401 052051 TYPE, PRIM1 ;TELL OPERATOR TO REMOVE RAM CABLE AND INSTERT ROM INTO A017
5563 042632 104411      RDLIN
5564 042634 012600      MOV (SP)+,R0 ;CLEAN STACK
5565 042636 104401 041222 TYPE, COMP ;COMPLETED
5566 042642 000137 002644 JMP RBEGO
5567 042646 000000 100$: 0
5568
5569          ;SUBROUTINE TO CONVERT THE MEMORY RAM VALUE INTO RAM MEMORY VALUE
5570
5571 042650 012703 042222 ZAPRAM: MOV #RAMVAL,R3 ;LOAD OUTPUT POINTER
5572 042654 012700 042162 MOV #ROMVAL,R0 ;LOAD INPUT POINTER
5573 042660 112001 1$: MOV B (R0)+,R1 ;GET A BYTE
5574 042662 105101 COMB R1 ;INVERT VALUE
5575 042664 005002 CLR R2 ;CLEAR RESULT
5576 042666 006001 ROR R1 ;MOVE INTO CARRY BIT
5577 042670 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5578 042672 006001 ROR R1 ;MOVE INTO CARRY BIT
5579 042674 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5580 042676 006001 ROR R1 ;MOVE INTO CARRY BIT
5581 042700 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5582 042702 006001 ROR R1 ;MOVE INTO CARRY BIT
5583 042704 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5584 042706 006001 ROR R1 ;MOVE INTO CARRY BIT
5585 042710 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5586 042712 006001 ROR R1 ;MOVE INTO CARRY BIT
5587 042714 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5588 042716 006001 ROR R1 ;MOVE INTO CARRY BIT
5589 042720 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5590 042722 006001 ROR R1 ;MOVE INTO CARRY BIT
5591 042724 006102 ROL R2 ;MOVE CARRY INTO BIT 0
5592 042726 110223 MOV B R2,(R3)+ ;SAVE OUTPUT BYTE
5593 042730 020027 042222 CMP R0,#ROMVAL+40 ;TEST IF END
5594 042734 001351 BNE 1$ ;BR IF NOT
5595          ;SUBROUTINE TO LOAD MEMORY RAM INTO RAM MEMORY
5596 042736 004537 043276 JSR R5,CONROM ;CONVERT 'ROMVAL' TABLE INTO BLASTER FORMAT
5597 042742 042222 RAMVAL
5598 042744 052777 000001 136724 BIS #BIT0,@DLICSR ;ENABLE RAM MEMORY TO BE LOADED
5599 042752 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER THE NEW RAM DATA
5600 042756 052663 DIO
5601 042760 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER TO CHECKSUM THE DATA
5602 042764 053105 CCO
5603 042766 004537 043112 JSR R5,RDBLST ;TELL THE BLASTER TO PROGRAM THE RAM-PAK
5604 042772 053110 PGO

```

5605	042774	042777	000001	136674	BIC	#BITO,@DLICSR	;RE-ENTER EMMULATE MODE
5606	043002	000207			RTS	PC	;EXIT


```

5607                ;BLASTING THE PROGRAM PROM (M8036)
5608
5609 043004 104401 051622          PBLAST: TYPE,   PRIM3      ;INFORM THE OPERATOR
5610 043010 042777 000001 136660 BIC      #BIT0,@DLICSR ;ENSURE ROM MODE
5611 043016 004537 043112          JSR      R5,RDBLST  ;TE L THE BLASTER TO INITILIZE
5612 043022 052644                      ESCP
5613 043024 004537 043112          JSR      R5,RDBLST  ;TE LL THE BLASTER THE KEYBOARD INPUT MODE
5614 043030 052646                      KFO
5615 043032 004537 043112          JSR      R5,RDBLST  ;TELL THE BLASTER THE DATA FORMAT <ASCII-OCTAL>
5616 043036 052655                      FMO
5617 043040 004537 043276          JSR      R5,CONROM  ;CONVERT THE 'PROROM' TABLE INTO BLASTER FORMAT
5618 043044 042262                      PROROM
5619 043046 004537 043112          JSR      R5,RDBLST  ;TELL THE BLASTER THE PROM DATA
5620 043052 052663                      DIO
5621 043054 004537 043112          JSR      R5,RDBLST  ;TELL THE BLASTER TO CHECKSUM THE DATA
5622 043060 053105                      CCO
5623 043062 004537 043112          JSR      R5,RDBLST  ;TELL THE BLASTER TO BLAST THE PROM
5624 043066 053110                      PGO
5625 043070 005237 043274          INC      NOEXIT    ;SET FLAG TO EXIT WITHOUT RECPT OF '>' FROM BLASTER
5626 043074 004537 043112          JSR      R5,RDBLST  ;TELL THE BLASTER TO RETURN TO KEYPAD MODE
5627 043100 053120                      ZPO
5628 043102 104401 041222          TYPE,   COMP      ;INFORM THE OPERATOR IT'S DONE
5629 043106 000137 002644          JMP      RBEGO

```

```

5630
5631
5632
5633
5634 043112 010046
5635 043114 012500
5636 043116 012737 000012 043272
5637 043124 111037 043264
5638 043130 017737 136544 043266
5639 043136 105777 136540 1$:
5640 043142 100375
5641 043144 105737 043264
5642 043150 001406
5643 043152 112037 043264
5644 043156 001403
5645 043160 113777 043264 136516
5646 043166 105777 136504 2$:
5647 043172 100416
5648 043174 005337 043270
5649 043200 001372
5650 043202 005337 043272
5651 043206 001367
5652 043210 005737 043274
5653 043214 001017
5654 043216 104401 041253
5655 043222 000137 002644
5656 043226 000240
5657 043230 017737 136444 043266 4$:
5658 043236 042737 177600 043266
5659 043244 022737 000076 043266
5660 043252 001331
5661 043254 012600 5$:
5662 043256 005037 043274
5663 043262 000205
5664
5665 043264 000000 10$: 0
5666 043266 000000 11$: 0
5667 043270 000000 12$: 0
5668 043272 000000 13$: 0
5669 043274 000000 NOEXIT: 0

;COME THERE TO OUTPUT A MESSAGE TO THE BLASTER
;EXIT ONLY WHEN THE BLASTER HAS ECHOED AN '>'
;REPORT AN ERROR UPON TIMEOUT UNLESS 'NOEXIT' IS NON-ZERO
RDBLST: MOV R0, -(SP) ;SAVE R0
MOV (R5)+, R0 ;GET ARG.
MOV #10, 13$ ;LOAD DELAY FOR ERROR COUNTER
MOVB (R0), 10$ ;LOAD A CHAR
MOV @DLIBD, 11$ ;FALSE READ TO CLEAR READY
1$: TSTB @DLOCSR ;WAIT FOR OUTPUT READY
BPL 1$
TSTB 10$ ;TEST IF FINISHED ?
BEQ 2$
MOVB (R0)+, 10$ ;GET A CHAR.
BEQ 2$ ;BR IF TERM
MOVB 10$, @DLODB ;OUTPUT THE CHARACTER
2$: TSTB @DLICSR ;WAIT FOR INPUT
BMI 4$ ;BR IF INPUT
DEC 12$ ;DELAY
BNE 2$
DEC 13$ ;DELAY
BNE 2$
TST NOEXIT ;TEST IF EXIT IS ALLOWED WITHOUT '>'
BNE 5$ ;BR IF YES
TYPE, TIMEO
JMP RBEGO
NOP
MOV @DLIBD, 11$ ;READ CHAR
BIC #177600, 11$ ;MASK OFF BITS
CMP #'>, 11$ ;TEST FOR >
BNE 1$ ;TRY NEXT CHAR.
5$: MOV (SP)+, R0 ;RESTORE R0
CLR NOEXIT
RTS R5 ;EXIT

```

```

5670
5671      ;CONVERT ROM DATA INTO PROM BLASTER FORMAT
5672 043276 012700 052676      CONROM: MOV      #DIDATA+3,R0      ;LOAD RESULT POINTER
5673 043302 012501              MOV      (R5)+,R1      ;LOAD INITIAL VALUE POINTER
5674 043304 010137 043360      MOV      R1,10$      ;COPY THE START
5675 043310 062737 000040 043360      ADD      #40,10$      ;MAKE THE END ADDRESS
5676 043316 111102              1$:  MOVB     (R1),R2      ;GET A BYTE OF DATA
5677 043320 004737 043402      JSR      PC,SHUF0     ;CONVERT IT TO ASCII
5678 043324 111102              MOVB     (R1),R2      ;GET SAME BYTE AGAIN
5679 043326 004737 043374      JSR      PC,SHUF3     ;CONVERT 2ND DIGIT
5680 043332 112102              MOVB     (R1)+,R2     ;GET MSD
5681 043334 004737 043362      JSR      PC,SHUF6     ;CONVERT IT
5682 043340 062700 000007      ADD      #7,R0       ;UPDATE RESULT POINTER
5683 043344 000240              NOP
5684 043346 000240              NOP
5685 043350 020137 043360      CMP      R1,10$      ;TEST IF DONE
5686 043354 001360              BNE     1$
5687 043356 000205              RTS     R5           ;EXIT
5688 043360 000000              10$:  0
5689
5690      ;ROTATE THE DATA RIGHT AND MAKE ASCII CHARACTER
5691
5692 043362 042702 177477      SHUF6:  BIC     #177477,R2      ;MASK OFF BITS
5693 043366 006202              ASR     R2
5694 043370 006202              ASR     R2
5695 043372 006202              ASR     R2
5696 043374 006202              SHUF3:  ASR     R2
5697 043376 006202              ASR     R2
5698 043400 006202              ASR     R2
5699 043402 042702 177770      SHUF0:  BIC     #177770,R2      ;MASK OFF BITS
5700 043406 062702 000060      ADD     #60,R2       ;MAKE ASCII
5701 043412 110240              MOVB    R2,-(R0)     ;LOAD RESULT BYTE
5702 043414 000240              NOP
5703 043416 000240              NOP
5704 043420 000207              RTS     PC           ;EXIT
5705

```

```

5706                                     ;SUBROUTINE TO ADJUST THE LSB VALUES
5707 043422 012700 062036 ADJFIX: MOV #BUF1+36,R0 ;GET BUFFER POINTER
5708 043426 013702 041740 MOV ROMPNT,R2 ;GET ROM POINTER
5709 043432 010204 MOV R2,R4
5710 043434 062704 000017 ADD #17,R4 ;MAKE END VALUE
5711 043440 012001 10$: MOV (R0)+,R1 ;GET LSB (17) M-1
5712 043442 061001 ADD (R0),R1 ;ADD LSB (00) M
5713 043444 163701 041744 SUB AVGVAL,R1 ;SUB 2 LSB AVG.
5714 043450 163701 041744 SUB AVGVAL,R1
5715 043454 010137 041734 MOV R1,LSBSVW ;SAVE FOR LATER USAGE
5716 043460 000240 NOP
5717 043462 000240 NOP
5718 043464 000240 NOP
5719 043466 000240 NOP
5720 043470 000240 NOP
5721 043472 000240 NOP
5722 043474 000240 NOP
5723 043476 000240 NOP
5724 043500 000240 NOP
5725 043502 000240 NOP
5726 043504 000240 NOP
5727 043506 000240 NOP
5728 043510 000240 NOP
5729 043512 000240 NOP
5730 043514 000240 NOP
5731 043516 000240 NOP
5732
5733 043520 010146 MOV R1,-(SP) ;SAVE ON STACK
5734 043522 012746 000316 MOV #316,-(SP) ;MULTI BY 316 (8)
5735 043526 004737 036504 JSR PC,$MULT
5736 043532 012637 043706 MOV (SP)+,100$ ;SAVE LSW
5737 043536 012637 043710 MOV (SP)+,101$ ;SAVE MSW
5738
5739 043542 013746 043706 MOV 100$,-(SP) ;
5740 043546 013746 043710 MOV 101$,-(SP) ;
5741 043552 013746 041744 MOV AVGVAL,-(SP)
5742 043556 004737 036616 JSR PC,$DIV
5743 043562 102004 BVC 1$ ;BR IF NO ERROR
5744 043564 000000 HALT
5745 043566 000000 HALT
5746 043570 000240 NOP
5747 043572 000240 NOP
5748 043574 012637 041732 1$: MOV (SP)+,LSBSVR ;SAVE INTGR. REMAINDER
5749 043600 012637 041730 MOV (SP)+,LSBSVQ ;SAVE QUOTIENT
5750
5751                                     ;NOW ADD THE QUOTIENT TO THE MEMORY RAM VALUE
5752
5753 043604 111203 MOVB (R2),R3 ;GET CURRENT MEMORY RAM VALUE
5754 043606 000240 NOP
5755 043610 042703 177400 BIC #177400,R3 ;CLEAR OFF UPPER BITS
5756 043614 000240 NOP
5757 043616 000240 NOP
5758 043620 000240 NOP
5759 043622 000240 NOP

```

```

5760 043624 163703 041730      SUB    LSBSVQ,R3      ;ADD THE QUOTIENT
5761 043630 000240              NOP
5762 043632 000240              NOP
5763 043634 000240              NOP
5764 043636 000240              NOP
5765 043640 000240              NOP
5766 043642 110322              MOVB   R3,(R2)+      ;RELOAD MEMORY RAM VALUE
5767
5768                               ;NOW UPDATE 'M' VALUE POINTER
5769
5770 043644 062700 000036      ADD    #36,R0        ;UPDATE POINTER
5771 043650 011001              MOV    (R0),R1      ;GET VALUE
5772 043652 000240              NOP
5773 043654 000240              NOP
5774 043656 000240              NOP
5775 043660 000240              NOP
5776 043662 063701 041734      ADD    LSBSVW,R1    ;CORRECT THE VALUE (#10)
5777 043666 000240              NOP
5778 043670 000240              NOP
5779 043672 000240              NOP
5780 043674 000240              NOP
5781 043676 010110              MOV    R1,(R0)      ;RESTORE FOR NEXT USAGE
5782 043700 020204              CMP    R2,R4        ;TEST WHEN FINISHED
5783 043702 001256              BNE   10$           ;BR UNTIL DONE
5784
5785 043704 000207              RTS   PC            ;EXIT
5786
5787 043706 000000      100$: 0
5788 043710 000000      101$: 0
  
```

```

5789
5790
5791 043712 005037 041736
5792 043716 112737 000061 040773
5793 043724 105737 001134
5794 043730 001007
5795 043732 104401 040730
5796 043736 005737 043712
5797 043742 100402
5798 043744 104401 051046
5799 043750 012737 000036 046146 10$:
5800 043756 004537 046152
5801 043762 046216
5802 043764 046226
5803 043766 012737 043776 046150
5804 043774 000433
5805 043776 112737 000062 040773 1$:
5806 044004 012737 002036 046146
5807 044012 004537 046152
5808 044016 046236
5809 044020 046246
5810 044022 012737 044044 046150
5811 044030 105737 001134
5812 044034 001002
5813 044036 104401 040730
5814 044042 000410
5815 044044 022737 177777 050106 11$:
5816 044052 001002 2$:
5817 044054 000137 030076
5818 044060 000137 002644 3$:
5819
5820
5821
5822 044064 012700 041744
5823 044070 005020
5824 044072 022700 041776
5825 044076 001374
5826 044100 012700 060000
5827 044104 005020 1$:
5828 044106 022700 064000
5829 044112 001374
5830
5831 044114 004737 045774
5832
5833 044120 012777 000000 135616
5834 044126 012777 000025 135604
5835
5836 044134 004737 045640
5837 044140 022702 000000 2$:
5838 044144 001773
5839 044146 105077 135570
5840 044152 152777 000002 135562
5841 044160 017737 135560 046032
5842

::DIFFERENTIAL LINEARITY
DIFLIN: CLR DIFEX1 ;CLEAR EXIT FLAG
          MOV #1,GAIN1 ;LOAD GAIN MESSAGE VALUE
          TSTB $AUTOB ;TEST IF UNDER MONITOR
          BNE 10$ ;BR IF YES
          TYPE, GAIN
          TST DIFLIN ;TEST BIT 15
          BMI 10$ ;DONT TELL OPER. ABOUT A017
          TYPE, PRIMO ;TELL OPERATOR ABOUT A017
          MOV #36,PRIME1 ;LOAD RESOLUTION AND GAIN
          JSR R5,LIMITS ;LOAD DIFLIN TOLERANCE
          G1LIMO ;G1 USER LIMIT
          G1LIM1 ;G1 OPTION AREA
          MOV #1$,DIFEXO ;LOAD RETURN ADDRESS
          BR DIFLN0
          MOV #2,GAIN1 ;LOAD GAIN MESSAGE VALUE
          MOV #2036,PRIME1 ;LOAD RESOLUTION AND GAIN
          JSR R5,LIMITS ;LOAD DIFLIN TOLERANCE
          G2LIMO ;G2 USER LIMIT
          G2LIM1 ;G2 OPTION AREA
          MOV #2$,DIFEXO ;LOAD RETURN ADDRESS
          TSTB $AUTOB ;RUNNING UNDER MONITOR
          BNE 11$ ;BR IF YES
          TYPE, GAIN
          BR DIFLN0
          CMP #-1,RUNDIF ;ENTER VIA 'F' SELECTION ?
          BNE 3$ ;BR IF NOT
          JMP $EOP ;YES REPORT END OF PASS
          JMP RBEGO

:DIFLIN ROUTINE
DIFLN0: MOV #AVGVAL,R0 ;LOAD CLEARING POINTER
DIFLN1: CLR (R0)+
          CMP #FIRST+2,R0 ;FINISHED ?
          BNE DIFLN2 ;BR IF NOT DONE
          MOV #BUF0,R0 ;LOAD BUFFER POINTER
          CLR (R0)+ ;CLEAR THE DATA BUFFER
          CMP #BUF2,R0 ;TEST IF FINISHED
          BNE 1$ ;BR IF NOT

:LOOK FOR THE DATA REGION FOR TO
          JSR PC,STDAT0 ;FIND THE NON-ZERO TO ZERO TRANS. DATA
          ;DATA HAS NOW GONE INTO THE GOOD 0 DATA REGION
          MOV #0,@KWPSR ;INIT THE CLOCK PRESET
          MOV #25,@KWCSR ;ENABLE THE CLOCK
          ;NOW TIME 'T1' LENGTH
          JSR PC,LISTDT ;GET CURRENT DATA VALUE
          CMP #0,R2 ;TEST FOR EXIT OF 0 DATA REGION
          BEQ 2$ ;BR UNTILL DONE
          CLRB @KWCSR1
          BISB #BIT1,@KWCSR1 ;FIRE ST2
          MOV @KWPSR,DIFT1 ;SAVE COUNTER VALUE
          ;NOW TIME 'T2' LENGTH

```

```

5843 044166 004737 045640      3$: JSR PC,LISTDT      ;GET CURRENT DATA VALUE
5844 044172 022702 000377      CMP #377,R2          ;TEST FOR ENTRY INTO MAX DATA REGION
5845 044176 001373              BNE 3$              ;BR IF NOT
5846 044200 105077 135536      CLR @KWCSR1
5847 044204 152777 000002 135530  BIS #BIT1,@KWCSR1   ;FIRE ST2
5848 044212 017737 135526 046034  MOV @KWPSR,DIFT2    ;SAVE COUNTER VALUE
5849                                     ;NOW TIME 'T3' LENGTH
5850 044220 004737 045640      4$: JSR PC,LISTDT      ;GET CURRENT DATA VALUE
5851 044224 022702 000377      CMP #377,R2          ;TEST FOR EXIT OF MAX DATA REGION
5852 044230 001773              BEQ 4$              ;BR IF NOT
5853 044232 105077 135504      CLR @KWCSR1
5854 044236 152777 000002 135476  BIS #BIT1,@KWCSR1   ;FIRE ST2
5855 044244 017737 135474 046036  MOV @KWPSR,DIFT3    ;SAVE COUNTER VALUE
5856 044252 005077 135462      CLR @KWCSR          ;STOP CLOCK
5857                                     ;NOW DETERMINE THE AVG TIME (T3+T2)/2 AND SAVE IN 'HT2T3'
5858 044256 013700 046034      MOV DIFT2,R0        ;GET T2 VALUE
5859 044262 063700 046036      ADD DIFT3,R0        ;ADD T3 VALUE
5860 044266 006000              ROR R0              ;/2
5861 044270 010037 046040      MOV R0,HT2T3       ;SAVE FOR LATER USE
5862                                     ;NOW RETURN TO LIST MODE AND LOOK FOR THE NON-ZERO TO ZERO EDGE AGAIN
5863 044274 004737 045774      DIFLN1: JSR PC,STDATO ;FIND THE EDGE
5864                                     ;NOW START THE CLOCK USING THE AVG TIME
5865                                     ;AND DO AN MATRIX MODE COLLECTION
5866 044300 013700 046040      MOV HT2T3,R0        ;GET AVG CLOCK
5867 044304 005400              NEG R0              ;FIX FOR CLOCK PRESET REG
5868 044306 010077 135432      MOV R0,@KWPSR      ;LOAD CLOCK PRESET
5869 044312 012777 000020 135420  MOV #20,@KWCSR     ;PRIME CLOCK
5870 044320 012777 004000 135430  MOV #CLRALL,@SFR   ;INIT THE NCV11
5871 044326 012777 060000 135414  MOV #BUFO,@OFF     ;LOAD OFFSET
5872 044334 005077 135412      CLR @WCR           ;CLEAR Z COUNTER
5873 044340 005077 135410      CLR @BAR
5874 044344 013777 046146 135374  MOV PRIME1,@CSR    ;SET MODE AND RES.
5875 044352 052777 000022 135376  BIS #TESTZ!BIT4,@SFR ;SET TESTZ AND TESTX
5876 044360 052777 000001 135360  BIS #BIT0,@CSR     ;ENABLE NCV11
5877 044366 052777 000001 135344  BIS #BIT0,@KWCSR   ;ENABLE CLOCK
5878 044374 105777 135340      1$: TSTB @KWCSR     ;WAIT FOR CLOCK FLAG
5879 044400 100375              BPL 1$
5880 044402 052777 000400 135346  BIS #ENDDMA,@SFR   ;STOP DMA XFR

```

```

5881      ;THE MATRIX MODE XFR IF NOW COMPLETED
5882      ;NOW ADD THE CURRENT BYTE DATA COLLECTION INTO A TOTAL INCREMENT BUFFER
5883 044410 005037 050100      CLR      $TEMP1
5884 044414 012700 060000      MOV      #BUF0,R0
5885 044420 012701 062000      MOV      #BUF1,R1
5886
5887 044424 111037 050100      2$:      MOVB      (R0),$TEMP1      ;GET THE WORD
5888 044430 105020                CLR      (R0)+      ;CLEAR THE BYTE TABLE ENTRY
5889 044432 063721 050100      ADD      $TEMP1,(R1)+      ;UPDATE THE LIST
5890 044436 103003                BCC      3$          ;BR IF NO OVERFLOW
5891 044440 005741                TST      -(R1)       ;POSITION POINTER
5892 044442 012721 177777      MOV      #-1,(R1)+      ;SET TO 177777
5893 044446 022701 064000      3$:      CMP      #BUF2,R1      ;FINISHED?
5894 044452 001364                BNE      2$          ;BR IF NOT
5895      ;GET THE AVERAGE OF THE 128 CENTER STATES
5896 044454 012700 062100      MOV      #BUF1+64.,R0    ;GET INITIAL POINTER
5897 044460 005001                CLR      R1
5898 044462 005002                CLR      R2
5899 044464 062001                4$:      ADD      (R0)+,R1      ;GET A VALUE
5900 044466 103001                BCC      5$          ;BR IF NO CARRY OVERFLOW
5901 044470 005202                INC      R2          ;UPDATE MSW
5902 044472 022700 062300      5$:      CMP      #BUF1+192.,R0  ;FINISHED THE BUFFER
5903 044476 001372                BNE      4$          ;BR IF NOT DONE AVERAGE
5904 044500 000241                CLC
5905 044502 006002                ROR      R2          ;ROTATE MSW
5906 044504 006001                ROR      R1          ;INTO LSW
5907 044506 000241                CLC
5908 044510 006002                ROR      R2          ;ROTATE MSW
5909 044512 006001                ROR      R1          ;INTO LSW
5910 044514 000241                CLC
5911 044516 006002                ROR      R2          ;ROTATE MSW
5912 044520 006001                ROR      R1          ;INTO LSW
5913 044522 000241                CLC
5914 044524 006002                ROR      R2          ;ROTATE MSW
5915 044526 006001                ROR      R1          ;INTO LSW
5916 044530 000241                CLC
5917 044532 006002                ROR      R2          ;ROTATE MSW
5918 044534 006001                ROR      R1          ;INTO LSW
5919 044536 000241                CLC
5920 044540 006002                ROR      R2          ;ROTATE MSW
5921 044542 006001                ROR      R1          ;INTO LSW
5922 044544 010137 041744      MOV      R1,AVGVAL      ;SAVE THE AVERAGE
5923 044550 004737 046042      JSR      PC,CTRLCG      ;TEST FOR ^C OR ^G
5924 044554 023737 041744 041742  CMP      AVGVAL,NOMIAL  ;TEST IF AVG. IF >
5925 044562 103644                BLO      DIFLN1       ;BR IF NOT
5926

```



```
5927 ;READ THE TOTAL INCREMENT BUFFER AND DETERMINE IF ANY VALUES OUT OF RANGE
5928
5929 044564 013737 041744 045260 READ1: MOV AVGVAL,101$ ;GET AVERAGE
5930 044572 012700 000001 MOV #1,R0
5931 044576 012701 062002 MOV #BUF1+2,R1
5932 044602 012137 045256 1$: MOV (R1)+,100$ ;GET A WORD
5933 044606 163737 045260 045256 SUB 101$,100$ ;REMOVE THE AVERAGE
5934 044614 100006 BPL 2$
5935 044616 005137 045256 COM 100$
5936 044622 112737 000055 041606 MOVB #'-,SGNVL1 ;INSERT '-' SIGN
5937 044630 000403 BR 3$
5938 044632 112737 000053 041606 2$: MOVB #'+,SGNVL1 ;INSERT '+' SIGN
5939 044640 013746 045256 3$: MOV 100$,-(SP)
5940 044644 012746 001750 MOV #1000.,-(SP)
5941 044650 004737 036504 JSR PC,$MULT ;MULTIPLY NUM-AVG BY 1000.
5942 044654 012637 045274 MOV (SP)+,107$ ;GET RESULT
5943 044660 012637 045276 MOV (SP)+,110$
5944 044664 013746 045274 MOV 107$,-(SP) ;DIVIDE RESULT
5945 044670 013746 045276 MOV 110$,-(SP)
5946 044674 013746 045260 MOV 101$,-(SP) ;DIVIDE AGAIN
5947 044700 004737 036616 JSR PC,$DIV
5948 044704 102003 BVC 4$
5949 044706 000000 HALT ;FATAL ERROR DURING CAL. OF ERROR TOLERANCES
5950 044710 000240 NOP
5951 044712 000240 NOP
5952 044714 012637 045266 4$: MOV (SP)+,104$ ;GET REMAINDER
5953 044720 012637 045270 MOV (SP)+,105$ ;GET QOUT
5954 044724 013702 045260 MOV 101$,R2 ;ROUND UP IF NEEDED
5955 044730 006202 ASR R2
5956 044732 023702 045266 CMP 104$,R2 ;COMPARE RESULT
5957 044736 002402 BLT 5$
5958 044740 005237 045270 INC 105$ ;ROUND UP
5959 044744 000240 5$: NOP
5960 ;DIFLIN ERROR CHECK
5961
5962 044746 123737 045270 046207 6$: CMPB 105$,NORTOL ;TEST AGAINST NORMAL
5963 044754 103446 BLO 50$ ;BR IF OK
5964 044756 123737 045270 046211 CMPB 105$,NARTOL ;TEST AGAINST WIDE/NARROW TOLERANCE
5965 044764 103416 BLO 20$ ;BR IF OK
5966 044766 123737 045270 046213 CMPB 105$,NURTOL ;TEST AGAINST LARGE/SMALL TOLERANCE
5967 044774 103424 BLO 21$ ;BR IF OK
5968 044776 122737 000053 041606 CMPB #'+,SGNVL1 ;TEST IF +
5969 045004 001403 BEQ 7$
5970 045006 005237 041750 INC BELOW ;COUNT THE LOWER VALUE
5971 045012 000435 BR 51$
5972 045014 005237 041752 7$: INC ABOVE ;COUNT THE HIGHER VALUE
5973 045020 000432 BR 51$
5974 045022 122737 000053 041606 20$: CMPB #'+,SGNVL1 ;TEST IF +
5975 045030 001403 BEQ 22$
5976 045032 005237 041762 INC NARB ;COUNT THE LOWER
5977 045036 000423 BR 51$
5978 045040 005237 041766 22$: INC WIDB ;COUNT THE HIGHER
5979 045044 000420 BR 51$
5980 045046 122737 000053 041606 21$: CMPB #'+,SGNVL1 ;TEST IF +
```

```

5981 045054 001403          BEQ      23$
5982 045056 005237 041760    INC      NARA          ;COUNT THE LOWER
5983 045062 000411          BR       51$
5984 045064 005237 041764    23$:    INC      WIDA          ;COUNT THE HIGHER
5985 045070 000406          BR       51$
5986 045072 005237 041754    50$:    INC      NORMAL        ;COUNT THE NORMAL
5987 045076 122737 000115 052474  CMPB    #'M,RUNIT      ;TEST IF FORCE REPORT
5988 045104 001054          BNE     77$
5989 045106 105737 001134    51$:    TSTB    $AUTOB        ;TEST IF MONITOR ?
5990 045112 001051          BNE     77$          ;BR IF YES
5991          ;REPORT THE STATE INFORMATION
5992 045114 005737 041774    TST     FIRST          ;FIRST TYPEOUT ?
5993 045120 001013          BNE     10$          ;BR IF YES
5994 045122 005237 041774    INC     FIRST          ;SET FLAG
5995 045126 104401 041446    TYPE    ,AVRG0        ;TELL THE OPERATOR THE AVERG.
5996 045132 013746 041744    MOV     AVGVAL,-(SP)
5997 045136 104405          TYPDS
5998 045140 104401 041617    TYPE,   RTN
5999 045144 104401 041132    TYPE,   STATE        ;ADD HEADER INFO
6000 045150 010046    10$:    MOV     RO,-(SP)      ;LOAD STATE #
6001 045152 104403          TYPOS
6002 045154 003 001          .BYTE  3,1
6003 045156 122737 000115 052474  CMPB    #'M,RUNIT      ;TEST IF EXPAND OUTPUT SELECTED
6004 045164 001005          BNE     11$
6005 045166 104401 040777    TYPE,   DASH          ;INSERT SPACING
6006 045172 013746 045256    MOV     100$,-(SP)    ;REPORT DIFFERENCE
6007 045176 104405          TYPDS
6008 045200 013746 045270    11$:    MOV     105$,-(SP)
6009 045204 004737 037232    JSR     PC,$SB2D      ;CONVERT TO ASCII
6010 045210 012602          MOV     (SP)+,R2      ;GET RESULT POINTER TO MESSAGE
6011 045212 062702 000002    ADD     #2,R2         ;ADJUST POINTER
6012 045216 112237 041610    MOVB   (R2)+,LSBTXT
6013 045222 112237 041611    MOVB   (R2)+,LSBTXT+1
6014 045226 112237 041613    MOVB   (R2)+,LSBTXT+3
6015 045232 104401 041600    TYPE,   SGNVAL
6016
6017 045236 004737 046042    77$:    JSR     PC,CTRLCG     ;TEST IF CTRL C/G
6018 045242 005200          INC     RO
6019 045244 022700 000377    CMP     #255.,RO      ;TEST IF DONE
6020 045250 001414          BEQ     READ          ;BR IF DONE
6021 045252 000137 044602    JMP     1$            ;TRY NEXT VALUE
6022
6023 045256 000000    100$:   0
6024 045260 000000    101$:   0
6025 045262 000000    102$:   0
6026 045264 000000    103$:   0
6027 045266 000000    104$:   0
6028 045270 000000    105$:   0
6029 045272 000000    106$:   0
6030 045274 000000    107$:   0
6031 045276 000000    110$:   0
6032 045300 000000    111$:   0

```

```
6033
6034
6035 045302 005037 041746
6036 045306 013702 041752
6037 045312 063702 041750
6038 045316 120237 046214
6039 045322 101026
6040 045324 013702 041760
6041 045330 063702 041764
6042 045334 120237 046212
6043 045340 101017
6044 045342 013702 041762
6045 045346 063702 041766
6046 045352 120237 046210
6047 045356 101010
6048 045360 123737 046206 041754
6049 045366 101004
6050 045370 012737 041702 045516
6051 045376 000405
6052 045400 012737 041714 045516 1$:
6053 045406 005237 041746
6054 045412 013746 041750 2$:
6055 045416 104405
6056 045420 104401 041623
6057 045424 013702 041760
6058 045430 063702 041762
6059 045434 010246
6060 045436 104405
6061 045440 104401 041066
6062 045444 013746 041754
6063 045450 104405
6064 045452 104401 041642
6065 045456 013702 041764
6066 045462 063702 041766
6067 045466 010246
6068 045470 104405
6069 045472 104401 041111
6070 045476 013746 041752
6071 045502 104405
6072 045504 104401 041663
6073
6074
6075
6076 045510 104401 040730
6077 045514 104401
6078 045516 041702
6079
```

;NOW ACCOUNT FOR ALL THE ERRORS

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

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

```

6080
6081 ;TYPE OUT MEMORY RAM AND RAM MEMORY VALUES IF SELECTED
6082 045520 032777 004000 133412 READXX: BIT #SW11,@SWR ;TEST IF THIS IS WANTED
6083 045526 001440 BEQ 4$ ;BR IF NOT
6084 045530 005737 041736 TST DIFEX1 ;TEST IF BLASTING MODE
6085 045534 001435 BEQ 4$ ;BR IF NOT
6086
6087 045536 012700 042162 MOV #ROMVAL,R0 ;LOAD INITIAL POINTER
6088 045542 004737 045572 1$: JSR PC,2$ ;TYPE OUT LINE
6089 045546 022700 042262 CMP #ROMVAL+100,R0 ;TEST IF ALL DONE
6090 045552 001373 BNE 1$ ;BR IF NOT
6091 045554 104401 041617 TYPE, RTN
6092 045560 104401 041617 TYPE, RTN
6093 045564 104401 041617 TYPE, RTN
6094 045570 000417 BR 4$
6095
6096 045572 012701 000020 2$: MOV #16.,R1 ;LOAD WIDE COUNTER
6097 045576 104401 041617 TYPE, RTN
6098 045602 112002 3$: MOVB (R0)+,R2 ;GET A BYTE
6099 045604 042702 177400 BIC #177400,R2 ;MASK OFF HIGHER BITS
6100 045610 010246 MOV R2,-(SP) ;LOAD VALUE
6101 045612 104403 TYPOS
6102 045614 003 001 .BYTE 3,1
6103 045616 104401 041002 TYPE, DASH+3
6104 045622 005301 DEC R1 ;FINISHED ?
6105 045624 001366 BNE 3$
6106 045626 000207 RTS PC ;EXIT
6107
6108
6109 045630 005037 041736 4$: CLR DIFEX1
6110 045634 000177 000310 JMP @DIFEX0 ;EXIT THIS CRAZYNES
6111

```

```

6112                                     ;DIF LIN SUBROUTINE LIST MODE DATA COLLECTOR
6113
6114 045640 012777 004000 134110 LISTDT: MOV #CLRALL,@SFR ;CLEAR THE DEVICE
6115 045646 005077 134076 CLR @OFF ;PRIME OFFSET REG.
6116 045652 012777 177770 134072 MOV #-8.,@WCR ;LOAD WC.
6117 045660 012777 060000 134066 MOV #BUFO,@BAR ;LOAD BUS ADDR.
6118 045666 012777 000000 134052 MOV #0,@CSR ;SET RES. AND MODE
6119 045674 032737 002000 046146 BIT #2000,PRIME1 ;TEST GAIN BIT
6120 045702 001403 BEQ 3$ ;BR IF CLEARED
6121 045704 052777 002000 134034 BIS #2000,@CSR ;SELECT GAIN = 2
6122 045712 052777 000002 134036 3$: BIS #TESTZ,@SFR ;SET Z PULSES
6123 045720 052777 000001 134020 BIS #BIT0,@CSR ;ENABLE THE DEVICE
6124
6125 045726 105777 134014 1$: TSTB @CSR ;WAIT FOR IDLE
6126 045732 100375 BPL 1$
6127 045734 012700 060000 MOV #BUFO,R0 ;LOAD POINTER TO NEW DATA
6128 045740 005002 CLR R2
6129 045742 012703 000010 MOV #8.,R3 ;LOAD COUNTER
6130 045746 011001 2$: MOV (R0),R1 ;GET DATA WORD
6131 045750 005020 CLR (R0)+ ;CLR BUFFER WORD
6132 045752 042701 177400 BIC #177400,R1 ;MAKE OFF BITS
6133 045756 060102 ADD R1,R2 ;UPDATE TOTAL
6134 045760 005303 DEC R3 ;FINISHED ?
6135 045762 001371 BNE 2$ ;BR IF NOT
6136 045764 006202 ASR R2
6137 045766 006202 ASR R2
6138 045770 006202 ASR R2
6139 045772 000207 RTS PC ;EXIT
6140
6141 045774 012737 177777 046030 ;SUBROUTINE TO FIND THE TRUE EDGE OF ZERO DATA
6142 046002 004737 045640 STDATO: MOV #-1,10$ ;SET ENTRY FLAG
6143 046006 005702 1$: JSR PC,LISTDT ;GET DATA
6144 046010 001403 TST R2 ;CHECK DATA
6145 046012 005037 046030 BEQ 2$ ;BR IF ZERO
6146 046016 000771 CLR 10$ ;CLEAR ALREADY NON-ZERO DATA FLAG
6147 BR 1$
6148 046020 005737 046030 ;DATA WAS ZERO - CHECK IF WE STARTED IN 0 DATA REGION
6149 046024 001366 2$: TST 10$ ;TEST FLAG
6150 BNE 1$ ;BR IF NOT A GOOD TIME TO EXIT
6151 ;DATA HAS NOW GONE TO A GOOD ZERO DATA REGION
6152 ;NOW EXIT
6153 RTS PC ;BYEBYE
6154 10$: 0
6155 DIFT1: 0
6156 DIFT2: 0
6157 DIFT3: 0
HT2T3: 0

```

```

6158                                     ;SUBROUTINE TO TEST IF OPERATOR TYPED CTRL C OR G
6159 046042 105777 133076                CTRLCG: TSTB  @TKS      ;INPUT FLAG
6160 046046 100033                       BPL 2$           ;BR IF NON
6161 046050 017737 133072 046140         MOV @TKB,CTRCHA ;GET CHARACTER
6162 046056 042737 177600 046140         BIC #177600,CTRCHA ;MASK OFF BITS
6163 046064 022737 000003 046140         CMP #3,CTRCHA   ;TEST IF CTRL C
6164 046072 001014                       BNE 1$         ;BR IF NOT ^C
6165 046074 052777 000400 133654         BIS #ENDDMA,@SFR ;STOP NPR'S
6166 046102 000240                       NOP
6167 046104 000240                       NOP
6168 046106 000240                       NOP
6169 046110 052777 004000 133640         BIS #CLRALL,@SFR ;CLEAR THE DEVICE
6170 046116 005726                       TST (SP)+      ;CLEAN STACK
6171 046120 000137 002012                 JMP RTRT       ;RE-START
6172 046124 022737 000007 046140 1$:    CMP #7,CTRCHA  ;TEST IF CTRL G
6173 046132 001001                       BNE 2$         ;BR IF NOT
6174 046134 104406                       GTSWR         ;GET SWR VALUE
6175 046136 000207                       2$: RTS PC     ;EXIT
6176 046140 000000                       CTRCHA: 0
6177 046142 177777                       PRIME: -1.
6178 046144 177400                       PRIME0: 177400
6179 046146 000036                       PRIME1: 36
6180 046150 000000                       DIFEX0: 0
6181
6182                                     ;SUBROUTINE TO DETERMINE ERROR LIMITS FOR DIPLIN
6183 046152 012500 050110                LIMITS: MOV (R5)+,R0 ;GET 1ST ARG.
6184 046154 005737 050110                TST WFMODE     ;TEST IF ON TESTER
6185 046160 001402                       BEQ 1$         ;BR IF NOT
6186 046162 012500 046206                MOV (R5)+,R0  ;GET TESTER LIMITS
6187 046164 000401                       BR 2$
6188 046166 005725 046206                1$: TST (R5)+  ;BUMP ADDRESS
6189 046170 012701 046206                2$: MOV #NORCNT,R1 ;GET POINTER
6190 046174 012021                       MOV (R0)+,(R1)+ ;GET VALUE
6191 046176 012021                       MOV (R0)+,(R1)+
6192 046200 012021                       MOV (R0)+,(R1)+
6193 046202 012021                       MOV (R0)+,(R1)+
6194 046204 000205                       RTS R5        ;EXIT

```

```

6195                ;ACTUAL VALUES FOR DIF LIN ERROR TOLERANCE
6196
6197
6198 046206          322          NORCNT: .BYTE 210.          ;'NORMAL'' MIN COUNT VALUE
6199 046207          013          NORTOL: .BYTE 11.           ;'NORMAL'' TOLERANCE VALUE
6200 046210          036          NARCNT: .BYTE 30.           ;'NARROW/WIDE'' MAX. COUNT VALUE
6201 046211          032          NARTOL: .BYTE 26.           ;'NARROW/WIDE'' TOLERANCE VALUE
6202 046212          016          NURCNT: .BYTE 14.           ;'LARGE/SMALL'' MAX. COUNT VALUE
6203 046213          145          NURTOL: .BYTE 101.          ;'LARGE/SMALL'' TOLERANCE VALUE
6204 046214          000          OBSCNT: .BYTE 0.            ;'OBESITY'' MAX. COUNT VALUE
6205 046215          377          OBSTOL: .BYTE 377          ;'OBESITY'' TOLERANCE VALUE
6206
6207                ;DIFLIN ERROR TOLERANCE          GAIN = 1          USER
6208
6209 046216          342          025          G1LIM0: .BYTE 226.,21.        ;226 MIN. COUNT, =<2.0%
6210 046220          024          063          .BYTE 20.,51.          ;20 MAX. COUNT, =<5.0%
6211 046222          010          145          .BYTE 8.,101.         ;8 MAX. COUNT, =<10.%
6212 046224          000          377          .BYTE 0.,377         ;0 MAX COUNT, >10%
6213
6214                ;DIFLIN ERROR TOLERANCE          GAIN = 1          OPTION AREA
6215
6216 046226          342          024          G1LIM1: .BYTE 226.,20.        ;226 MIN COUNT, =<1.9%
6217 046230          024          061          .BYTE 20.,49.          ;20 MAX. COUNT, =<4.8%
6218 046232          010          140          .BYTE 8.,96.           ;8 MAX COUNT, =<9.5%
6219 046234          000          377          .BYTE 0.,377         ;0 MAX COUNT, >9.5%
6220
6221                ;DIFLIN ERROR TOLERANCE          GAIN = 2          USER
6222
6223 046236          330          031          G2LIM0: .BYTE 216.,25.        ;216. MIN COUNT, =<2.4%
6224 046240          036          073          .BYTE 30.,59.          ;30. MAX COUNT, =<6.0%
6225 046242          010          171          .BYTE 8.,121.         ;14. MAX COUNT, =<12.0%
6226 046244          000          377          .BYTE 0.,377         ;0 MAX COUNT, >12.0%
6227
6228                ;DIFLIN ERROR TOLERANCE          GAIN = 2          OPTION AREA
6229
6230 046246          330          025          G2LIM1: .BYTE 216.,21.        ;216. MIN COUNT, =<2.0%
6231 046250          036          063          .BYTE 30.,51.          ;30. MAX COUNT, =<5.0%
6232 046252          010          145          .BYTE 8.,101.         ;8. MAX COUNT, =<10.0%
6233 046254          000          377          .BYTE 0.,377         ;0 MAX COUNT, >10.0%
6234
  
```

6235	046256	015	012
6236	046260	020111	020075
6237	046266	052111	040511
6238	046274	030101	033461
6239	046302	045104	051525
6240	046310	047105	020124
6241	046316	047111	020107
6242	046324	020105	047101
6243	046332	043517	052040
6244	046340	042524	122
6245	046343	015	012
6246	046345	102	036440
6247	046352	040514	052123
6248	046360	020107	043117
6249	046366	042510	046040
6250	046374	040505	044522
6251	046402	041440	051117
6252	046410	052103	047511
6253	046416	051120	046517
6254	046422	015	012
6255	046424	020103	020075
6256	046432	052116	047522
6257	046440	051120	043517
6258	046446	020115	051120
6259	046454	041040	040514
6260	046462	047111	000107
6261	046466	015	012
6262	046470	020114	020075
6263	046476	044507	020103
6264	046504	052123	047440
6265	046512	034115	031060
6266	046520	047101	020104
6267	046526	031460	066
6268	046531	015	012
6269	046533	104	036440
6270	046540	043111	042506
6271	046546	052116	040511
6272	046554	044514	042516
6273	046562	052111	020131
6274	046570	052040	042510
6275	046576	030460	067
6276	046601	015	012
6277	046603	106	036440
6278	046610	047111	046101
6279	046616	041503	050105
6280	046624	041516	020105
6281	046632	040440	042116
6282	046640	051	
6283	046641	015	012
6284	046643	101	036440
6285	046650	045104	051525
6286	046656	047105	020124
6287	046664	040440	030460
6288	046672	052101	043040

LIST0: .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/


```

6320
6321 047106 000005          POTIME: RESET          ;CLEAR THE WORLD
6322 047110 104401 052242    TYPE, PRIM2          ;TELL OPERATOR ABOUT CABLES
6323 047114 012777 006035 132564  MOV #6035,@VTCHAR    ;HOME AND ERASE SCREEN
6324 047122 005077 132562    CLR @VTYPOS          ;CLEAR THE MAP P.C.
6325 047126 005037 046140    CLR CTRCHA           ;CLEAR TTY CHARACTER
6326
6327          ;ERASE THE SCREEN MAP
6328 047132 105777 132560    1$: TSTB @VTCSR      ;WAIT FOR READY
6329 047136 100375          BPL 1$              ;
6330 047140 052777 001000 132550  BIS #BIT9,@VTCSR    ;CLEAR THE MAP
6331 047146 105777 132544    2$: TSTB @VTCSR      ;WAIT FOR READY AGAIN
6332 047152 100375          BPL 2$              ;
6333 047154 042777 001000 132534  BIC #BIT9,@VTCSR    ;CLEAR THE ERASE BIT
6334          ;NOW LOAD THE REF. PATTERN INTO THE MAP
6335 047162 012701 010421    MOV #10421,R1        ;LOAD THE PIXEL VALUE
6336 047166 012700 003740    MOV #3740,R0         ;LOAD THE PIXEL ADDRESS
6337 047172 004737 050044    3$: JSR PC,DISPLY    ;LOAD THE DATA
6338 047176 005200          INC R0               ;UPDATE THE ADDRESS
6339 047200 022700 004000    CMP #4000,R0         ;FINISHED ?
6340 047204 001372          BNE 3$              ;BR UNTIL DONE
6341
6342 047206 012700 000000    MOV #0,R0            ;LOAD PIXEL ADDRESS
6343 047212 004737 050044    4$: JSR PC,DISPLY    ;LOAD THE DATA
6344 047216 005200          INC R0               ;UPDATE THE PIXEL ADDRESS
6345 047220 022700 000040    CMP #40,R0           ;TEST IF DONE
6346 047224 001372          BNE 4$              ;BR IF DONE THE VERTICAL
6347
6348 047226 012700 007740    MOV #7740,R0         ;LOAD PIXEL ADDRESS
6349 047232 004737 050044    5$: JSR PC,DISPLY    ;LOAD THE DATA
6350 047236 005200          INC R0               ;UPDATE THE PIXEL ADDRESS
6351 047240 022700 010000    CMP #10000,R0        ;TEST IF DONE
6352 047244 001372          BNE 5$              ;BR IF DONE
6353
6354          ;POSITION THE MAP AND DISPLAY
6355
6356 047246 012777 000005 132442  MOV #5,@VTCSR        ;LOAD ORGIN VALUE
6357 047254 012777 000060 132442  MOV #60,@VTINT       ;LOAD LUT 0
6358 047262 012777 000463 132434  MOV #463,@VTINT      ;LOAD LUT 1
6359 047270 052777 000400 132420  BIS #BIT8,@VTCSR     ;ENABLE THE MAP

```

```
6360          :NOW SAMPLE THE CAMERA CHANNELS
6361 047276    SAMCAM:
6362          :CAMERA 0, X INPUT
6363 047276 004537 047422      JSR      R5,ADJCAM      :
6364 047302 050126              ADMSG0              :CAM 0 MESSAGE
6365 047304 001730              DAC0                :DAC ADDRESS
6366 047306      000      000      .BYTE      0,0      :CAMERA 0, X AXIS
6367          :DO CAMERA 1, X INPUT
6368 047310 004537 047422      JSR      R5,ADJCAM      :
6369 047314 050312              ADMSG2              :CAM 1, X MESSAGE
6370 047316 001732              DAC1                :DAC ADDRESS
6371 047320      001      000      .BYTE      1,0      :CAMERA 1, X AXIS
6372          :DO CAMERA 2, X INPUT
6373 047322 004537 047422      JSR      R5,ADJCAM      :
6374 047326 050476              ADMSG4              :CAM 2, X MESSAGE
6375 047330 001734              DAC2                :DAC ADDRESS
6376 047332      002      000      .BYTE      2,0      :CAMERA 2, X AXIS
6377          :DO CAMERA 3, X INPUT
6378 047334 004537 047422      JSR      R5,ADJCAM      :
6379 047340 050662              ADMSG6              :CAM 3, X MESSAGE
6380 047342 001736              DAC3                :DAC ADDRESS
6381 047344      003      000      .BYTE      3,0      :CAMERA 3, X AXIS
6382          :NOW SAMPLE THE Y CAMERA CHANNELS
6383          :DO CAMERA 0 Y INPUT
6384 047346 004537 047422      JSR      R5,ADJCAM      :
6385 047352 050220              ADMSG1              :CAM 0 Y MESSAGE
6386 047354 001732              DAC1                :DAC ADDRESS
6387 047356      000      001      .BYTE      0,1      :CAMERA 0, Y AXIS
6388          :DO CAMERA 1, Y INPUT
6389 047360 004537 047422      JSR      R5,ADJCAM      :
6390 047364 050404              ADMSG3              :CAM 1, Y MESSAGE
6391 047366 001730              DAC0                :DAC ADDRESS
6392 047370      001      001      .BYTE      1,1      :CAMERA 1, Y AXIS
6393          :DO CAMERA 2, Y INPUT
6394 047372 004537 047422      JSR      R5,ADJCAM      :
6395 047376 050570              ADMSG5              :CAM 2, Y MESSAGE
6396 047400 001736              DAC3                :DAC ADDRESS
6397 047402      002      001      .BYTE      2,1      :CAMERA 2, Y AXIS
6398          :DO CAMERA 3, Y INPUT
6399 047404 004537 047422      JSR      R5,ADJCAM      :
6400 047410 050754              ADMSG7              :CAM 3, Y MESSAGE
6401 047412 001734              DAC2                :DAC ADDRESS
6402 047414      003      001      .BYTE      3,1      :CAMERA 3 Y AXIS
6403          :
6404 047416 000137 002644      JMP      RBEGO
```

```

6405                ;REPORT THE MESSAGE FIRST
6406
6407 047422 012500    ADJCAM: MOV      (R5)+,R0      ;GET ASCII POINTER
6408 047424 005777 132256 3$:      TST      @VTCHAR      ;WAIT FOR READY
6409 047430 100375    BPL      3$
6410 047432 012777 012000 132254 1$:      MOV      #12000,@VTCXY ;POSITION THE CAHRACTERS
6411 047440 005777 132242      TST      @VTCHAR      ;WAIT FOR CHAR. READY
6412 047444 100375    BPL      1$
6413 047446 112077 132234      MOVB     (R0)+,@VTCHAR ;LOAD THE CHAR.
6414 047452 105710    TSTB     (R0)          ;TEST IF TERM.
6415 047454 001371    BNE      1$          ;BR IF NOT
6416
6417 047456 013537 050104      MOV      @(R5)+,DACADR ;SAVE ADDRESS
6418 047462 012537 050112      MOV      (R5)+,CAMVAL ;SAVE CAMERA DATA
6419
6420 047466 012777 002315 000410 2$:      MOV      #2315,@DACADR ;PRESET DAC TO - GAIN INPUT
6421 047474 004737 047576      JSR      PC,SAMDAT
6422 047500 010137 050120      MOV      R1,VAL0      ;SAVE RESULTS
6423 047504 004737 050022      JSR      PC,FXLIN     ;POSITION THE CROSS HAIR LINE
6424 047510 012777 004000 000366      MOV      #4000,@DACADR ;PRESET DAC TO OFFSET
6425 047516 004737 047576      JSR      PC,SAMDAT
6426 047522 010137 050122      MOV      R1,VAL1      ;SAMPLE THE DATA
6427 047526 004737 050022      JSR      PC,FXLIN     ;SAVE AVERAGE RESULTS
6428 047532 012777 005463 000344      MOV      #5463,@DACADR ;POSITION THE CROSS HAIR LINE
6429 047540 004737 047576      JSR      PC,SAMDAT
6430 047544 010137 050124      MOV      R1,VAL2      ;PRESET DAC TO + GAIN INPUT
6431 047550 004737 050022      JSR      PC,FXLIN     ;SAMPLE THE DATA
6432 047554 004737 046042      JSR      PC,CTRLCG    ;SAVE AVERAGE RESULTS
6433 047560 122737 000103 046140      CMPB     #'C,CTRCHA   ;POSITION THE CROSS HAIR LINE
6434 047566 001337      BNE      2$          ;TEST FOR CTRL C/G
6435 047570 005037 046140      CLR      CTRCHA      ;CHECK IF CHARACTER WAS A \C\
6436 047574 000205      RTS      R5          ;BR IF NOT
6437
6438                ;COLLECT 64 SAMPLES FROM THE SELECTED CAMERA
6439                ;R1 CONTAINS THE AVERAGE
6440
6441 047576 012700 060000      SAMDAT: MOV      #BUF0,R0
6442 047602 012701 062000      MOV      #BUF1,R1
6443 047606 005020 1$:      CLR      (R0)+      ;CLEAR THE BUFFER
6444 047610 020001      CMP      R0,R1      ;FINISHED ?
6445 047612 001375      BNE      1$          ;BR IF NOT
6446
6447 047614 012777 177700 132130      MOV      #-100,@WCR  ;LOAD WORD COUNT
6448 047622 012777 060000 132124      MOV      #BUF0,@BAR  ;LOAD ADDRESS
6449 047630 113777 050112 132130      MOVB     CAMVAL,@CSRHB ;SELECT THE CAMERA
6450 047636 052777 000002 132112      BIS      #TESTZ,@SFR ;SET TEST Z ENABLE
6451 047644 052777 000001 132074      BIS      #BIT0,@CSR  ;ENABLE NCV11
6452
6453 047652 105777 132070 2$:      TSTB     @CSR        ;WAIT UNTIL DONE
6454 047656 100375      BPL      2$
6455
6456 047660 012777 004000 132070      MOV      #CLRALL,@SFR ;CLEAR THE DEVICE
6457 047666 005037 050100      CLR      $TEMP1
6458 047672 005037 050102      CLR      $TEMP2
  
```


6513	050112	000000				CAMVAL: 0
6514	050114	000000				CURRENT: 0
6515	050116	000000				AGAN: 0
6516	050120	000000				VAL0: 0
6517	050122	000000				VAL1: 0
6518	050124	000000				VAL2: 0
6519	050126	040503	042515	040522		ADMSG0: .ASCII /CAMERA 00/
6520	050134	030040	060			
6521	050137	015	012			.BYTE 15,12
6522	050141	101	045104	051525		.ASCII /ADJUST R09 FOR X OFFSET/
6523	050146	020124	030122	020071		
6524	050154	047506	020122	020130		
6525	050162	043117	051506	052105		
6526	050170	015	012			.BYTE 15,12
6527	050172	042101	052512	052123		.ASCIIZ /ADJUST R24 FOR X GAIN/
6528	050200	051040	032062	043040		
6529	050206	051117	054040	043440		
6530	050214	044501	000116			
6531	050220	040503	042515	040522		ADMSG1: .ASCII /CAMERA 00/
6532	050226	030040	060			
6533	050231	015	012			.BYTE 15,12
6534	050233	101	045104	051525		.ASCII /ADJUST R13 FOR Y OFFSET/
6535	050240	020124	030522	020063		
6536	050246	047506	020122	020131		
6537	050254	043117	051506	052105		
6538	050262	015	012			.BYTE 15,12
6539	050264	042101	052512	052123		.ASCIIZ /ADJUST R20 FOR Y GAIN/
6540	050272	051040	030062	043040		
6541	050300	051117	054440	043440		
6542	050306	044501	000116			
6543	050312	040503	042515	040522		ADMSG2: .ASCII /CAMERA 01/
6544	050320	030040	061			
6545	050323	015	012			.BYTE 15,12
6546	050325	101	045104	051525		.ASCII /ADJUST R10 FOR X OFFSET/
6547	050332	020124	030522	020060		
6548	050340	047506	020122	020130		
6549	050346	043117	051506	052105		
6550	050354	015	012			.BYTE 15,12
6551	050356	042101	052512	052123		.ASCIIZ /ADJUST R23 FOR X GAIN/
6552	050364	051040	031462	043040		
6553	050372	051117	054040	043440		
6554	050400	044501	000116			
6555	050404	040503	042515	040522		ADMSG3: .ASCII /CAMERA 01/
6556	050412	030040	061			
6557	050415	015	012			.BYTE 15,12
6558	050417	101	045104	051525		.ASCII /ADJUST R14 FOR Y OFFSET/
6559	050424	020124	030522	020064		
6560	050432	047506	020122	020131		
6561	050440	043117	051506	052105		
6562	050446	015	012			.BYTE 15,12
6563	050450	042101	052512	052123		.ASCIIZ /ADJUST R19 FOR Y GAIN/
6564	050456	051040	034461	043040		
6565	050464	051117	054440	043440		
6566	050472	044501	000116			

6567	050476	040503	042515	040522	ADMSG4: .ASCII	/CAMERA 02/
6568	050504	030040	062			
6569	050507	015	012		.BYTE	15,12
6570	050511	101	045104	051525	.ASCII	/ADJUST R11 FOR X OFFSET/
6571	050516	020124	030522	020061		
6572	050524	047506	020122	020130		
6573	050532	043117	051506	052105		
6574	050540	015	012		.BYTE	15,12
6575	050542	042101	052512	052123	.ASCIZ	/ADJUST R22 FOR X GAIN/
6576	050550	051040	031062	043040		
6577	050556	051117	054040	043440		
6578	050564	044501	000116			
6579	050570	040503	042515	040522	ADMSG5: .ASCII	/CAMERA 02/
6580	050576	030040	062			
6581	050601	015	012		.BYTE	15,12
6582	050603	101	045104	051525	.ASCII	/ADJUST R15 FOR Y OFFSET/
6583	050610	020124	030522	020065		
6584	050616	047506	020122	020131		
6585	050624	043117	051506	052105		
6586	050632	015	012		.BYTE	15,12
6587	050634	042101	052512	052123	.ASCIZ	/ADJUST R18 FOR Y GAIN/
6588	050642	051040	034061	043040		
6589	050650	051117	054440	043440		
6590	050656	044501	000116			
6591	050662	040503	042515	040522	ADMSG6: .ASCII	/CAMERA 03/
6592	050670	030040	063			
6593	050673	015	012		.BYTE	15,12
6594	050675	101	045104	051525	.ASCII	/ADJUST R12 FOR X OFFSET/
6595	050702	020124	030522	020062		
6596	050710	047506	020122	020130		
6597	050716	043117	051506	052105		
6598	050724	015	012		.BYTE	15,12
6599	050726	042101	052512	052123	.ASCIZ	/ADJUST R21 FOR X GAIN/
6600	050734	051040	030462	043040		
6601	050742	051117	054040	043440		
6602	050750	044501	000116			
6603	050754	040503	042515	040522	ADMSG7: .ASCII	/CAMERA 03/
6604	050762	030040	063			
6605	050765	015	012		.BYTE	15,12
6606	050767	101	045104	051525	.ASCII	/ADJUST R16 FOR Y OFFSET/
6607	050774	020124	030522	020066		
6608	051002	047506	020122	020131		
6609	051010	043117	051506	052105		
6610	051016	015	012		.BYTE	15,12
6611	051020	042101	052512	052123	.ASCIZ	/ADJUST R17 FOR Y GAIN/
6612	051026	051040	033461	043040		
6613	051034	051117	054440	043440		
6614	051042	044501	000116			
6615						
6616	051046	015	012		PRIMO: .BYTE	15,12
6617	051050	044124	020105	042523	.ASCII	/THE SELF-TEST CONNECTOR MUST BE INSTALLED ON A017/
6618	051056	043114	052055	051505		
6619	051064	020124	047503	047116		
6620	051072	041505	047524	020122		

6621	051100	052515	052123	041040
6622	051106	020105	047111	052123
6623	051114	046101	042514	020104
6624	051122	047117	040440	030460
6625	051130	067		
6626	051131	015	012	
6627	051133	124	042510	051440
6628	051140	044527	041524	020110
6629	051146	047117	040440	030460
6630	051154	020067	052515	052123
6631	051162	041040	020105	047111
6632	051170	021040	040515	047111
6633	051176	027124	020042	047520
6634	051204	044523	044524	047117
6635	051212	024040	047524	040527
6636	051220	042122	052040	042510
6637	051226	044440	047457	041440
6638	051234	047117	042516	052103
6639	051242	051117	051	
6640	051245	015	012	000
6641	051250	015	012	
6642	051252	044124	020105	042523
6643	051260	043114	052055	051505
6644	051266	020124	047503	047116
6645	051274	041505	047524	020122
6646	051302	052515	052123	041040
6647	051310	020105	042522	047515
6648	051316	042526	020104	051106
6649	051324	046517	052040	042510
6650	051332	040440	030460	067
6651	051337	015	012	
6652	051341	124	042510	051440
6653	051346	044527	041524	020110
6654	051354	047117	040440	030460
6655	051362	020067	052515	052123
6656	051370	041040	020105	047111
6657	051376	021040	050117	051105
6658	051404	021056	050040	051517
6659	051412	052111	047511	020116
6660	051420	040450	040527	020131
6661	051426	051106	046517	052040
6662	051434	042510	044440	047457
6663	051442	041440	047117	042516
6664	051450	052103	051117	051
6665	051455	015	012	000
6666	051460	047111	042523	052122
6667	051466	051040	046501	050055
6668	051474	045501	041440	041101
6669	051502	042514	044440	052116
6670	051510	020117	044124	020105
6671	051516	047523	045503	052105
6672	051524	047440	020116	044124
6673	051532	020105	030101	033461
6674	051540	015	012	

.BYTE 15,12
.ASCII \THE SWITCH ON A017 MUST BE IN 'MAINT.'" POSITION (TOWARD THE I/O CONNECT

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

6675	051542	042523	020124	040522
6676	051550	026515	040520	020113
6677	051556	053523	052111	044103
6678	051564	051505	052040	020117
6679	051572	044442	021116	020054
6680	051600	047042	051117	021115
6681	051606	040440	042116	021040
6682	051614	043117	021106	
6683	051620	015	012	
6684	051622	047111	042523	052122
6685	051630	041040	040514	045516
6686	051636	051040	046517	044440
6687	051644	052116	020117	047522
6688	051652	020115	046102	051501
6689	051660	042524	020122	047523
6690	051666	045503	052105	
6691	051672	015	012	
6692	051674	046120	040505	042523
6693	051702	042040	050105	042522
6694	051710	051523	052040	042510
6695	051716	021040	027511	021117
6696	051724	040440	042116	021040
6697	051732	054105	041505	052125
6698	051740	021105	041040	052125
6699	051746	047524	051516	052040
6700	051754	043517	052105	042510
6701	051762	122		
6702	051763	015	012	
6703	051765	124	042510	032040
6704	051772	050040	047522	042503
6705	052000	051523	046040	042105
6706	052006	020123	044527	046114
6707	052014	046040	043511	052110
6708	052022	053440	042510	020116
6709	052030	047504	042516	041440
6710	052036	051117	042522	052103
6711	052044	054514		
6712	052046	015	012	000
6713	052051	015	012	
6714	052053	120	042514	051501
6715	052060	020105	042522	047515
6716	052066	042526	051040	046501
6717	052074	050055	045501	041440
6718	052102	041101	042514	
6719	052106	015	012	
6720	052110	040440	042116	044440
6721	052116	051516	051105	020124
6722	052124	044124	020105	046102
6723	052132	051501	042524	020104
6724	052140	047522	020115	047111
6725	052146	047524	051440	041517
6726	052154	042513	020124	047117
6727	052162	040440	030460	020067
6728	052170	047515	052504	042514

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

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

.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/

6729	052176	015	012			.BYTE 15,12
6730	052200	042504	051120	051505		.ASCII /DEPRESS THE "CR" KEY WHEN READY/
6731	052206	020123	044124	020105		
6732	052214	041442	021122	045440		
6733	052222	054505	053440	042510		
6734	052230	020116	042522	042101		
6735	052236	131				
6736	052237	015	012	000		.BYTE 15,12,0
6737	052242	015	012		PRIM2:	.BYTE 15,12
6738	052244	050042	030512	020042		.ASCII /"PJ1" CABLE MUST BE CONNECTED TO THE A017 CONNECTOR/
6739	052252	040503	046102	020105		
6740	052260	052515	052123	041040		
6741	052266	020105	047503	047116		
6742	052274	041505	042524	020104		
6743	052302	047524	052040	042510		
6744	052310	040440	030460	020067		
6745	052316	047503	047116	041505		
6746	052324	047524	122			
6747	052327	015	012	000		.BYTE 15,12,0
6748	052332	015	012		PRIM4:	.BYTE 15,12
6749	052334	050042	031512	020042		.ASCII /"PJ3" CABLE MUST BE CONNECTED TO THE M8036 CONNECTOR/
6750	052342	040503	046102	020105		
6751	052350	052515	052123	041040		
6752	052356	020105	047503	047116		
6753	052364	041505	042524	020104		
6754	052372	047524	052040	042510		
6755	052400	046440	030070	033063		
6756	052406	041440	047117	042516		
6757	052414	052103	051117			
6758	052420	015	012	000		.BYTE 15,12,0
6759		052424				.EVEN
6760	052424	015	012	007	FATAL0:	.BYTE 15,12,7 ;'RUNIT' MUST BE ON A WORD BOUNDRY
6761	052427	106	052101	046101		.ASCII /FATAL BUS TRAP WHEN PERFORMING TEST '/
6762	052434	041040	051525	052040		
6763	052442	040522	020120	044127		
6764	052450	047105	050040	051105		
6765	052456	047506	046522	047111		
6766	052464	020107	042524	052123		
6767	052472	021040				
6768	052474	021103	040411	020124	RUNIT:	.ASCIZ /C" AT LOCATION / ;'RUNIT' MUST BE A WORD BOUNDRY
6769	052502	047514	040503	044524		
6770	052510	047117	000040			
6771	052514	005015	051120	043517	RUNITA:	.ASCIZ <15><12>/PROGRAM RESTARTING/ ;'RUNIT' MUST BE A WORD BOUNDRY
6772	052522	040522	020115	042522		
6773	052530	052123	051101	044524		
6774	052536	043516	000			
6775	052541	015	012	007		.BYTE 15,12,7,0
6776	052544	000				
6777	052545	015	012		MSGMEM:	.BYTE 15,12
6778	052547	122	047125	044516		.ASCIZ /RUNNING WITH /
6779	052554	043516	053440	052111		
6780	052562	020110	000			
6781	052565	040	020113	043117	MSGK:	.ASCIZ / K OF MEMORY/<15><12>
6782	052572	046440	046505	051117		

6783	052600	006531	000012						
6784	052604	015	012	007	WARNO:	.BYTE	15,12,7		
6785	052607	117	052103	046101		.ASCII	/OCTAL ADDRESS OF TERMINAL = /		
6786	052614	040440	042104	042522					
6787	052622	051523	047440	020106					
6788	052630	042524	046522	047111					
6789	052636	046101	036440	000040					
6790	052644	033	000		ESCP:	.BYTE	33,0		
6791	052646	043113	027462	062	KFO:	.ASCII	\KF2/2\		
6792	052653	015	000			.BYTE	15,0		
6793	052655	106	031515	060	FMO:	.ASCII	\FM30\		
6794	052661	015	000			.BYTE	15,0		
6795	052663	104	030111	031457	DIO:	.ASCII	\DIO/37\		
6796	052670	067							
6797	052671	015	002			.BYTE	15,2		
6798	052673				DIDATA:				
6799	052673	060	060	060		.BYTE	60,60,60,40		
6800	052676	040							
6801	052677	060	060	060		.BYTE	60,60,60,40		
6802	052702	040							
6803	052703	060	060	060		.BYTE	60,60,60,40		
6804	052706	040							
6805	052707	060	060	060		.BYTE	60,60,60,40		
6806	052712	040							
6807	052713	060	060	060		.BYTE	60,60,60,40		
6808	052716	040							
6809	052717	060	060	060		.BYTE	60,60,60,40		
6810	052722	040							
6811	052723	060	060	060		.BYTE	60,60,60,40		
6812	052726	040							
6813	052727	060	060	060		.BYTE	60,60,60,40		
6814	052732	040							
6815	052733	060	060	060		.BYTE	60,60,60,40		
6816	052736	040							
6817	052737	060	060	060		.BYTE	60,60,60,40		
6818	052742	040							
6819	052743	060	060	060		.BYTE	60,60,60,40		
6820	052746	040							
6821	052747	060	060	060		.BYTE	60,60,60,40		
6822	052752	040							
6823	052753	060	060	060		.BYTE	60,60,60,40		
6824	052756	040							
6825	052757	060	060	060		.BYTE	60,60,60,40		
6826	052762	040							
6827	052763	060	060	060		.BYTE	60,60,60,40		
6828	052766	040							
6829	052767	060	060	060		.BYTE	60,60,60,40		
6830	052772	040							
6831	052773	060	060	060		.BYTE	60,60,60,40		
6832	052776	040							
6833	052777	060	060	060		.BYTE	60,60,60,40		
6834	053002	040							
6835	053003	060	060	060		.BYTE	60,60,60,40		
6836	053006	040							

6837	053007	060	060	060		.BYTE	60,60,60,40
6838	053012	040					
6839	053013	060	060	060		.BYTE	60,60,60,40
6840	053016	040					
6841	053017	060	060	060		.BYTE	60,60,60,40
6842	053022	040					
6843	053023	060	060	060		.BYTE	60,60,60,40
6844	053026	040					
6845	053027	060	060	060		.BYTE	60,60,60,40
6846	053032	040					
6847	053033	060	060	060		.BYTE	60,60,60,40
6848	053036	040					
6849	053037	060	060	060		.BYTE	60,60,60,40
6850	053042	040					
6851	053043	060	060	060		.BYTE	60,60,60,40
6852	053046	040					
6853	053047	060	060	060		.BYTE	60,60,60,40
6854	053052	040					
6855	053053	060	060	060		.BYTE	60,60,60,40
6856	053056	040					
6857	053057	060	060	060		.BYTE	60,60,60,40
6858	053062	040					
6859	053063	060	060	060		.BYTE	60,60,60,40
6860	053066	040					
6861	053067	060	060	060		.BYTE	60,60,60,40
6862	053072	040					
6863	053073	003	000			.BYTE	3,0
6864	053075	103	030115	031457	CMO:	.ASCII	\CMO/37\
6865	053102	067					
6866	053103	015	000			.BYTE	15,0
6867	053105	103	000103		CCO:	.ASCII	/CC/
6868	053110	043520	027460	033463	PGO:	.ASCII	\PGO/37\
6869	053116	015	000			.BYTE	15,0
6870	053120	050132			ZPO:	.ASCII	/ZP/
6871	053122	015	000			.BYTE	15,0
6872						.EVEN	
6873							
6874						:BUFFER AREA	
6875							
6876		060000				.=60000	
6877	060000	001000			BUF0:	.BLKW	512.
6878	062000	001000			BUF1:	.BLKW	512.
6879	064000	000240			BUF2:	NOP	
6880							
6881		000001				.END	

CZNCCA NCV11
CZNCCA.P11

DIAGNOSTIC
CROSS REFERENCE

MACY11 27(654)

8-AUG-78

15:05

C 14
PAGE 159

SEQ 0171

LSBTXT 041610
MMVEC = 000250
MORTST= 000000
MCGK 052565
MSGMEM 052545
MSG21 041175
NARA 041760
NARB 041762
NARCNT 046210
NARMSG 041066
NARROW 041756
NARTOL 046211
NLSI11 003354
NOEXIT 043274
NOMIAL 041742
NORCNT 046206
NORMAL 041754
NORMSG 041642
NORTOL 046207
NURCNT 046212
NURTOL 046213
OBSCNT 046214
OBSTOL 046215
OFF 001750

5366# 6012* 6013* 6014*
135# 4676* 4677*
580# 877 3975 5023
835 6781#
825 6777#
5313#
5400# 5982* 6040 6057
5401# 5976* 6044 6058
6046 6200#
5299# 6061
3480* 3543* 3590* 3656* 3970 5399#
5964 6201#
797# 838* 842* 2673
5558* 5625* 5652 5662* 5669#
5391# 5924
6048 6189 6198#
5398# 5986* 6048 6062
5374# 6064
5962 6199#
6042 6202#
5966 6203#
6038 6204#
6205#

548#	852	1039*	1040	1054*	1061	1093*	1095	1375*	1388	2066*	2101	2140
2171*	2206*	2240*	2280*	2291	2306*	2320	2457*	2495*	2525*	2572*	2612*	2654*
2701*	2722*	2743*	2764*	2785*	2806*	2827*	2848*	2869*	2890*	2911*	2932*	2956*
2991*	3012*	3042*	3063*	3089*	3112*	3286*	3486*	3535*	3578*	3610*	3644*	5871*
6115*												
5297#												
3463*	3464	3470	3504*	3505	3515*	3516	3522	3556*	3557	5405#		
3962#	5173*											
3951#	5174											
5380#	6050	6078										
741	5609#											
50#	806*	813*	3688*	3691*	3701*	3706	4029*	4235*	4312*	4331*	4338*	4345*
4359*	4361*	4469*	4475*	4533*	4574*	4591*	4670	4719*	4815*	4933*	4938*	4973*
4976*	5025*	5166*	5526*	5533*	5538*	5541*	5550*	5606*	5677*	5679*	5681*	5704*
5735*	5742*	5785*	5831*	5836*	5843*	5850*	5863*	5923*	5941*	5947*	6009*	6017*
6088*	6106*	6139*	6142*	6152*	6175*	6337*	6343*	6349*	6421*	6423*	6425*	6427*
6429*	6431*	6432*	6488*	6497*	6506*							
5557	5604	5624	6868#									

PGO 053110
PIRQ = 177772
PIRQVE= 000240
POTIME 047106
PRIME 046142
PRIME0 046144
PRIME1 046146
PRIMO 051046
PRIM1 052051
PRIM2 052242
PRIM3 051622
PRIM4 052332
PRIM5 051460
PRIM6 051250

36#
130#
721 6321#
6177#
6178#
5520* 5527* 5799* 5806* 5874 6119 6179#
5506 5798 6616#
5562 6713#
6322 6737#
5609 6684#
804 6748#
5507 6666#
784 6641#

TST15	005052	1050#							
TST150	025152	3222	3236	3242#					
TST151	025276	3245	3259	3265#					
TST152	025664	3305	3315	3327	3329	3332	3340#		
TST153	026244	3376	3386	3393	3399	3401	3404	3412#	
TST154	026466	3442	3450	3457#					
TST155	026762	3460	3501	3503	3510#				
TST156	027260	3513	3553	3555	3562#				
TST157	027474	3568	3596	3601#					
TST16	005312	1089#							
TST160	027634	3609	3622	3628#					
TST161	030054	3634	3663	3668#					
TST17	005362	1096	1102#						
TST2	004020	864	872	880#					
TST20	005554	1131	1136#						
TST21	005726	1150	1157	1162#					
TST22	006112	1177	1185	1190#					
TST23	006252	1204	1208	1213#					
TST24	006366	1226	1231#						
TST25	006460	1241	1247#						
TST26	006652	1270	1277	1282#					
TST27	007052	1307	1314	1319#					
TST3	004122	900#							
TST30	007322	1369#							
TST31	007464	1385	1389	1395#					
TST32	007572	1412	1418#						
TST33	007736	1442	1448#						
TST34	010032	1459	1465#						
TST35	010126	1476	1482#						
TST36	010222	1493	1498#						
TST37	010316	1509	1515#						
TST4	004170	907	913#						
TST40	010412	1526	1532#						
TST41	010506	1543	1548#						
TST42	010602	1559	1565#						
TST43	010676	1576	1583#						
TST44	010772	1594	1600#						
TST45	011066	1611	1618#						
TST46	011162	1629	1635#						
TST47	011256	1646	1652#						
TST5	004242	920	926#						
TST50	011360	1663	1669#						
TST51	011462	1680	1686#						
TST52	011564	1697	1703#						
TST53	011666	1714	1720#						
TST54	011770	1731	1737#						
TST55	012072	1748	1754#						
TST56	012174	1765	1771#						
TST57	012276	1782	1788#						
TST6	004352	940	945#						
TST60	012400	1799	1805#						
TST61	012502	1816	1822#						
TST62	012604	1833	1839#						
TST63	012706	1850	1856#						

CZNCCA NCV11
CZNCCA.P11

DIAGNOSTIC
CROSS REFERENCE

MACY11 27(654)
TABLE

8-AUG-78 15:05

B 15
PAGE 171

SEQ 0183

\$MSGAD 001210
\$MSGLG 001212
\$MSGTY 001174
\$MSWR 033536
\$MTYP1 001225
\$MTYP2 001231
\$MTYP3 001235
\$MTYP4 001241
\$MULT 036504
\$MXCNT 034370
\$NULL 001154
\$NWTST= 000001

396# 4563* 4566
397# 4568*
390# 4561 4569* 4581 4585*
3997 4110#
411#
419#
422#
425#
4783# 5735 5941
4273 4283#
374# 4334 4363
844# 877# 897# 910# 923# 942# 959# 972# 984# 996# 1013# 1030# 1047#
1086# 1099# 1133# 1159# 1187# 1210# 1228# 1244# 1279# 1316# 1366# 1392# 1415#
1445# 1462# 1479# 1495# 1512# 1529# 1545# 1562# 1580# 1597# 1615# 1632# 1649#
1666# 1683# 1700# 1717# 1734# 1751# 1768# 1785# 1802# 1819# 1836# 1853# 1870#
1887# 1904# 1921# 1938# 1955# 1972# 1989# 2006# 2024# 2042# 2059# 2111# 2164#
2199# 2233# 2270# 2296# 2324# 2349# 2374# 2399# 2426# 2450# 2477# 2517# 2564#
2604# 2641# 2695# 2716# 2737# 2758# 2779# 2800# 2821# 2842# 2863# 2884# 2905#
2926# 2947# 2982# 3006# 3036# 3057# 3078# 3104# 3138# 3165# 3192# 3216# 3239#
3262# 3337# 3409# 3454# 3507# 3559# 3598# 3625# 3665#

\$OCNT 035076
\$OMODE 035100
\$OVER 034354
\$PASS 001202
\$PASTM 001006
\$POWER 036052
\$PWRDN 035704
\$PWRMG 036040
\$PWRUP 035756
\$QUES 001170
\$RDCHR 033266
\$RDDEC= ***** U
\$RDLIN 033406
\$RDOCT 033560
\$RDSZ = 000010
\$RESRE 036446
\$RTNAD 030176
\$R2A = ***** U
\$SAVRE 036410
\$SAVR6 036050
\$SB2D 037232
\$SCOPE 034106
\$SETUP= 000137

4395* 4424* 4437#
4390* 4394* 4399 4402* 4413* 4439#
4237 4253 4261 4271 4280#
393# 628* 800 3326 3398 3500 3552 3686* 3687* 3695 3708 4267 4284
338#
4635 4640#
604 4604# 4632
4635#
4614 4620#
381# 755 4043 4092 4108 4363 4495
4056# 5018
5021
4084# 5019
4124# 5020
4077#
4757# 5022
3707#
5023
4741# 5021
4613* 4621 4622* 4623* 4639#
4971# 6009
598 4233#
582# 597 598 600 602 604 606 607 608 610 637 640 3684
3980 4114 4234 4456 4482 4490
806 813 4659#

\$SIZE 036062
\$SIZEX 036340
\$STUP = 177777
\$SVLAD 034320
\$SVPC = 000106
\$SWR = 167400

582#
4245 4274#
312# 317
4# 21 285 286 287 288 289 290 291 378 379 380 607
608 610 611 848 881 901 914 927 946 963 976 988 1000
1017 1034 1051 1090 1103 1137 1163 1191 1214 1232 1248 1283 1320
1370 1396 1419 1449 1466 1483 1499 1516 1533 1549 1566 1584 1601
1619 1636 1653 1670 1687 1704 1721 1738 1755 1772 1789 1806 1823

	1495	1512	1529	1545	1562	1580	1597	1615	1632	1649	1666	1683	1700	1717	1734
	1751	1768	1785	1802	1819	1836	1853	1870	1887	1904	1921	1938	1955	1972	1989
	2006	2024	2042	2059	2111	2164	2199	2233	2270	2296	2324	2349	2374	2399	2426
	2450	2477	2517	2564	2604	2641	2695	2716	2737	2758	2779	2800	2821	2842	2863
	2884	2905	2926	2947	2982	3006	3036	3057	3078	3104	3138	3165	3192	3216	3239
	3262	3337	3409	3454	3507	3559	3598	3625	3665	3673	3712	3975	4114	4152	4219
	4284	4363	4441	4496	4543	4600	4643	4724	4769	4816	4898	4960	4978	5001	5024
.TITLE	11														
.WORD	298	299	300	316	335	336	337	338	339	340	349	352	353	354	355
	358	359	360	361	362	363	364	367	368	369	390	391	392	393	394
	395	396	397	401	402	403	416	420	423	426	427	428	429	430	431
	3689	3692	3707	3965	3966	3968	3970	3973	3974	4151	4313	4360	4439	4522	4527
	4575	4635	4671	4720	4721	4934	4977	5008							

ERRORS DETECTED: 0

CZNCCA NCV11 DIAGNOSTIC MACY11 27(654) 8-AUG-78 15:05 F 16
CZNCCA.P11 PAGE 188

SEQ 0200

*CZNCCA,CZNCCA/CRF/SOL=CZNCCA
RUN-TIME: 43 31 6 SECONDS
CORE USED: 30K