

LSI-11

SUITCASE TEST
MD-11-DWQAA-A

EP-DWQAA-A-DL-A

OCT 1976

COPYRIGHT ©1976

digital

FICHE 1 OF 1

Made in U.S.A.

Frame	Frame	Frame	Frame	Frame
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DWGAA-A-D
PRODUCT NAME: LSI-11 SUITCASE 11M03-AA SYSTEM
DATE RELEASED: 21 APRIL 1976
MAINTAINER: DIAGNOSTIC GROUP

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH A LICENSE.

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

COPYRIGHT (C) 1976 DIGITAL EQUIPMENT CORPORATION

PROGRAM HISTORY

PRODUCT CODE: MAINDEC-11-DW0AA-A-D
PRODUCT NAME: LSI-11 SUITCASE 11W03-AA SYSTEM
DATE CREATED: DECEMBER 1975
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: JOHN EGOLF

COPYRIGHT (C) DECEMBER 1975
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754

1. ABSTRACT

MD-11-DMQAA-A IS A DIAGNOSTIC WHICH WAS WRITTEN TO TEST THE MS911 MODULE. THIS PROGRAM VERIFIES THE DATA WITHIN THE ROM, VERIFIES FUNCTIONS OF THE 600KHZ CLOCK, AND CHECKS THE BUS BUFFER BY MEANS OF THE TEST BOX PROVIDED WITH THE 11M03-AA SYSTEM. THIS PROGRAM WAS WRITTEN EXPRESSLY FOR THE 11M03-AA SYSTEM AND TAKES ADVANTAGE OF ALL UNITS WITHIN THE SYSTEM. THIS PROGRAM MUST BE RUN AFTER ALL CPU TESTS, MEMORY TESTS, AND DLV11 TESTS.

2. REQUIREMENTS

2.1 EQUIPMENT

11M03-AA SYSTEM, WHICH CONSISTS OF:

- A. KD11-F LSI-11 PROCESSOR (WITH 4K OF MEMORY)
- B. 12K MOS RAM MEMORY
- C. 2 DLV11 SERIAL LINE UNITS
- D. SPECIAL MODULE
 - MS911 - BUS BUFFER
 - BOOTSTRAP ROM
 - 600KHZ CLOCK
- E. H780 POWER SUPPLY
- F. ALUMINUM CASE 17 X 21 X 7.5"
- G. SPECIAL OPERATORS PANEL
- H. POWER CORD
- I. CABLE, EXTERNAL BUS -6FT.
- J. H9270 BACKPLANE ASSEMBLY
- K. TEST BOX

2.2 STORAGE

PROGRAM WILL LOAD IN 4K BUT RESERVES THE RIGHT TO USE 16K (ADDRESS 000000 - 077777).

3. LOADING PROCEDURE

THIS PROGRAM MAY BE LOADED LIKE ANY OTHER PROGRAM IS LOADED. THERE ARE NO SPECIAL LOADING PROCEDURES.

4. STARTING PROCEDURE

IF DESIRED, AFTER PROGRAM IS LOADED, SOFTWARE SWITCH REGISTER MAY BE SET AS PER SECTION 4.1 (USING LSI-00T) BEFORE START OF EXECUTION OF PROGRAM. AS DISCUSSED IN SECTION 4.2, THERE ARE TWO STARTING ADDRESSES:

SA 200 NORMAL START
 SA 210 NORMAL START BUT THE OPPORTUNITY TO "GOTO"
 A SELECTED TEST IS GIVEN TO THE USER.

4.1 CONTROL SWITCH SETTINGS

THE SOFTWARE SWITCH REGISTER MAY BE ALTERED BY USING LSI-00T AND MODIFYING ADDRESS 000176. AN EASIER WAY TO CHANGE THE SWITCH REGISTER IS THAT AFTER THE DIAGNOSTIC IS STARTED AND WHILE IT IS RUNNING THE USER MAY TYPE "CONTROL G" (<+G>) AND IF THE CPU PRIORITY ALLOWS (AND A RESET INSTRUCTION ISN'T BEING EXECUTED) THE PROGRAM WILL PRINT OUT:

(SMR)=/000000/=/

THE USER MAY THEN HIT <CR> CARRIAGE RETURN OR TYPE (IN OCTAL) NEW SETTINGS. SWITCHES ARE:

SW15	SET: HALT ON ERROR
SW14	SET: LOOP ON TEST
SW13	SET: INHIBIT ERROR PRINT-OUT.
SW12	SET: ESCAPE TO NEXT TEST ON ERROR.
SW11	SET: INHIBIT ITERATIONS
SW10	SET: BELL ON ERROR
SW09	SET: LOOP ON ERROR
SW08	SET: CONFIDENCE; -PRINT TEST # AT 1ST END OF EACH TEST.
SW07	RESERVED
SW06	RESERVED
SW05	RESERVED
SW04	RESERVED
SW03	RESERVED
SW02	RESERVED
SW01	RESERVED
SW00	RESERVED

4.1.2 SWITCH REGISTER RESTRICTIONS

NONE. FOR THE EXCEPTION OF SW12 ALL SWITCH REGISTER OPTIONS ARE "SYSMAC.SML" COMPATABLE.

4.2 STARTING ADDRESS

AS MENTIONED BEFORE THERE ARE TWO STARTING ADDRESSES:

SA 200 NORMAL OPERATION
 SA 210 NORMAL OPERATION WITH OPTION TO SPECIFY
 STARTING TEST NUMBER.

5. OPERATING PROCEDURE

PROGRAM SHOULD BE STARTED AS PER SECTION 4.2. ON INITIAL START PROGRAM WILL PRINT OUT:

MD-11-DW00A-A
LSI-11 SUIT-CASE ROM, 600 HZ, AND BUS TESTS

AND BEGIN EXECUTION (OR ASK FOR TEST NO. IF SA=210).

5.2 PROGRAM AND/OR OPERATOR ACTION

SWR SHOULD BE SET TO "100000" HALT ON ERROR. AFTER ERROR MODIFY SWR TO "060000" LOOP ON TEST AND INHIBIT PRINTOUT AND START AT ADDRESS 210 SPECIFYING FAILING TEST NUMBER.

6. ERRORS

ALL CONTROLABLE ERRORS WILL PRINT:

1) ERRPC	ERROR PC (IN OCTAL)
2) STSTNM	TEST NUMBER (IN OCTAL)
3) SPASS	PASSES MADE (IN DECIMAL)
4) SERTLL	ERRORS MADE (IN DECIMAL)

ALL PROCESSOR REGISTERS ARE SAVED IN MEMORY LOCATIONS ON ERRORS.

SAVR0	LOC: 1420
SAVR1	LOC: 1422
SAVR2	LOC: 1424
SAVR3	LOC: 1426
SAVR4	LOC: 1430
SAVR5	LOC: 1432

ADDITIONAL INFORMATION WILL BE PRINTED AS NEEDED.

7. RESTRICTIONS

7.1 HARDWARE RESTRICTIONS

SYSTEM MUST BE CONFIGURED #EXACTLY# AS OUTLINED FOR
11M03-AR SYSTEM.

8. MISCELLANEOUS

8.1 CONTROL "G" (<G>)

THIS WILL ENABLE USER TO CHANGE SWITCH REGISTER (PROVIDED
PROGRAM IS RUNNING).

8.2 CONTROL "T" (<T>)

THIS WILL ENABLE USER TO "GOTO" SELECTED TEST. (CPU MUST
NOT BE DOING A RESET.)

8.3 PASS COMPLETE

THIS IS A "SYSMAC.SHL" COMPATIBLE END PASS ROUTINE. AT
PASS COMPLETE THE FOLLOWING WILL BE PRINTED:

END PASS # 1
END PASS # 2

PASS COUNT IS IN DECIMAL.

8.4 EXECUTION TIME

WITH NO ERRORS AND SW11=0 PASS TIME IS <8 MINS.

WITH NO ERRORS AND SW11=1 PASS TIME IS <2 MINS.

4715	OPERATIONAL SWITCH SETTINGS
4716	BASIC DEFINITIONS
4717	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
4720	COMMON TAGS
(1)	ERROR POINTER TABLE
4935	T1 ROM "BREPLY" TEST
4961	T2 ROM WRITE TEST
4987	T3 ROM DATA TEST
5007	T4 600 HZ ADDRESS TEST
5038	T5 600 HZ "EVENT ENABLE" R/W TEST
5091	T6 600HZ INTERRUPT TEST
5138	T7 UNEXPECTED "BREPLY" TEST
5178	T10 "DEVICE 'A'" ADDRESS TEST
5217	T11 R/W TEST OF BIT 0 IN DEVICE 'A'
(4)	T12 R/W TEST OF BIT 1 IN DEVICE 'A'
(4)	T13 R/W TEST OF BIT 2 IN DEVICE 'A'
(4)	T14 R/W TEST OF BIT 3 IN DEVICE 'A'
(4)	T15 R/W TEST OF BIT 4 IN DEVICE 'A'
(4)	T16 R/W TEST OF BIT 5 IN DEVICE 'A'
(4)	T17 R/W TEST OF BIT 6 IN DEVICE 'A'
(4)	T20 R/W TEST OF BIT 7 IN DEVICE 'A'
(4)	T21 R/W TEST OF BIT 8 IN DEVICE 'A'
(4)	T22 R/W TEST OF BIT 9 IN DEVICE 'A'
(4)	T23 R/W TEST OF BIT 10 IN DEVICE 'A'
(4)	T24 R/W TEST OF BIT 11 IN DEVICE 'A'
(4)	T25 R/W TEST OF BIT 12 IN DEVICE 'A'
(4)	T26 R/W TEST OF BIT 13 IN DEVICE 'A'
(4)	T27 R/W TEST OF BIT 14 IN DEVICE 'A'
(4)	T30 R/W TEST OF BIT 15 IN DEVICE 'A'
5225	T31 BINARY COUNT TEST
5242	T32 1'S AND 0'S TEST
5264	T33 DEVICE 'A' BYTE OPERATIONS
5303	T34 BUS INITIALIZE TEST FOR DEVICE 'A'
5320	T35 "DEVICE 'B'" ADDRESS TEST (SEL 0)
5361	T36 R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 0)
(4)	T37 R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 0)
5372	T40 DEVICE 'B' INTERRUPT TEST (SEL 0)
5413	T41 "DEVICE 'B'" ADDRESS TEST (SEL 2)
5457	T42 R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 2)
(4)	T43 R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 2)
5468	T44 DEVICE 'B' INTERRUPT TEST (SEL 2)
5510	T45 TEST OF 'EVENT' AT DEVICE 'B'
5545	T46 DLV11 CHARACTER TEST
5624	T47 DLV11 INTERRUPT TEST
5678	T50 MINI MEMORY TEST
5758	END OF PASS ROUTINE
5759	SCOPE HANDLER ROUTINE
5760	ERROR HANDLER ROUTINE
5761	ERROR MESSAGE TIMEOUT ROUTINE
5762	TYPE ROUTINE
5763	BINARY TO OCTAL (ASCII) AND TYPE
5764	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5765	TTY INPUT ROUTINE
5766	READ AN OCTAL NUMBER FROM THE TTY
5768	TRAP DECODER

MAINDEC-11-11W03A-A MACY11 27(732) 24-AUG-76 16:05
DWQAA.P11 TABLE OF CONTENTS

SEQ 0008

(3) TRAP TABLE

(1) 000000
(1) 000040
(1) 000100
(1) 000140
(1) 000200
(1) 000240
(1) 000300
(1) 000340

PR0= 0
PR1= 40
PR2= 100
PR3= 150
PR4= 200
PR5= 240
PR6= 300
PR7= 340

... PRIORITY LEVEL 0
... PRIORITY LEVEL 1
... PRIORITY LEVEL 2
... PRIORITY LEVEL 3
... PRIORITY LEVEL 4
... PRIORITY LEVEL 5
... PRIORITY LEVEL 6
... PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

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

SW15= 100000
SW14= 400000
SW13= 200000
SW12= 100000
SW11= 40000
SW10= 20000
SW09= 10000
SW08= 4000
SW07= 2000
SW06= 1000
SW05= 400
SW04= 200
SW03= 100
SW02= 40
SW01= 2
SW00= 1

.EQUIV SW09, SW08, SW07, SW06, SW05, SW04, SW03, SW02, SW01, SW00

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

BIT15= 100000
BIT14= 400000
BIT13= 200000
BIT12= 100000
BIT11= 40000
BIT10= 20000
BIT09= 10000
BIT08= 4000
BIT07= 2000
BIT06= 1000
BIT05= 400
BIT04= 200
BIT03= 100
BIT02= 40
BIT01= 2
BIT00= 1

.EQUIV BIT09, BIT9
.EQUIV BIT08, BIT8

```

(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0

(1) ;#BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERVVEC= 4 ;TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;"T" BIT
(1) 000014 TRTVEC= 14 ;TRACE TRAP
(1) 000014 BPTVEC= 14 ;BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;POWER FAIL
(1) 000030 EMTVEC= 30 ;EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;"TRAP" TRAP
(1) 000060 TKVEC= 60 ;TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;PROGRAM INTERRUPT REQUEST VECTOR

4717 (1) .SBTTL TRAP CATCHER
(1)
(1) 000000 ;#0
(1) ;#ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1) ;#SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1) ;#LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)
(1) 000174 ;#174
(1) 000174 000000 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
(1) 000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER

(1) .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 003732 JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM
4718 (1) 000210 000210 ;#210
4719 (1) 000210 000137 003726 JMP SETTST ;ASK FOR TEST NUMBER.

```


4760	001262	003716	DF1	;OCTAL	OCTAL	DEC	DEC	OCTAL		
4761										
4762										
4763	001264	001741								
4764	001266	003210	EM10	;DEVICE "A" R/W FAILURE.						
4765	001270	003664	DH10	;ERRPC TSTNO PASCNT	ERRCNT	DEV.A	WANTED	FOUND		
4766	001272	003716	DT2	;SERPC MSKTST SPASS	SERTTL	SAVRD	SAVRS	SAVR4		
4767			DF1	;OCTAL OCTAL DEC	DEC	OCTAL	OCTAL	OCTAL		
4768										
4769	001274	001771								
4770	001276	003074	EM11	;NO "BREPLY" FROM DEVICE 'A'.						
4771	001300	003650	DH6	;ERRPC TSTNO PASCNT	ERRCNT	DEV.A				
4772	001302	003716	DT1	;SERPC MSKTST SPASS	SERTTL	SAVRD				
4773			DF1	;OCTAL OCTAL DEC	DEC	OCTAL				
4774										
4775	001304	002026								
4776	001306	003142	EM12	;NO "BREPLY" FROM DEVICE 'B'.						
4777	001310	003650	DH7	;ERRPC TSTNO PASCNT	ERRCNT	DEV.B				
4778	001312	003716	DT1	;SERPC MSKTST SPASS	SERTTL	SAVRD				
4779			DF1	;OCTAL OCTAL DEC	DEC	OCTAL				
4780										
4781	001314	002063								
4782	001316	003276	EM13	;DEVICE "B" R/W FAILURE.						
4783	001320	003664	DH11	;ERRPC TSTNO PASCNT	ERRCNT	DEV.B	WANTED	FOUND		
4784	001322	003716	DT2	;SERPC MSKTST SPASS	SERTTL	SAVRD	SAVRS	SAVR4		
4785			DF1	;OCTAL OCTAL DEC	DEC	OCTAL	OCTAL	OCTAL		
4786										
4787	001324	002113								
4788	001326	003364	EM14	;UNEXPECTED "BREPLY" FROM UNKNOWN DEVICE.						
4789	001330	003650	DH12	;ERRPC TSTNO PASCNT	ERRCNT	DEVICE				
4790	001332	003716	DT1	;SERPC MSKTST SPASS	SERTTL	SAVRD				
4791			DF1	;OCTAL OCTAL DEC	DEC	OCTAL				
4792										
4793	001334	002164								
4794	001336	003142	EM15	;NO INTERRUPT FROM DEVICE 'B'.						
4795	001340	003650	DH7	;ERRPC TSTNO PASCNT	ERRCNT	DEV.B				
4796	001342	003716	DT1	;SERPC MSKTST SPASS	SERTTL	SAVRD				
4797			DF1	;OCTAL OCTAL DEC	DEC	OCTAL				
4798										
4799	001344	002220								
4800	001346	003433	EM16	;GENERAL ERROR!						
4801	001350	003704	DH13	;ERRPC TSTNO PASCNT	ERRCNT					
4802	001352	003716	DT3	;SERPC MSKTST SPASS	SERTTL					
4803			DF1	;OCTAL OCTAL DEC	DEC					
4804										
4805	001354	002237								
4806	001356	003474	EM17	;DLV11 DATA ERROR.						
4807	001360	003664	DH14	;ERRPC TSTNO PASCNT	ERRCNT	DLV11	WANTED	FOUND		
4808	001362	003716	DT2	;SERPC MSKTST SPASS	SERTTL	SAVRD	SAVRS	SAVR4		
4809			DF1	;OCTAL OCTAL DEC	DEC	OCTAL	OCTAL	OCTAL		
4810										
4811	001364	002261								
4812	001366	003562	EM20	;MEMORY ERROR.						
4813	001370	003664	DH15	;ERRPC TSTNO PASCNT	ERRCNT	MEMORY	WANTED	FOUND		
4814	001372	003716	DT2	;SERPC MSKTST SPASS	SERTTL	SAVRD	SAVRS	SAVR4		
4815			DF1	;OCTAL OCTAL DEC	DEC	OCTAL	OCTAL	OCTAL		


```

4916 004420 012600
4917 004422 001773
4918 004424 020037 015140
4919 004430 101370
4920 004432 110037 001102
4921 004436 000241
4922 004440 006100
4923 004442 016037 015012 001106
4924 004444 000000
4925 004450 062737 000002 001106
4926 004456 013737 001106 001110
4927 004464 000177 174416

```

```

MOV (SP)+,R0
REQ X1$ ;BR IF NO NUMBER INPUT
CMP R0, LASTN ;VALID TEST?
BHI X1$ ;BR IF NO
MOV8 R0, STSTNM ;LOAD INITIAL TEST NO.
CLC
ROL R0 ;MAKE POWER OF 2.
MOV TEST.TABLE(R0), SLPERR ;GET TEST PC.
ADD #2, SLPADR ;GET PAST SCOPE
MOV SLPERR, SLPERR
JMP @SLPADR ;GOTO TEST.

```

```

(3)
(4)
(4)
(4)
(4)
(3)
(2)

```

```

*****
*TEST 1 ROM "BREPLY" TEST
*TEST TO REFERENCE ALL 256 ADDRESS
*OF THE ROM ONLY MAKING SURE THAT
*THERE IS A "BREPLY" RESPONSE - NO DATA IS
*CHECKED IN THIS TEST.
*****

```

```

4936 004470 000004
4937 004472 012700 173000
4938 004476 012701 000400
4939 013746 013746 013004
4940 013746 013746 013006
4941 004512 012737 004544 000004
4942 004520 012737 000200 000006
4943 004526 005710
4944 004530 005710
4945 004532 062700 000002
4946 004536 005301
4947 004540 011372
4948 004544 000404
4949 004548 104001
4950 004556 012716 004532
4951 004560 000002
4952 004564 012637 000006
4953 004568 012637 000004

```

```

TST1: SCOPE
MOV #173000, R0 ;SET INITIAL START ADDRESS OF ROM.
MOV #256, R1 ;SET NUMBER OF WORDS TO BE TESTED.
MOV 4, -(SP) ;SAVE LOC 4.
MOV 6, -(SP) ;SAVE LOC 6.
MOV #4, SLPERR ;SET TIME-OUT TRAP VECTOR.
MOV #00, 6 ;LOAD PRIORITY = 4
1$: TST (R0) ;REFERENCE ROM ADDRESS.
TST (R0) ;DO IT AGAIN.
2$: ADD #2, R0 ;UPDATE ADDRESS.
DEC R1 ;ALL DONE?
REQ X1$ ;BR IF NO
EX 4$ ;CONTINUE TEST
3$: ERROR 1 ;ROM DID NOT ISSUE "BREPLY".
MOV #2$, (SP) ;SET RETURN ADDRESS
RTI ;RETURN
4$: MOV (SP)+, 6 ;RESTORE ADDR 6.
MOV (SP)+, 4 ;RESTORE ADDR 4.

```

```

(3)
(4)
(4)
(4)
(3)
(2)

```

```

*****
*TEST 2 ROM WRITE TEST
*TEST THAT WRITING THE ROM
*PRODUCES A TIME-OUT TRAP.
*
*****

```

```

4962 004564 000004
4963 004566 012700 173000
4964 004572 012701 000400
4965 004576 013746 000004
4966 004602 013746 000006
4967 004606 012737 004632 000004
4968 004614 012737 000200 000006
4969 004622 005020
4970 004624 000240

```

```

TST2: SCOPE
MOV #173000, R0 ;SET ROM ADDRESS
MOV #256, R1 ;DO ALL ADDRESS
MOV 4, -(SP) ;STORE 4
MOV 6, -(SP) ;STORE 6
MOV #4, SLPERR ;SET TRAP VECTOR
MOV #200, 6 ;PRIORITY
1$: CLR (R0)+ ;WRITE ROM WITH ZERO.
NOP

```

```

4970 004625 104016
4971 004630 024646
4972 004632 105301
4973 004634 001403
4974 004636 012716 004622
4975 004642 000002
4976 004644 106427 000000
4977 004650 022626
4978 004652 012637 000006
4979 004656 012637 000004

```

```

                ERROR 16          ;WRITE ROM AND NO TRAP.
                CMP      -(SP),-(SP) ;FAKE AN INTERRUPT.
25:            DECB     R1          ;256 DONE?
                BEQ     35          ;IF YES
                MOV     #15,(SP)   ;SET RETURN
                RTI                    ;RETURN
35:            MTPS    #0          ;SET PTY TO 0
                CMP     (SP)+,(SP)+ ;FAKE RTI
                MOV     (SP)+,6    ;RESTORE 6
                MOV     (SP)+,4    ;RESTORE 4

```

```

4987 (3)
4988 (4)
4989 (4)
4990 (4)
4991 (4)
4992 (3)
4993 (2)

```

```

*****
*TEST 3      ROM DATA TEST
*TEST TO READ AND COMPARE ALL
*256 ROM ADDRESS TO THE ROM DATA
*IN MEMORY. ALL ADDRESSES ARE
*VERIFIED FOR GOOD DATA.
*****
TST3: SCOPE

```

```

4994 004662 000004
4995 004664 012700 173000
4996 004670 012701 000400
4997 004674 012702 013516
4998 004700 011004
4999 004702 011205
5000 004704 020504
5001 004706 001401
5002 004710 104002
5003 004712 022022
5004 004714 001301
5005 004716 001370

```

```

                MOV     #173000,R0 ;SET START OF ROM ADDRESS.
                MOV     #256,R1    ;SET NUMBER OF WORDS TO CHECK.
                MOV     #0,R2      ;GET SOFTWARE ADDRESS
15:            MOV     (R0),R4     ;READ ROM.
                MOV     (R2),R5     ;READ SOFTWARE IMAGE
                CMP     R5,R4       ;ARE THEY GOOD?
                BEQ     +4         ;ROM DATA ERROR?
                ERROR 2           ;BAD DATA IN ROMS!!
                CMP     (R0)+,(R2)+ ;POP POINTERS
                DEC     R1          ;ALL DONE?
                BNE     15         ;BR IF NOT DONE.

```

```

5006 (3)
5007 (4)
5008 (4)
5009 (4)
5010 (4)
5011 (4)
5012 (3)
5013 (2)

```

```

*****
*TEST 4      600 HZ ADDRESS TEST
*TEST TO VERIFY A "E EPLY" RESPONSE
*FROM THE 600 HZ CLOCK.
*IT IS ASSUMED THAT THE CLOCK
*IS AT ADDRESS "177546".
*****
TST4: SCOPE

```

```

5008 004720 000004
5009 004722 013746 000004
5010 004726 013746 000006
5011 004732 012701 000017
5012 004736 013700 001376
5013 004742 012737 004766 000004
5014 004750 012737 000200 000006
5015 004756 005710
5016 004760 005301
5017 004762 001375
5018 004764 000404
5019 004766 104003
5020 004770 012716 004760
5021 004774 000002
5022 004776 012637 000006
5023 005002 012637 000004

```

```

                MOV     4,-(SP)    ;SAVE ADDRESS 4.
                MOV     6,-(SP)    ;SAVE ADDRESS 6.
                MOV     #15,R1     ;SET INTERNAL ICOUNT TO 15.
                MOV     CLKCSR,R0  ;GET CLOCK CSR.
                MOV     #35,4      ;SET TIME-OUT VECTOR.
                MOV     #100,6     ;SET PRIO.
15:            TST     (R0)        ;REFERENCE CLOCK CSR.
25:            DEC     R1          ;ICOUNT = 0?
                BNE     15         ;BR IF NO.
                BR     45         ;CONT TEST
35:            ERROR 3           ;TIME-OUT ERROR.
                MOV     #25,(SP)   ;SET RETURN ADD.
                RTI                    ;RETURN.
45:            MOV     (SP)+,6    ;RESTORE 6
                MOV     (SP)+,4    ;RESTORE 4

```


5078	005204	022626	
5079	005206	012677	174170
5080	005212	012677	174162
5081	005216	106427	000000

```

55:  CMP      (SP)+,(SP)+
56:  MOV      (SP)+,@CLKPTY ;RESTORE CLKPTY
      MOV      (SP)+,@CLKVEC ;RESTORE CLKVEC
      MTPS     #0 ;SET Prio TO LVL 0

```

5090

5091
(3)
(4)
(4)
(4)
(4)
(4)
(3)
(2)

```

*****
*TEST 6      600HZ INTERRUPT TEST
*TEST THAT THE 600 HZ CLOCK
*CAN INTERRUPT AND THAT THE
*INTERPTS ARE WITHIN THE SAME
*TIME SPAN. 200. INTERRUPTS WILL BE
*EXECUTED
*****

```

5100	000004		
5101	017746	174150	
5102	017746	174146	
5103	012777	005310	174136
5104	012777	000200	174132
5105	012704	000310	
5106	012705	000002	
5107	005301		
5108	106427	000000	
5109	013700	001376	
5110	005377	000100	174076
5111	005377		
5112	005377		
5113	005377		
5114	005377		
5115	005377		
5116	005377		
5117	005377		
5118	005377		
5119	005377		
5120	005377		
5121	005377		
5122	005377		
5123	005377		
5124	005377		
5125	005377		
5126	005377		
5127	005377		
5128	005377		
5129	005377		
5130	005377		
5131	005377		
5132	005377		
5133	005377		
5134	005377		
5135	005377		
5136	005377		
5137	005377		
5138	005377		
5139	005377		
5140	005377		
5141	005377		
5142	005377		
5143	005377		
5144	005377		
5145	005377		
5146	005377		
5147	005377		
5148	005377		
5149	005377		
5150	005377		
5151	005377		
5152	005377		
5153	005377		
5154	005377		
5155	005377		
5156	005377		
5157	005377		
5158	005377		
5159	005377		
5160	005377		
5161	005377		
5162	005377		
5163	005377		
5164	005377		
5165	005377		
5166	005377		
5167	005377		
5168	005377		
5169	005377		
5170	005377		
5171	005377		
5172	005377		
5173	005377		
5174	005377		
5175	005377		
5176	005377		
5177	005377		
5178	005377		
5179	005377		
5180	005377		
5181	005377		
5182	005377		
5183	005377		
5184	005377		
5185	005377		
5186	005377		
5187	005377		
5188	005377		
5189	005377		
5190	005377		
5191	005377		
5192	005377		
5193	005377		
5194	005377		
5195	005377		
5196	005377		
5197	005377		
5198	005377		
5199	005377		
5200	005377		

```

15:  SCOPE
      MOV      @CLKVEC,-(SP) ;SAVE CLKVEC
      MOV      @CLKPTY,-(SP) ;SAVE CLKPTY
      MOV      #25,@CLKVEC ;SET INTERRUPT VECTOR
      MOV      #200,@CLKPTY ;SET Prio.
      MOV      #200,R4 ;SET ICOUNT
      MOV      #2,R5 ;SET COUNT POSITION
      CLR      R1 ;ZERO COUNT UP.
      MTPS     #0 ;CLEAR PSM
      MOV      @CLKCSR,R0 ;SAVE FOR PRINTOUT
      BIS      #BIT6,@CLKCSR ;SET EVENT ENABLE
      INC      R1 ;COUNT TIME
      BNE     18 ;NO INTERRUPT
      BR      18 ;EXIT TEST
25:  DEC      R5 ;HOW MANY INTERRUPTS?
      BEQ     38 ;BR IF 2ND!
      BPL     48 ;BR IF 1ST!
      MOV     R1,R2 ;MUST BE 3RD OR MORE!
      ROR     #10,R2 ;GIVE TOLARENCE OF +10
      SUB     R3,R2 ;GET DIFFERENCE
      BPL     48 ;SKIP IF POSITIVE
      NEG     R2 ;MAKE POSITIVE
      CMP     R2,#270 ;GIVE +270 TOLARENCE
      BLOS    7 ;INTERRUPTS OUT OF TOLERANCE
      BR      48 ;CONT TEST
35:  MOV     R1,R3 ;SAVE COUNT
45:  CLR     R1 ;ZERO COUNTER
      DEC     R4 ;ICOUNT =0?
      BEQ     55 ;BR IF YES
      RTI ;EXIT
55:  BIC     #BIT6,@CLKCSR ;CLEAR EVENT ENABLE
      CMP     (SP)+,(SP)+ ;POP INTR OFF STACK
      MOV     (SP)+,@CLKPTY ;RESTORE CLKPTY
      MOV     (SP)+,@CLKVEC ;RESTORE CLKVEC.
      MTPS     #0 ;SET Prio TO LEVEL 0

```

5120

5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200

```

*****
*TEST 7      UNEXPECTED "BREPLY" TEST.
*TEST TO SCAN THROUGH ALL ADDRESS

```

(3)
(4)

```

(4)
(4)
(4)
(4)
(4)
(3)
(2) 005402 000004
(1) 005404 012737 000003 001160
5139 005412 013746 000004
5140 005416 013746 000006
5141 005422 012737 005444 000004
5142 005430 005737 000006
5143 005436 005700
5144 005442 005710 15:
5145 005720
5146 000775
5147 00026 25:
5148 0023700 001374
5149 001403
5150 104016
5151 104400 002277
5152
5153 012737 005520 000004
5154 005710 35:
5155 012701 014650
5156 022100 45:
5157 001002
5158 062100
5159 00771
5160 005721 55:
5161 005510 005711
5162 001371
5163 104014
5164 000401
5165 022526
5166 062700 000002
5167 022700 177700
5168
5169 001356
5170 012637 000006
5171 012637 000004
5172
5173
(3)
(4)
(4)
(4)
(4)
(3)
(2) 005544 000004
5179 005546 013746 000004
5180 005552 013746 000006
5181 005556 012701 000017
5182 005562 013700 001404
5183 005566 012737 005612 000004
5184 005574 012737 000200 000006
5185 005602 005710

```

```

: #VERIFYING THAT ONLY EXPECTED DEVICES
: #RETURN "BREPLY". THIS TEST CHECKS
: #ONLY THAT "UNEXPECTED BREPLYS" AREN'T RECEIVED
: #NOT THAT ALL DEVICE ISSUE "BREPLY".
:
: *****
TST7: SCOPE
      MOV      #3,STIMES          ; DO 3 ITERATIONS
      MOV      4,-(SP)           ; SAVE 4
      MOV      6,-(SP)           ; SAVE 6
      MOV      #25,4             ; SET TIME-OUT TRAP
      CLR      6
      CLR      RO                ; SET FIRST ADDRESS TO 0
15:   TST      (RO)              ; READ ADDRESS
      TST      (RO)+             ; POP ADDRESS
      BR       15                ; CONTINUE TEST
25:   CMP      (SP)+,(SP)+       ; FAKE AN RTI
      CMP      MEMSIZ,RO         ; 16K ?
      BEQ      .+10              ; BR IF MEMORY SIZE IS CORRECT
      ERROR    16                ; INCORRECT MEMORY SIZE FOUND
      TYPE     ,EM21             ; REPORT CHANGE MSG.
      MOV      #65,4            ; LOC SAVRO HAS MEMORY FOUND.
      TST      (RO)              ; RESET TIME-OUT TRAP
      MOV      #EPLY.TABLE,R1    ; SET EXPECTED DEVICES
45:   CMP      (R1)+,RO          ; EXPECTED DEVICE?
      BNE      55                ; BR IF NO
      ROR      (R1)+,RO          ; POP MODULO OFFSET
      BR       35                ; CONT TEST
55:   TST      (R1)+             ; POP PAST OFFSET
      TST      (R1)              ; END OF TABLE?
      BNE      45                ; BR IF NO
      ERROR    14                ; UNEXPECTED "BREPLY" FROM DEVICE.
      BR       75                ; SKIP FAKE RTI
65:   CMP      (SP)+,(SP)+       ; FAKE RTI
75:   ROR      #2,RO             ; POP TO NEXT ADDRESS
      CMP      #177700,RO        ; ALL DONE?
      BNE      35
      MOV      (SP)+,6           ; RESTOE 6
      MOV      (SP)+,4           ; RESTORE 4

: *****
: #TEST 10 "DEVICE 'A'" ADDRESS TEST
: #TEST TO VERIFY A "BREPLY" RESPONSE
: #FROM "DEVICE 'A'".
: #IT IS ASSUMED THAT THE DEVICE
: #IS AT ADDRESS "172524".
: *****
TST10: SCOPE
      MOV      4,-(SP)           ; SAVE ADDRESS 4.
      MOV      6,-(SP)           ; SAVE ADDRESS 6.
      MOV      #15,R1            ; SET INTERNAL ICOUNT TO 15.
      MOV      DEV.A,RO          ; GET DEVICE CSR.
      MOV      #35,4             ; SET TIME-OUT VECTOR.
      MOV      #200,6            ; SET PRIO.
15:   TST      (RO)              ; REFERENCE DEVICE CSR.

```

```

5186 005604 005301
5187 005606 001375
5188 005610 000404
5189 005612 104011
5190 005614 012716 005604
5191 005620 000002
5192 005622 012637 000006
5193 005626 012637 000004
5194
5217

```

```

25: DEC R1 ;ICOUNT = 0?
    BNE 15 ;BR IF NO
    BR 45 ;CONT TEST
35: ERROR 11 ;TIME-OUT ERROR.
    MOV #25, (SP) ;SET RETURN ADD.
    RTI ;RETURN.
45: MOV (SP)+, 6 ;RESTORE 6
    MOV (SP)+, 4 ;RESTORE 4

```

```

*****
*TEST 11 R/W TEST OF BIT 0 IN DEVICE 'A'
*R/W TEST OF BIT 0 IN DEVICE 'A'.
*WRITE BIT 0 VERIFY ONLY BIT 0 IS SET.
*CLEAR BIT 0 VERIFY BIT 0 IS CLEARED.
*

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)
(3) 005632 000004
(1) 005634 013700 001404
(1) 005640 012705 000001
(1) 005644 010510
(1) 005646 011004
(1) 005650 020504
(2) 005652 001401
(1) 005654 104010
(1) 005656 040510
(1) 005660 011004
(1) 005662 042705 000001
(1) 005666 005704
(2) 005670 001401
(1) 005672 104010
(1)
(5)
(4)
(5)
(5)
(5)
(4)

```

```

*****
TST11: SCOPE
        MOV DEV.A, R0 ;LOAD DEVICE 'A' CSR.
        MOV #BIT0, R5 ;SET EXPECTED.
        MOV R5, (R0) ;WRITE BIT0
        MOV (R0), R4 ;READ CSR
        CMP R5, R4 ;BIT OK?
        BEQ .+4 ;BR IF OK
        ERROR 10 ;REGISTER HAS WRONG DATA.
        BIC R5, (R0) ;CLEAR BIT0
        MOV (R0), R4 ;READ REGISTER.
        BIC #BIT0, R5 ;ZERO EXPECTED
        TST R4 ;REGISTER OK?
        BEQ .+4 ;BR IF YES
        ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 12 R/W TEST OF BIT 1 IN DEVICE 'A'
*R/W TEST OF BIT 1 IN DEVICE 'A'.
*WRITE BIT 1 VERIFY ONLY BIT 1 IS SET.
*CLEAR BIT 1 VERIFY BIT 1 IS CLEARED.
*

```

```

(3) 005674 000004
(1) 005676 013700 001404
(1) 005702 012705 000002
(1) 005706 010510
(1) 005710 011004
(1) 005712 020504
(2) 005714 001401
(1) 005716 104010
(1) 005720 040510
(1) 005722 011004
(1) 005724 042705 000002
(1) 005730 005704
(2) 005732 001401
(1) 005734 104010
(1)
(5)
(4)

```

```

*****
TST12: SCOPE
        MOV DEV.A, R0 ;LOAD DEVICE 'A' CSR.
        MOV #BIT1, R5 ;SET EXPECTED.
        MOV R5, (R0) ;WRITE BIT1
        MOV (R0), R4 ;READ CSR
        CMP R5, R4 ;BIT OK?
        BEQ .+4 ;BR IF OK
        ERROR 10 ;REGISTER HAS WRONG DATA.
        BIC R5, (R0) ;CLEAR BIT1
        MOV (R0), R4 ;READ REGISTER.
        BIC #BIT1, R5 ;ZERO EXPECTED
        TST R4 ;REGISTER OK?
        BEQ .+4 ;BR IF YES
        ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 13 R/W TEST OF BIT 2 IN DEVICE 'A'

```

```

(5)
(5)
(5)
(5)
(4)
(3) 005736 000004
(1) 005740 013700 001404
(1) 005744 012705 000004
(1) 005750 010510
(1) 005752 011004
(1) 005754 020504
(2) 005756 001401
(1) 005760 104010
(1) 005762 040510
(1) 005764 011004
(1) 005766 042705 000004
(1) 005772 005704
(2) 005774 001401
(1) 005776 104010

```

```

;#R/W TEST OF BIT 2 IN DEVICE 'A'.
;#WRITE BIT 2 VERIFY ONLY BIT 2 IS SET.
;#CLEAR BIT 2 VERIFY BIT 2 IS CLEARED.
;#
;*****
†ST13: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT2,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT2
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BFO .+4 ;BR IF OK
EOR IO ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT2
MOV (R0),R4 ;READ REGISTER.
BIC #BIT2,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR IO ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)
(3) 006000 000004
(1) 006002 013700 001404
(1) 006006 012705 000010
(1) 006012 010510
(1) 006014 011004
(1) 006016 020504
(2) 006020 001401
(1) 006022 104010
(1) 006024 040510
(1) 006026 011004
(1) 006030 042705 000010
(1) 006034 005704
(2) 006036 001401
(1) 006040 104010

```

```

;*****
;#TEST 14 R/W TEST OF BIT 3 IN DEVICE 'A'
;#R/W TEST OF BIT 3 IN DEVICE 'A'
;#WRITE BIT 3 VERIFY ONLY BIT 3 IS SET.
;#CLEAR BIT 3 VERIFY BIT 3 IS CLEARED.
;#
;*****
†ST14: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT3,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT3
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR IO ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT3
MOV (R0),R4 ;READ REGISTER.
BIC #BIT3,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR IO ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)
(3) 006042 000004
(1) 006044 013700 001404
(1) 006050 012705 000020
(1) 006054 010510
(1) 006056 011004
(1) 006060 020504
(2) 006062 001401

```

```

;*****
;#TEST 15 R/W TEST OF BIT 4 IN DEVICE 'A'
;#R/W TEST OF BIT 4 IN DEVICE 'A'
;#WRITE BIT 4 VERIFY ONLY BIT 4 IS SET.
;#CLEAR BIT 4 VERIFY BIT 4 IS CLEARED.
;#
;*****
†ST15: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT4,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT4
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK

```

```

(1) 006064 104010
(1) 006066 040510
(1) 006070 011004
(1) 006072 042705 000020
(1) 006076 005704
(2) 006100 001401
(1) 006102 104010

```

```

ERROR 10 ;REGISTER HAS WRONG DATA.
BIC RS,(R0) ;CLEAR BIT4
MOV (R0),R4 ;READ REGISTER.
BIC @BIT4,RS ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 16 R/W TEST OF BIT 5 IN DEVICE 'A'
*R/W TEST OF BIT 5 IN DEVICE 'A'.
*WRITE BIT 5 VERIFY ONLY BIT 5 IS SET.
*CLEAR BIT 5 VERIFY BIT 5 IS CLEARED.
*

```

```

(3) 006104 000004
(1) 006106 013700 001404
(1) 006112 012705 000040
(1) 006116 010510
(1) 006120 011004
(1) 006122 020504
(2) 006124 001401
(1) 006126 104010
(1) 006130 040510
(1) 006132 011004
(1) 006134 042705 000040
(1) 006140 005704
(2) 006142 001401
(1) 006144 104010

```

```

TST16: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV @BIT5,RS ;SET EXPECTED.
MOV RS,(R0) ;WRITE BIT5
MOV (R0),R4 ;READ CSR
CMP RS,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC RS,(R0) ;CLEAR BIT5
MOV (R0),R4 ;READ REGISTER.
BIC @BIT5,RS ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 17 R/W TEST OF BIT 6 IN DEVICE 'A'
*R/W TEST OF BIT 6 IN DEVICE 'A'.
*WRITE BIT 6 VERIFY ONLY BIT 6 IS SET.
*CLEAR BIT 6 VERIFY BIT 6 IS CLEARED.
*

```

```

(3) 006146 000004
(1) 006150 013700 001404
(1) 006154 012705 000100
(1) 006160 010510
(1) 006162 011004
(1) 006164 020504
(2) 006166 001401
(1) 006170 104010
(1) 006172 040510
(1) 006174 011004
(1) 006176 042705 000100
(1) 006182 005704
(2) 006184 001401
(1) 006206 104010

```

```

TST17: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV @BIT6,RS ;SET EXPECTED.
MOV RS,(R0) ;WRITE BIT6
MOV (R0),R4 ;READ CSR
CMP RS,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC RS,(R0) ;CLEAR BIT6
MOV (R0),R4 ;READ REGISTER.
BIC @BIT6,RS ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 20 R/W TEST OF BIT 7 IN DEVICE 'A'
*R/W TEST OF BIT 7 IN DEVICE 'A'.
*WRITE BIT 7 VERIFY ONLY BIT 7 IS SET.

```

```

(1)
(5)
(4)
(5)
(5)

```

```

(5)
(5)
(4)
(3) 006210 000004
(1) 006212 013700 001404
(1) 006216 012705 000200
(1) 006222 010510
(1) 006224 011004
(1) 006226 020504
(2) 006230 001401
(1) 006232 104010
(1) 006234 040510
(1) 006236 011004
(1) 006240 042705 000200
(1) 006244 005704
(2) 006246 001401
(1) 006250 104010

```

```

;#CLEAR BIT 7 VERIFY BIT 7 IS CLEARED.
;#
;*****
†ST20: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT7,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT7
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT7
MOV (R0),R4 ;READ REGISTER.
BIC #BIT7,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)
(3) 006252 000204
(1) 006254 013700 001404
(1) 006256 012705 000400
(1) 006264 010510
(1) 006266 011004
(1) 006270 020504
(2) 006272 001401
(1) 006274 104010
(1) 006276 040510
(1) 006300 011004
(1) 006302 042705 000400
(1) 006306 005704
(2) 006310 001401
(1) 006312 104010

```

```

;*****
;#TEST 21 R/W TEST OF BIT 8 IN DEVICE 'A'
;#R/W TEST OF BIT 8 IN DEVICE 'A'.
;#WRITE BIT 8 VERIFY ONLY BIT 8 IS SET.
;#CLEAR BIT 8 VERIFY BIT 8 IS CLEARED.
;#
;*****
†ST21: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT8,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT8
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT8
MOV (R0),R4 ;READ REGISTER.
BIC #BIT8,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)
(3) 006314 000004
(1) 006316 013700 001404
(1) 006322 012705 001000
(1) 006326 010510
(1) 006330 011004
(1) 006332 020504
(2) 006334 001401
(1) 006336 104010
(1) 006340 040510

```

```

;*****
;#TEST 22 R/W TEST OF BIT 9 IN DEVICE 'A'
;#R/W TEST OF BIT 9 IN DEVICE 'A'.
;#WRITE BIT 9 VERIFY ONLY BIT 9 IS SET.
;#CLEAR BIT 9 VERIFY BIT 9 IS CLEARED.
;#
;*****
†ST22: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT9,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT9
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT9

```

```

(1) 006342 011004
(1) 006344 042705 001000
(1) 006350 005704
(2) 006352 001401
(1) 006354 104010

```

```

MOV (R0),R4 ;READ REGISTER.
BIC #BIT9,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ +4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 23 R/W TEST OF BIT 10 IN DEVICE 'A'
*R/W TEST OF BIT 10 IN DEVICE 'A'
*WRITE BIT 10 VERIFY ONLY BIT 10 IS SET.
*CLEAR BIT 10 VERIFY BIT 10 IS CLEARED.
*
*****

```

```

(3) 006356 000004
(1) 006360 013700 001404
(1) 006364 012705 002000
(1) 006370 010510
(1) 006372 011004
(1) 006374 020504
(2) 006376 001401
(1) 006400 104010
(1) 006402 040510
(1) 006404 011004
(1) 006406 042705 002000
(1) 006412 005704
(2) 006414 001401
(1) 006416 104010

```

```

*ST23: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT10,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT10
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
EQ +4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT10
MOV (R0),R4 ;READ REGISTER.
BIC #BIT10,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ +4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 24 R/W TEST OF BIT 11 IN DEVICE 'A'
*R/W TEST OF BIT 11 IN DEVICE 'A'
*WRITE BIT 11 VERIFY ONLY BIT 11 IS SET.
*CLEAR BIT 11 VERIFY BIT 11 IS CLEARED.
*
*****

```

```

(3) 006420 000004
(1) 006424 013700 001404
(1) 006428 012705 004000
(1) 006432 010510
(1) 006434 011004
(1) 006436 020504
(2) 006440 001401
(1) 006442 104010
(1) 006444 040510
(1) 006446 011004
(1) 006450 042705 004000
(1) 006454 005704
(2) 006456 001401
(1) 006460 104010

```

```

*ST24: SCOPE
MOV DEV,A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT11,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT11
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
EQ +4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT11
MOV (R0),R4 ;READ REGISTER.
BIC #BIT11,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ +4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

*****
*TEST 25 R/W TEST OF BIT 12 IN DEVICE 'A'
*R/W TEST OF BIT 12 IN DEVICE 'A'
*WRITE BIT 12 VERIFY ONLY BIT 12 IS SET.
*CLEAR BIT 12 VERIFY BIT 12 IS CLEARED.
*
*****

```

```

(1)
(5)
(4)
(5)
(5)
(5)

```

```

(4)
(3) 006462 000004
(1) 006464 013700 001404
(1) 006470 012705 010000
(1) 006474 010510
(1) 006476 011004
(1) 006500 020504
(2) 006502 001401
(1) 006504 104010
(1) 006506 040510
(1) 006510 011004
(1) 006512 042705 010000
(1) 006516 005704
(2) 006520 001401
(1) 006522 104010

```

```

*****
↑ST25: SCOPE
MOV DEV A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT12,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT12
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT12
MOV (R0),R4 ;READ REGISTER.
BIC #BIT12,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)

```

```

*****
#TEST 26 R/W TEST OF BIT 13 IN DEVICE 'A'
#R/W TEST OF BIT 13 IN DEVICE 'A'
#WRITE BIT 13 VERIFY ONLY BIT 13 IS SET.
#CLEAR BIT 13 VERIFY BIT 13 IS CLEARED.
#
*****

```

```

(3) 006524 000004
(1) 006526 013700 001404
(1) 006532 012705 020000
(1) 006536 010510
(1) 006540 011004
(1) 006542 020504
(2) 006544 001401
(1) 006546 104010
(1) 006550 040510
(1) 006552 011004
(1) 006554 042705 020000
(1) 006560 005704
(2) 006562 001401
(1) 006564 104010

```

```

↑ST26: SCOPE
MOV DEV A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT13,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT13
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT13
MOV (R0),R4 ;READ REGISTER.
BIC #BIT13,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

```

```

(5)
(4)
(5)
(5)
(5)
(5)
(4)

```

```

*****
#TEST 27 R/W TEST OF BIT 14 IN DEVICE 'A'
#R/W TEST OF BIT 14 IN DEVICE 'A'
#WRITE BIT 14 VERIFY ONLY BIT 14 IS SET.
#CLEAR BIT 14 VERIFY BIT 14 IS CLEARED.
#
*****

```

```

(3) 006566 000004
(1) 006570 013700 001404
(1) 006574 012705 040000
(1) 006600 010510
(1) 006602 011004
(1) 006604 020504
(2) 006606 001401
(1) 006610 104010
(1) 006612 040510
(1) 006614 011004
(1) 006616 042705 040000

```

```

↑ST27: SCOPE
MOV DEV A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT14,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT14
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT14
MOV (R0),R4 ;READ REGISTER.
BIC #BIT14,R5 ;ZERO EXPECTED

```

(1) 006622 005704
(2) 006624 001401
(1) 006626 104010

TST R4 : REGISTER OK?
BEQ +4 : BR IF YES
ERROR 10 : REGISTER NOT =0.

: TEST 30 R/W TEST OF BIT 15 IN DEVICE 'A'
: R/W TEST OF BIT 15 IN DEVICE 'A'
: WRITE BIT 15 VERIFY ONLY BIT 15 IS SET.
: CLEAR BIT 15 VERIFY BIT 15 IS CLEARED.
: *

(3) 006630 000004
(1) 006632 013700 001404
(1) 006634 012705 100000
(1) 006636 010510
(1) 006638 011004
(1) 006640 020504
(2) 006642 001401
(1) 006644 104010
(1) 006646 040510
(1) 006648 011004
(1) 006650 042705 100000
(1) 006652 005704
(2) 006654 001401
(1) 006656 104010

: TEST30: SCOPE
: MOV DEV.A,R0 : LOAD DEVICE 'A' CSR.
: MOV #BIT15,RS : SET EXPECTED.
: MOV RS,(R3) : WRITE BIT15
: MOV (R0),R4 : READ CSR
: CMP RS,R4 : BIT OK?
: BEQ +4 : BR IF OK
: ERROR 10 : REGISTER HAS WRONG DATA.
: BIC RS,(R0) : CLEAR BIT15
: MOV (R0),R4 : READ REGISTER.
: BIC #BIT15,RS : ZERO EXPECTED
: TST R4 : REGISTER OK?
: BEQ +4 : BR IF YES
: ERROR 10 : REGISTER NOT =0.

5224
5225
(3) 006672 000004
(1) 006674 012737 000003 001160
5226 006702 013700 001404
5227 006706 005005
5228 006710 010510
5229 006712 011004
5230 006714 020504
5231 006716 001401
5232 006720 104010
5233 006722 005205
5234 006724 001371

: TEST 31 BINARY COUNT TEST
: TEST TO WRITE A BINARY COUNT PATTERN
: ((000000-177777) THRU DEVICE 'A' REGISTER
: *
: *****
: TEST31: SCOPE
: MOV R3,STIMES : DO 3 ITERATIONS
: MOV DEV.A,R0 : GET DEVICE 'A' CSR.
: CLR RS : ZERO EXPECTED
18: : MOV RS,(R0) : LOAD DATA
: MOV (R0),R4 : READ DATA.
: CMP RS,R4 : IS DATA GOOD?
: BEQ +4 : IF YES
: ERROR 10 : REGISTER DATA ERROR.
: INC RS : INCR DATA
: BNE 18 : BR IF NOT DONE

5241
5242
(3) 006726 000004
(1) 006730 012737 000012 001160
5243 006736 013700 001404
5244 006742 005003

: TEST 32 1'S AND 0'S TEST
: TEST TO WRITE ALTERNATE 1'S AND 0'S
: ((125252 AND 052525) INTO DEVICE 'A').
: *
: *****
: TEST32: SCOPE
: MOV #10,STIMES : DO 10. ITERATIONS
: MOV DEV.A,R0 : GET DEVICE 'A' CSR
: CLR R3 : SET TO DO 128. TIMES

006744 012705
006746 010510
006748 011004
006750 020504
006752 001401
006754 104010
006756 005105
006758 105203
006760 100370

125252

```
18:  MOV      #125252,R5      ;LOAD DATA
      MOV      R5,(R0)      ;WRITE DATA
      MOV      (R0),R4      ;R5 TO REGISTER
      CMP      R5,R4        ;IS OK?
      BEQ      .+4           ;IF OK?
      ERROR   10           ;REGISTER R/W ERROR
      COM      R5           ;CHANGE DATA
      INCB    R3           ;DO 128 TIMES?
      BPL     18           ;BR IF NO.
```

```
*****
;TEST 33      DEVICE 'A' BYTE OPERATIONS
;TEST OF DEVICE 'A' BYTE OPEATIONS
;SET LOW BYTE OF DEVICE 'A' TO 0 AND
;SET HIGH BYTE OF DEVICE 'A' TO 3 THEN
;DO A 'INCB' ON LOW BYTE CHECK RESULTS
;DO A 'INCB' ON HIGH BYTE CHECK RESULTS
;CONTINUE TILL LOW BYTE GOES TO ZERO
*****
```

006770 000004
006772 013700
06776 017710
007002 005002
007004 012703
007010 010001
007012 005201
007014 105210
007016 111004
007020 105202
007022 110205
007024 020504
007026 001401
007030 104010
007032 011004
007034 110237
007040 110337
007044 013705
007050 020504
007052 001401
007054 104010
007056 105705
007060 001410
007062 105211
007064 111104
007066 105203
007070 110305
007072 020504
007074 001401
007076 104010
007100 000745
007102 000240

001404
001400

000003

001156
001157
001156

```
TST33:  SCOPE
        MOV      DEV.A,R0      ;LOAD DEVICE 'A' CSR
        MOV      R3,#400,(R0) ;PLACE 3 IN HIGH BYTE
        CLR      R2           ;ZERO LOW BYTE IMAGE
        MOV      R3,R3        ;SET HIGH BYTE IMAGE TO 3.
        MOV      R0,R1        ;GET CSR
        INC      R1           ;MAKE EQUAL TO HIGH BYTE.
        INCB    (R0)         ;ALTER LOW BYTE
        MOV      (R0),R4      ;READ IT
        INCB    R2           ;ALTER LOW BYTE IMAGE
        MOV      R2,R5        ;READ GOOD DATA
        MOV      R2,R4        ;IS DATA OK?
        CMP      R5,R4        ;IF YES
        BEQ      .+4           ;IF YES
        ERROR   10           ;REGISTER ERROR
        MOV      (R0),R4      ;READ WORD
        MOV      R2,$TMP0     ;GET LOW BYTE
        MOV      R3,$TMP0+1   ;GET HIGH BYTE
        INCB    $TMP0,R5      ;GOOD DATA
        CMP      R5,R4        ;STILL OK?
        BEQ      .+4           ;COMPARISON ERROR
        ERROR   10           ;IF LOW BYTE =0 THEN DONE
        TSTB   R5            ;BR IF END OF TEST
        BEQ      28           ;ALTER HIGH BYTE
        INCB    (R1)         ;READ IT
        MOV      (R1),R4      ;ALTER HIGH BYTE IMAGE
        MOV      R3,R5        ;LOAD GOOD DATA
        MOV      R3,R5        ;DATA OK?
        CMP      R5,R4        ;BR IF YES.
        BEQ      .+4           ;REGISTER ERROR
        ERROR   10           ;CONT TEST
        BR     18
28:     NOP
```

```
*****
;TEST 34      BUS INITIALIZE TEST FOR DEVICE 'A'
*****
```

F03

MAINDEC-11-11W03A-A
DW00A.P11 T34

MACY11 27(732) 24-AUG-76 16:05 PAGE 54-22
BUS INITIALIZE TEST FOR DEVICE 'A'

SEQ 0031

(4)
(4)
(4)
(3)
(2)
(1)
5305
5306
5307
(1)
5308
5309
5310
5311

007104 000004
007106 012737 000005 001160
007114 013700 001404
007120 012710 177777
007124 005005
007126 000005
007130 052777 000100 172004
007136 011004
007140 001401
007142 104010

: *TEST THAT ALL BITS IN DEVICE 'A'
: *REGISTER CAN BE CLEARED BY "INIT" (RESET INSTR).
: *
: *****
TST34: SCOPE
MOV #5, STIMES ; DO 5 ITERATIONS
MOV DEV.A, R0 ; GET CSA
MOV #1, (R0) ; SET ALL BITS
CLR RS ; SET EXPECTED TO ALL ZEROS
RESET ; ISSUE INIT
BIS #100, 2STKS ; SET INTR ENABLE
MOV (R0), R4 ; READ REGISTER
BEQ +4 ; BR IF ALL ZEROS
ERROR 10 ; REGISTER NOT ALL ZEROS

5319
(3)
(4)
(4)
(4)
(3)
(2)
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5361
(5)
(4)
(5)
(5)
(5)
(5)
(5)
(4)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4)
(5)
(5)
(5)
(5)

007144 000004
007146 013746 000004
007152 013746 000006
007156 012701 000017
007162 013700 001406
007166 012737 007212 000004
007174 012737 000200 000006
007202 005710
007204 005301
007206 001375
007210 005404
007212 104012
007214 012716 007204
007220 005002
007222 012637 000006
007226 012637 000004

```
*****
*TEST 35 "DEVICE 'B'" ADDRESS TEST (SEL 0)
*TEST TO VERIFY A "E EPLY" RESPONSE
*FROM "DEVICE 'B'" (SEL 0)
*IT IS ASSUMED THAT THE DEVICE
*IS AT ADDRESS "165250" (SEL 0).
*****
TST35: SCOPE
      MOV 4, -(SP)          ;SAVE ADDRESS 4.
      MOV 6, -(SP)          ;SAVE ADDRESS 6.
      MOV 15, R1            ;SET INTERNAL ICOUNT TO 15.
      MOV DEV.B, R0         ;GET DEVICE CSR.
      MOV #35, 4            ;SET TIME-OUT VECTOR.
      MOV #200, 6           ;SET PRIO.
15:   TST (R0)              ;REFERENCE DEVICE CSR.
25:   DEC R1                ;ICOUNT = 0?
      BNE 15                 ;BR IF NO.
      BR 45                  ;CONT TEST
35:   ERROR 12              ;TIME-OUT ERROR.
      MOV #25, (SP)         ;SET RETURN ADD.
      RTI                   ;RETURN.
45:   MOV (SP)+, 6          ;RESTORE 6
      MOV (SP)+, 4          ;RESTORE 4
*****
```

```
*****
*TEST 36 R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 0)
*TEST THAT BIT 6 IN DEVICE 'B' (SEL 0)
*IS R/W.
*SET BIT 6 VERIFY IT IS SET.
*CLEAR BIT 6 VERIFY IT IS CLEAR.
*****
TST36: SCOPE
      MTPS #200
      MOV DEV.B, R0         ;LOAD CSR INTO R0
      MOV #16, R5           ;SET BIT 6 INTO R5
      MOV R5, (R0)         ;WRITE BIT
      MOV (R0), R4         ;READ REGISTER
      CMP R5, R4           ;OK?
      EQ .+4               ;BR IF OK.
      ERROR 13             ;REGISTER R/W ERROR
      BIC R5, (R0)         ;CLEAR BIT 6
      MOV (R0), R4         ;READ DEVICE
      BIC #BIT6, R5        ;CLEAR EXPECTED
      TST R4               ;REGISTER =0?
      BQ .+4               ;BR IF =0!
      ERROR 13             ;REGISTER ERROR
*****
```

```
*****
*TEST 37 R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 0)
*TEST THAT BIT 7 IN DEVICE 'B' (SEL 0)
*IS R/W.
*SET BIT 7 VERIFY IT IS SET.
*CLEAR BIT 7 VERIFY IT IS CLEAR.
*****
```

```

(5)
(4)
(3) 007300 000004
(1) 007302 106427 000200
(1) 007306 013700 001406
(1) 007312 012705 000200
(1) 007316 010510
(1) 007320 011004
(1) 007322 020504
(2) 007324 001401
(1) 007326 104013
(1) 007330 040510
(1) 007332 011004
(1) 007334 042705 000200
(1) 007338 005704
(2) 007342 001401
(1) 007344 104013
5362
5371
5372
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 007346 000004
5373 007350 106427 000000
5374 007354 017746 172030
5375 007358 017746 172036
5376 007364 012777 007500 172016
5377 007372 012777 000200 172012
5378 007400 000001
5379 007402 013700 001406
007404 013710 000100
007406 013710 000100
007408 013710 000200
007410 013710 000200
007412 013710 000100
007414 013710 000100
007416 013710 000200
007418 013710 000200
007420 013710 000100
007422 013710 000200
007424 013710 000100
007426 013710 000200
007428 013710 000100
007430 013710 000200
007432 013710 000100
007434 013710 000200
007436 013710 000100
007438 013710 000200
007440 013710 000100
007442 013710 000200
007444 013710 000100
007446 013710 000200
007448 013710 000100
007450 013710 000200
007452 013710 000100
007454 013710 000200
007456 013710 000100
007458 013710 000200
5383 007460 000002
007462 013710 000100
007464 013710 000200
007466 013710 000100
007468 013710 000200
007470 013710 000100
007472 013710 000200
007474 013710 000100
007476 013710 000200

```

```

;*
*****
↑ST37: SCOPE
MTPS #200
MOV DEV.B,RO ;LOAD CSR INTO RO
MOV #BIT7,RS ;SET BIT 7 INTO RS
MOV RS,(R0) ;WRITE BIT
MOV (R0),R4 ;READ REGISTER
CMP RS,R4 ;OK?
BEQ .+4 ;BR IF OK.
ERROR 13 ;REGISTER R/W ERROR
BIC RS,(R0) ;CLEAR BIT 7
MOV (R0),R4 ;READ DEVICE
BIC #BIT7,RS ;CLEAR EXPECTED
TST R4 ;REGISTER =0?
P#0 .+4 ;ER IF =0!
ERROR 13 ;REGISTER ERROR

```

```

*****
*TEST 40 DEVICE 'B' INTERRUPT TEST (SEL 0)
*TEST OF DEVICE 'B' INTERRUPTS (SEL 0)
*SET BIT6 VERIFY NO INTERRUPT
*CLEAR BIT6; SET BIT7 -VERIFY NO INTERRUPT
*NOW SET BIT6 AGAIN -EXPECT INTERRUPT AT
*VECTOR "340".
*
```

```

*****
↑ST40: SCOPE
MTPS #0
MOV #28,AVEC,-(SP) ;SAVE DVB VECTOR
MOV #28,APTY,-(SP) ;SAVE DVB PRIO
MOV #28,AVEC ;LOAD VECTOR
MOV #200,28,APTY ;LOAD PRIO
CLR R1
MOV DEV.B,RO
15: BIS #BIT6,(R0) ;SET EVENT ENABLE
NOP ;WASTE TIME
BIC #BIT6,(R0) ;CLEAR IT
BIS #BIT7,(R0) ;SET OTHER INTR
NOP ;WASTE TIME
MOV #28,28,AVEC ;SET GOOD VECTOR
BIS #BIT6,(R0) ;SET IE
NOP ;WASTE TIME
ERROR 15 ;NO INTERRUPT
BR 45 ;CONT TEST
25: CLR (R0) ;ZERO REGISTER
INCB R3 ;UPDATE ICOUNT
BMI 35 ;DONE?
MOV #15,(SP) ;SET RETURN
RTI ;EXIT
35: CMP (SP)+,(SP)+ ;POP PC+PSW
45: CLR (R0) ;DISABLE DEVICE
MOV (SP)+,28,APTY ;RESTORE PTY
MOV (SP)+,28,AVEC ;RESTORE VEC
BR 65 ;CONT TEST

```


(1) 007656 104013
(1) 007660 106427 000000

ERROR 13 : REGISTER ERROR
MTPS 80 : SET PTY TO 0

(5) *****
(4) *TEST 43 R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 2)
(5) *TEST THAT BIT 7 IN DEVICE 'B' (SEL 2)
(5) *IS R/W.
(5) *SET BIT 7 VERIFY IT IS SET.
(5) *CLEAR BIT 7 VERIFY IT IS CLEAR.
(5) *
(4) *****

(3) 007664 000004
(1) 007666 106427 000200
(1) 007672 013700 001406
(1) 007676 062700 000002
(1) 007702 012705 000200
(1) 007706 010510
(1) 007710 011004
(1) 007712 020504
(2) 007714 001401
(1) 007716 104013
(1) 007720 040510
(1) 007722 011004
(1) 007724 042705 000200
(1) 007730 005704
(2) 007732 001401
(1) 007734 104013
(1) 007736 106427 000000

↑ST43: SCOPE
MTPS 8000
MOV DEV.B,R0 : LOAD CSR INTO R0
ADD #2,R0 : MAKE IT SEL 2
MOV #BIT7,RS : SET BIT 7 INTO RS
MOV RS,(R0) : WRITE BIT
MOV (R0),R4 : READ REGISTER
CMP RS,R4 : OK?
BFG +4 : BR IF OK.
ERROR 13 : REGISTER R/W ERROR
BIC RS,(R0) : CLEAR BIT 7
MOV (R0),R4 : READ DEVICE
BIC #BIT7,RS : CLEAR EXPECTED
TST R4 : REGISTER =0?
BFG +4 : BR IF =0!
ERROR 13 : REGISTER ERROR
MTPS 80 : SET PTY TO 0

5458
5467
5468
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2)
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483
5484

*TEST 44 DEVICE 'B' INTERRUPT TEST (SEL 2)
*TEST OF DEVICE 'B' INTERRUPTS (SEL 2)
*SET BIT6 VERIFY NO INTERRUPT
*CLEAR BIT6 SET BIT7 -V -IFY NO INTERRUPT
*R/W SET BIT6 AGAIN -EXPECT INTERRUPT AT
*VECTOR "344".

↑ST44: SCOPE
MTPS 80
MOV #29,BVEC,-(SP) : SAVE DVB VECTOR
MOV #0,BPTY,-(SP) : SAVE DVB PRIO
MOV #55,28,BVEC : LOAD VECTOR
MOV #200,28,BPTY : LOAD PRIO
CLR R1
MOV DEV.B,R0
ADD #2,R0 : MAKE IT SEL 2
BIS #BIT6,(R0) : SET EVENT ENABLE
NOP : WASTE TIME
BIC #BIT6,(R0) : CLEAR IT
BIS #BIT7,(R0) : SET OTHER INTR
NOP : WASTE TIME
MOV #25,28,BVEC : SET GOOD VECTOR
BIS #BIT6,(R0) : SET IE
NOP : WASTE TIME

000004
106427 000000
017746 171440
017746 171436
012777 010100 171426
012777 000200 171422
005001
013700 001406
062700 000002
02710 000100
060240
042710 000100
02710 000200
060240
012777 010046 171360
052710 000100
000240

IS:

5485	010042	104015		ERROR	15		: NO INTERRUPT
5486	010044	000407		BR	45		: CONT TEST
5487	010046	005010		CLR	(R0)		: ZERO REGISTER
5488	010050	105203		INCB	R3		: UPDATE ICOUNT
5489	010052	104403		BMI	35		: DONE?
5490	010054	012716	010006	MOV	#15, (SP)		: SET RETURN
5491	010060	000002		RTI			: EXIT
5492	010062	022626		CMP	(SP)+, (SP)+		: POP PC+PSW
5493	010064	005010		CLR	(R0)		: DISABLE DEVICE
5494	010066	012677	171324	MOV	(SP)+, @B.BPTY		: RESTORE PTY
5495	010072	012677	171316	MOV	(SP)+, @B.BVEC		: RESTORE VEC
5496	010076	000404		BR	65		: CONT TEST
5497	010100	010002		MOV	R0, R2		: SAVE R0
5498	010102	011600		MOV	(SP), R0		: SAVE PC
5499	010104	104005		ERROR	5		: UNEXPECTED INTERUPT
5500	010106	010200		MOV	R2, R0		: RESTORE R0
5501	010110	106427	000000	MTPS	#0		: ZERO PTY

5502
5509
5510
(3)
(F)
(F)
(F)
(F)
(3)
(2)
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5545
(3)

```

*****
: *TEST 45      TEST OF 'EVENT' AT DEVICE 'B'
: *TEST THAT SETTING BIT1 OF DEVICE 'B' (SEL 2)
: *CAUSES AN 'EVENT' VECTORING TO ADDRESS 100.
: *
: *
*****

```

5511	010114	000004		TST45: SCOPE			
5512	010116	106427	000000	MTPS	#0		: ZERO PSW
5513	010122	017746	171252	MOV	@CLKVEC, -(SP)		: SAVE VEC
5514	010126	017746	171250	MOV	@CLKPTY, -(SP)		: SAVE PTY
5515	010132	013700	001406	MOV	DEV.B, R0		: GET CSR
5516	010136	062700	000002	ADD	#2, R0		: MAKE IT SEL 2
5517	010142	005003		CLR	R3		: ICOUNTER=0
5518	010144	012777	010206 171226	MOV	#25, @CLKVEC		: SET VECTOR
5519	010152	012777	030200 171222	MOV	#200, @CLKPTY		: SET PRIO
5520	010160	005010		CLR	(R0)		: ZERO REGISTER
5521	010162	052710	000002	BIS	#BIT1, (R0)		: SET EVENT
5522	010166	005027		CLR	(PC)+		: STALL FOR TIME.
5523	010170	000000		O			
5524	010172	062737	000001 010170	ADD	#1, 645		: INC TIMER
5525	010200	001374		BNE	.-6		: TIMER DONE?
5526	010202	104015		ERROR	15		: NO INTERRUPT
5527	010204	000406		BR	45		: CONT
5528	010206	105203		INCB	R3		: ICOUNT=ICOUT+1
5529	010210	100403		BMI	35		: BR IF DONE
5530	010212	012716	010160	MOV	#15, (SP)		: SET RTN
5531	010216	000002		RTI			: EXIT
5532	010220	022626		CMP	(SP)+, (SP)+		: POP PC+PSW
5533	010222	005010		CLR	(R0)		: DISABLE DEVICE
5534	010224	012677	171152	MOV	(SP)+, @CLKPTY		: RESTORE PTY
5535	010230	012677	171144	MOV	(SP)+, @CLKVEC		: RESTORE VEC
5536	010234	106427	000000	MTPS	#0		: PTY 0

```

*****
: *TEST 46      DLV11 CHARACTER TEST

```

```

(4)
(4)
(4)
(4)
(4)
(3)
(2) 010240 000004
(1) 010242 012737 000002 001160
5546 010250 012700 175610
5547 010254 012701 000002
5548 010260 005003
5549 010262 005005
5550 010264 00760 007602
5551 010270 00760 007602
5552 010274 105760 000004
5553 010300 100407
5554 010302 023737 000000 000000
5555 010310 062703 000001
5556 010314 001367
5557 010316 104016
5558 010320 110560 000006
5559 010324 000003
5560 010326 100710
5561 010330 100407
5562 010332 023737 000000 000000
5563 010340 062703 000001
5564 010344 001370
5565 010346 104016
5566 010350 016004 000002
5567 010354 070504
5568 010356 001401
5569 010360 104017
5570 010362 105205
5571 010364 001343
5572 010366 005301
5573 010370 001341
5574 010372 105760 000004
5575 010376 100375
5576 010400 112760 000015 000006
5577 010406 105760 000004
5578 010412 100375
5579 010414 112760 000012 000006
5580 010422 012705 000005
5581 010426 012703 000040
5582 010430 000003
5583 010434 062703 000176
5584 010440 001772
5585 010442 012701 000110
5586 010446 010304
5587 010450 105760 000004
5588 010454 100375
5589 010456 112760 000015 000006
5590 010464 105760 000004
5591 010470 100375
5592 010472 112760 000012 000006
5593 010500 022704 000176

```

```

*TEST TO OUTPUT CHARACTERS ON THE DLV11
*NOT BEING USED AS THE CONSOLE INTERFACE.
*PROGRAM WILL DO 2 FULL BINARY COUNT PATTERNS
*AND THEN A SLIDE ASCII PATTERN FOR 5 ROWS.
*
*****
TST46: SCOPE
MOV #2,STIMES ;DO 2 ITERATIONS
MOV #175610,R0 ;SET DEVICE ADDRESS
MOV #2,R1 ;SET BINARY COUNTER
CLR R3
CLR R5 ;CLEAR COUNTER
TST 2(R0) ;CLR RX DONE
TST 2(R0)
15: TSTB 4(R0) ;PRINTER READY?
BMI 64$ ;BR IF YES
CMP 0,0 ;WASTE TIME
ADD #1,R3 ;DELAY COUNTER+1
ZNE 1$ ;DELAY DONE? NO!
ERROR 16 ;TPS BIT7 (>1 (NOT READY)
64$: MOVB R5,6(R0) ;LOAD DATA CHAR.
CLR R3 ;CLEAR POINTER
65$: TSTB (R0) ;RECEIVER READY(DONE)?
BMI 66$ ;BR IF YES
CMP 0,0 ;WASTE TIME.
ADD #1,R3
E E 65$ ;DELAY DONE? NO!
ERROR 16 ;RECEIVER NOT DONE.
66$: MOV 2(R0),R4 ;READ DATA
CMP R5,R4 ;DATA GOOD?
BEQ 67$ ;BR IF YES
ERROR 17 ;DATA COMPARE ERROR.
67$: INCB R5 ;NEXT CHAR
BNE 1$
DEC R1
E E 1$
TSTB 4(R0)
BPL .-4
5576: MOVB #15,6(R0)
TSTB 4(R0)
BPL .-4
5579: MOVB #12,6(R0)
MOV #5,R5
25: MOV #40,R3
35: INC R3
CMP #176,R3
BEQ 25
MOV #72,R1
MOV R3,R4
TSTB 4(R0)
BPL .-4
5589: MOVB #15,6(R0)
TSTB 4(R0)
BPL .-4
45: MOVB #12,6(R0)
CMP #176,R4

```

```

5594 010504 001002
5595 010506 012704 000040
5596 010512 105760 000004
5597 010516 100375
5598 010520 110460 000006
5599 010524 005204
5600 010528 005301
5601 010532 001362
5602 010536 005305
5603 010540 001336
5604 010544 105760 000004
5605 010548 160375
5606 010552 005360 000006
5607 010556 105760 000004
5608 010560 100375
5609 010564 005360 000006
5610 010568 105760 000004
5611 010572 100375
5612 010576 005005
5613 010580 005205
5614 010584 001376
5615 010588 005760 000002
5616
5623

```

```

BNE .+6
MOV #40,R4
TSTB 4(R0)
BPL .+1
MOVB R4,6(R0)
INC R4
DEC R4
BNE 45
DEC R4
BNE 35
TSTB 4(R0)
BPL .+1
CLR 6(R0)
TSTB 4(R0)
BPL .+1
CLR 6(R0)
TSTB 4(R0)
BPL .+1
CLR R5
INC R5
BNE .-2
TST 2(R0)

```

```

TP READY?
BR IF NO
XMIT A ZERO.
READY?
BR IF NO.
LOAD A ZERO.
READY?
BR IF NO
CLEAR RX FLAG.

```

```

(3)
(4)
(4)
(4)
(4)
(3)
(2)
(1)
5625 010602 000004
5626 010604 012737 000002 001160
5627 010612 012700 175610
5628 010616 013746 000300
5629 010622 013746 000302
5630 010626 013746 000304
5631 010632 013746 000306
5632 010636 012737 010732 000304
5633 010644 012737 000200 000306
5634 010652 012737 010760 000300
5635 010660 012737 000200 000302
5636 010666 005005
5637 010670 052760 000100 000004
5638 010676 005003
5639 010700 062703 000001
5640 010704 001375
5641 010706 104016
5642 010710 012637 000306
5643 010714 012637 000304
5644 010720 012637 000302
5645 010724 012637 000300
5646 010730 000444
5647 010732
5647 010732 110560 000006

```

```

*****
*TEST 47 DLV11 INTERRUPT TEST
*TEST OF DLV11 INTERRUPTS.
*THIS TEST WILL EXECUTE 32. INTERRUPTS ON
*BOTH THE RECEIVER AND TRANSMITTER AND
*WILL ALSO CHECK THE DATA.
*****
†ST47: SCOPE
MOV #2,STIMES
MOV #175610,R0
MOV 300,-(SP)
MOV 302,-(SP)
MOV 304,-(SP)
MOV 306,-(SP)
MOV #45,304
MOV #200,306
MOV #55,300
MOV #200,302
CLR R5
BIS #100,4(R0)
18: CLR P3
28: ADD #1,R3
BNE 28
ERROR 16
38: MOV (SP)+,306
MOV (SP)+,304
MOV (SP)+,302
MOV (SP)+,300
BR TST50
48: ;TRANSMITTER INTERRUPTS TO HERE.
MOVB R5,6(R0) ;LOAD DATA CHAR.

```

```

;DO 2 ITERATIONS
;GET DLV11 CSR
;SAVE
;VECTORS
;ON THE
;STACK.
;SET TX VECTOR.
;SET PRIO.
;SET RX VECTOR
;SET PRIO.
;SET DATA POINTER TO ZERO
;SET TX IE.
;SET TIMER TO ZERO.
;TIME WAITING FOR INTERRUPTS.
;KEEP COUNTING.
;DLV11 INTERRUPT PROBLEM.
;RESTORE
;ALL
;DLV11
;VECTORS.
;;EXIT TEST

```

```

5648 010736 042760 000100 000004      BIC      #100,4(R0)      ;DSABLE TX INTERUPTS.
5649 010744 052760 000100 000000      BIS      #100,0(R0)      ;ENABLE RX INTERUPTS.
5650 010752 012716 010676          MOV      #1$, (SP)      ;SET RETURN
5651 010756 000002          RTI                      ;EXIT ISR.
5652
5653 C:0760          5S:      ;RECEIVER INTERUPTS TO HERE.
5654 010760 000240          NOP
5655 010762 116004 000002      MOV      2(R0),R4      ;GET DATA REVEIVERO
5656 010766 020504          CMP      R5,R4      ;DATA GOOD?
5657 010770 001401          BEQ      6S          ;BE IF GOOD DATA
5658 010772 104017          ERROR   17          ;DLV11 DATA ERROR (INTERUPTS)
5659 010774 005205          6S:      INC      R5      ;UPDATE DATA CHAR.
5660 010776 022705 000040      CMP      #32.,R5      ;ALL CHARS DONE?
5661 011002 001006          BNE      7S          ;BR IF NO
5662 011004 042760 000100 000000      BIC      #100,0(R0)      ;DSABLE RX INTERUPTS.
5663 011012 012716 010710          MOV      #3$, (SP)      ;SET RETURN
5664 011016 000002          RTI                      ;RETURN
5665 011020 042760 000100 000000      7S:      BIC      #100,0(R0)      ;DSABLE RX INTERUPTS
5666 011026 052760 000100 000004      BIS      #100,4(R0)      ;ENABLE TX INTERUPTS
5667 011034 012716 010676          MOV      #1$, (SP)      ;SET RETURN
5668 011040 000002          RTI
5669
5677
5678 ;*****
(3) ;*TEST 50      MINI MEMORY TEST
(4) ;*MINI MEMORY TESTS
(4) ;*ALL UNSED MEMORY WILL BE FILLED WITH ALL 1'S
(4) ;*AND THEN EACH ADDR. SS WILL BE TESTED
(4) ;*WITH A FLOATING ZERO.
(4) ;*
(3) ;*****
(2) 011042 000004      †TST50: SCOPE
(1) 011044 012737 000002 001160      MOV      #2,$TIMES      ;DO 2 ITERATIONS
5679 011052 013701 001374          MOV      MEMSIZ,R1      ;GET MAX MEMORY SIZE.
5680 011056 162701 000310          SUB      #310,R1      ;PROTECT ABL
5681 011062 012700 015142          MOV      #CORMAX,R0      ;GET LAST ADDRESS USED BY PROGRAM
5682 011066 012720 177777          1S:      MOV      #177777,(R0)+
5683 011072 020100          CMP      R1,R0      ;ALL MEMORY FILLED?
5684 011074 001374          BNE      1S
5685 011076 012700 015142          MOV      #CORMAX,R0
5686 011102 042710 000001          2S:      BIC      #BIT0,(R0)
5687 011106 012705 177776          MOV      #1<1>,R5
5688 011112 011004          3S:      MOV      (R0),R4
5689 011114 020504          CMP      R5,R4
5690 011116 001401          BEQ      4S
5691 011120 104020          ERROR   20
5692 011122 000261          4S:      SEC
5693 011124 006105          ROL      R5
5694 011126 000261          SEC
5695 011130 006110          ROL      (R0)
5696 011132 103767          BCS      3S
5697 011134 005720          TST      (R0)+
5698 011136 100401          BMI      .+4
5699 011140 104016          ERROR   16      ;GENERAL ERROR. BIT15<>1!
5700 011142 020100          CMP      R1,R0      ;ALL MEMORY DONE?
5701 011144 001356          BNE      2S

```

804

MAINDEC-11-11403A-A
DWQAA.P11 TSO

MACY11 27(732) 24-AUG-76 16:05 PAGE 55-8
MINI MEMORY TEST

SEQ 0040

5702

(3)	011276	001425			BEQ	998			BR IF NO
(3)	011300	123737	001102	001440	CHPB	\$STNM, XXSCOP			WAS THIS TEST ALREADY TYPED OUT?
(3)	011306	001421			BEQ	998			BR IF YES
(3)	011310	123737	001102	015140	CHPB	\$STNM, LASTN			LEGAL TEST #
(3)	011316	003015			BGT	998			BR IF NO
(3)	011320	105737	001102		TSTB	\$STNM			TEST 0?
(3)	011324	001412			BEQ	998			BR IF YES (THERE IS NO TEST 0)
(3)	011326	113737	001102	001440	MOVB	\$STNM, XXSCOP			LOAD FOR NEXT TIME
(3)	011334	104400	014644		TYPE	XTST			TYPE "T".
(3)	011340	013746	001440		MOV	XXSCOP, -(SP)			
(3)	011344	104401			TYPOC				
(3)	011346	104400	014610		TYPE	, XSPA			
(3)	011352	032777	040000	167556	998:				
(1)	011352	001101			18:	BIT	8BIT14, 2SMR		:: LOOP ON PRESENT TEST?
(1)	011360	001101			BNE	\$OVER			:: YES IF SM14=1
(1)	011362	000416				START OF CODE FOR THE XOR			TESTER#####
(1)	011364	013746	000004		XTSTR: BR	68			:: IF RUNNING ON THE "XOR" TESTER CHANGE
(1)	011370	012737	011410	000004	MOV	2ERRVEC, -(SP)			THIS INSTRUCTION TO A "NOP" (NOP=240)
(1)	011376	005737	177060		MOV	855, 2ERRVEC			SAVE THE CONTENTS OF THE ERROR VECTOR
(1)	011402	012637	000004		TST	2177060			SET FOR TIMEOUT
(1)	011406	000453			MOV	(SP)+, 2ERRVEC			TIME OUT ON XOR?
(1)	011410	022626			BR	\$SVLAD			RESTORE THE ERROR VECTOR
(1)	011412	012637	000004		55:	CHP	(SP)+, (SP)+		GO TO THE NEXT TEST
(1)	011416	000413			MOV	(SP)+, 2ERRVEC			CLEAR THE STACK AFTER A TIME OUT
(1)	011420				BR	75			RESTORE THE ERROR VECTOR
(1)	011420	105737	001103		65:; #####	END OF CODE FOR THE XOR			TESTER#####
(1)	011424	001421			25:	TSTB	2ERFLG		:: HAS AN ERROR OCCURRED?
(1)	011426	123737	001115	001103	BEQ	35			BR IF NO
(1)	011434	101015			CHPB	2ERMAX, 2ERFLG			MAX. ERRORS FOR THIS TEST OCCURRED?
(1)	011436	032777	001000	167472	BHI	35			BR IF NO
(1)	011444	001404			BIT	8BIT09, 2SMR			LOOP ON ERROR?
(1)	011446	013737	001110	001106	BEQ	45			BR IF NO
(1)	011454	000443			75:	MOV	2LPERR, 2LPADR		:: SET LOOP ADDRESS TO LAST SCOPE
(1)	011456	105037	001103		BR	\$OVER			
(1)	011462	005037	001160		45:	CLRB	2ERFLG		:: ZERO THE ERROR FLAG
(1)	011467	000415			CLR	2TIMES			CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)	011470	032777	004000	167440	BR	15			ESCAPE TO THE NEXT TEST
(1)	011476	001011			35:	BIT	8BIT11, 2SMR		:: INHIBIT ITERATIONS?
(1)	011500	005737	001100		BNE	15			BR IF YES
(1)	011504	001406			TST	2PASS			IF FIRST PASS OF PROGRAM
(1)	011506	005237	001104		BEQ	15			INHIBIT ITERATIONS
(1)	011512	023737	001160	001104	INC	2ICNT			INCREMENT ITERATION COUNT
(1)	011520	002021			CHP	2TIMES, 2ICNT			CHECK THE NUMBER OF ITERATIONS MADE
(1)	011522	012737	000001	001104	BGE	\$OVER			BR IF MORE ITERATION REQUIRED
(1)	011530	013737	011600	001160	15:	MOV	81, 2ICNT		REINITIALIZE THE ITERATION COUNTER
(1)	011536	105237	001102		MOV	2ICNT, 2TIMES			SET NUMBER OF ITERATIONS TO DO
(1)	011542	011637	001106		SSVLAD:	INC	2STNM		COUNT TEST NUMBERS
(1)	011546	011637	001110		MOV	(SP), 2LPADR			SAVE SCOPE LOOP ADDRESS
(1)	011552	005037	001162		MOV	(SP), 2LPERR			SAVE ERROR LOOP ADDRESS
(1)	011556	112737	000001	001115	CLR	2ESCA E			CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)	011564	013777	001102	167346	MOVB	81, 2ERMAX			ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)	011572	013716	001106		SOVER:	MOV	2STNM, 2DISPLAY		DISPLAY TEST NUMBER
(1)	011576	000002			MOV	2LPADR, (SP)			FUDGE RETURN ADDRESS
(1)	011600	000100			RTI				FIXES PS
					SMXCNT:	100			MAX. NUMBER OF ITERATIONS


```

(3) 012016 020120          CMP      R1,(R0)+      ;LOOK FOR CURRENT TEST.
(3) 012017 001379          RMC      91$          ;BR IF NOT FOUND
(3) 012018 011037 012076  MOV      (R0),92$     ;SAVE TEST PC
(3) 012019 062737 000002 012076  ROD      R2,92$     ;POP PAST SCOPE
(3) 012020 012737 000001 001104  MOV      R1,$TCHT
(3) 012021 013737 011600 001160  MOV      $MXCNT,$TIMES
(3) 012022 012706 001100          MOV      @STACK,SP    ;SET SP
(3) 012023 012746 000000          MOV      R0,-(SP)     ;SET PTYO
(3) 012024 013746 012076          MOV      92$,-(SP)
(3) 012025 000137 011536          JMP      $SVLAD       ;GO INTO SCOPE ROUTINE
(3) 012070 012601          90$: MOV      (SP)+,R1    ;RESTORE R1
(3) 012072 012730          MOV      (SP)+,R0    ;RESTORE R0
(3) 012074 000000          RTI
(3) 012076 000000          92$: 0              ;EXIT
(3) 012076 000000          ;SAVE TEST PC

5761 012100 010037 001420  XERR1: MOV      R0,SAVR0
(1) 012104 010137 001422  MOV      R1,SAVR1
(1) 012110 010237 001424  MOV      R2,SAVR2
(1) 012114 010337 001426  MOV      R3,SAVR3
(1) 012120 010437 001430  MOV      R4,SAVR4
(1) 012124 010537 001432  MOV      R5,SAVR5
(1) 012130 113737 001102 001434  MOVB    $TINM,MSKTST
(3) ;*****
(2)
(2) .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
(2) ;#THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(2) ;#ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(2) ;#AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(2)
(2) 012136          SERRTYP:
(2) 012136 104400 001171      TYPE    $CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(2) 012142 010046          MOV      R0,-(SP)     ;;SAVE R0
(2) 012144 005300          CLR      R0           ;;PICKUP THE ITEM INDEX
(2) 012146 153700 001114      BISB    2*$ITEMB,R0
(2) 012152 001004          BNE     1$           ;; IF ITEM NUMBER IS ZERO, JUST
(3) 012154 013746 001116      MOV      SERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(3) 012160 104401          TYP0C          ;;SAVE SERRPC FOR TYPEOUT
(2) 012162 000445          BR      10$         ;;ERROR ADDRESS
(2) 012164 005300          1$: DEC      R0       ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(2) 012166 006300          ASL     R0          ;;GET OUT
(2) 012170 005300          ASL     R0          ;;ADJUST THE INDEX SO THAT IT WILL
(2) 012172 006300          ASL     R0          ;;WORK FOR THE ERROR TABLE
(2) 012174 062700 001174      ADD      @SERRTB,R0  ;;FORM TABLE POINTER
(2) 012200 012037 012210      MOV      (R0)+,2$    ;;PICKUP "ERROR MESSAGE" POINTER
(2) 012204 001404          BEQ     3$          ;;SKIP TYPEOUT IF NO POINTER
(2) 012206 104400          TYPE    $E           ;;TYPE THE "ERROR MESSAGE"
(2) 012210 000000          2$: .WORD 0         ;;"ERROR MESSAGE" POINTER GOES HERE
(2) 012212 104400 001171      TYPE    $CRLF        ;;"CARRIAGE RETURN" & "LINE FEED"
(2) 012216 012037 012226      MOV      (R0)+,4$    ;;PICKUP "DATA HEADER" POINTER
(2) 012222 001404          BEQ     5$          ;;SKIP TYPEOUT IF 0
(2) 012224 104400          TYPE    $H           ;;TYPE THE "DATA HEADER"
(2) 012226 000000          3$: .WORD 0         ;;"DATA HEADER" POINTER GOES HERE
(2) 012230 104400 001171      TYPE    $CRLF        ;;"CARRIAGE RETURN" & "LINE FEED"

```

```

(2) 012234 010146 58: MOV R1, -(SP) ;; SAVE R1
(2) 012236 012001 MOV (R0)+, R1 ;; PICKUP "DATA TABLE" POINTER
(2) 012240 001415 BEQ 98 ;; BR IF NO DATA TO BE TYPED
(2) 012242 012000 MOV (R0)+, R0 ;; PICKUP "DATA FORMAT" POINTER
(2) 012244 105720 68: TSTB (R0)+ ;; "OCTAL" OR "DECIMAL"
(2) 012246 001003 BNE 78 ;; BR IF DECIMAL
(2) 012250 013146 MOV 2(R1)+, -(SP) ;; SAVE 2(R1)+ FOR TYPEOUT
(2) 012252 104401 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(2) 012254 000402 BR 88
(2) 012256 013146 78: MOV 2(R1)+, -(SP) ;; SAVE 2(R1)+ FOR TYPEOUT
(2) 012258 104404 TYPOS ;; GO TYPE--DECIMAL ASCII WITH SIGN
(2) 012260 005711 88: TST (R1) ;; IS THERE ANOTHER NUMBER?
(2) 012262 001403 BEQ 98 ;; BR IF NO
(2) 012264 104400 012306 TYPE 118 ;; TYPE TWO(2) SPACES
(2) 012272 000764 BR 68 ;; LOOP
(2) 012274 012601 98: MOV (SP)+, R1 ;; RESTORE R1
(2) 012276 012600 108: MOV (SP)+, R0 ;; RESTORE R0
(2) 012300 104400 001171 TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
(2) 012304 000207 RTS PC ;; RETURN
(2) 012306 020040 000 118: .ASCIZ / / ;; TWO(2) SPACES
(2) 012312 012312 .EVEN

```

5762 ;*****

```

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

```

```

(1) 012312 105737 001155 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
(1) 012316 100002 BPL 18 ;; BR IF YI'S
(1) 012320 000000 HALT ;; HALT HERE IF NO TERMINAL
(1) 012322 000407 BR 38 ;; LEAVE
(1) 012324 010046 18: MOV R0, -(SP) ;; SAVE R0
(1) 012326 017600 000002 MOV 2(SP), R0 ;; GET ADDRESS OF ASCIZ STRING
(1) 012332 112046 28: MOVB (R0)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 012334 001005 BNE 48 ;; BR IF IT ISN'T THE TERMINATOR
(1) 012336 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
(1) 012340 012600 608: MOV (SP)+, R0 ;; RESTORE R0
(1) 012342 062716 38: ADD #2, (SP) ;; ADJUST RETURN PC
(1) 012346 000002 RTI ;; RETURN
(1) 012350 122716 000011 48: CMPB #HT, (SP) ;; BRANCH IF <HT>
(1) 012354 001426 BEQ 88
(1) 012356 122716 000200 CMPB #TCRLF, (SP) ;; BRANCH IF NOT <CRLF>

```

```

(1) 012362 001004 BNE 55
(1) 012364 005726 TST (SP)+ ;:POP (CR)<LF> EQUIV
(1) 012366 104400 TYPE ;:TYPE A CR AND LF
(1) 012370 001171 SCRLF
(1) 012372 000757 BR 25 ;:GET NEXT CHARACTER
(1) 012374 004737 012456 55: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
(1) 012400 123726 001154 65: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
(1) 012404 001352 BNE 25 ;:IF NO GO GET NEXT CHAR.
(1) 012406 013746 001152 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
(1) ;:AND THE NULL CHAR.
(1) 012412 105366 000001 75: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
(1) 012416 002770 BLT 65 ;:OR IF NO--GO POP THE NULL OFF OF STACK
(1) 012420 004737 012456 JSR PC,$TYPEC ;:GO TYPE A NULL
(1) 012424 105337 012522 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
(1) 012430 000770 BR 75 ;:LOOP

```

;HORIZONTAL TAB PROCESSOR

```

(1) 012432 112716 000040 85: MOVB #40,(SP) ;:REPLACE TAB WITH SPACE
(1) 012436 004737 012456 95: JSR PC,$TYPEC ;:TYPE A SPACE
(1) 012442 132737 000007 012522 BITB #7,$CHARCNT ;:BRANCH IF NOT AT
(1) 012450 001372 BNE 95 ;:TAB STOP
(1) 012452 005726 TST (SP)+ ;:POP SPACE OFF STACK
(1) 012454 000726 BR 25 ;:GET NEXT CHARACTER
(1) 012456 105777 166464 $TYPEC: TSTB @STPB ;:WAIT UNTIL PRINTER IS READY
(1) 012462 100375 BPL $TYPEC
(1) 012464 116677 000002 166456 MOVB 2(SP),@STPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 012472 122766 000015 000002 CMPB #15,2(SP) ;:BRANCH IF
(1) 012500 001003 BNE 15 ;:NOT <CR>
(1) 012502 105037 012522 CLRB $CHARCNT
(1) 012506 000406 BR $TYPEX ;:EXIT
(1) 012510 122766 000012 000002 15: CMPB #12,2(SP) ;:BRANCH IF
(1) 012516 002002 BGE $TYPEX ;:<LF>
(1) 012520 105227 INCB (PC)+ ;:INC SPACE
(1) 012522 070000 $CHARCNT: WORD 0 ;:COUNT
(1) 012524 000207 $TYPEX: RTS PC

```

```

;:
;: EQUATES
;: THT=11
;: TCRLF=200

```

;*****

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

;: *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;: *OCTAL (ASCII) NUMBER AND TYPE IT.
;: *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;: *CALL:
;: *   MOV   NUM,-(SP) ;:NUMBER TO BE TYPED
;: *   TYPOS ;:CALL FOR TYPEOUT
;: *   .BYTE N ;:N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;: *   .BYTE M ;:M=1 OR 0
;: * ;:1=TYPE LEADING ZEROS
;: * ;:0=SUPPRESS LEADING ZEROS
;: *
;: *$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST

```

5763

```

(1)      :#STYPOS OR STYPOC
(1)      :#CALL:
(1)      :#      MOV     NUM,-(SP)      :NUMBER TO BE TYPED
(1)      :#      TYPON                    :CALL FOR TYPEOUT
(1)      :#
(1)      :#STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)      :#CALL:
(1)      :#      MOV     NUM,-(SP)      :NUMBER TO BE TYPED
(1)      :#      TYPOC                    :CALL FOR TYPEOUT
(1)      :
(1) 012526 017646 000000      STYPOS: MOV     2(SP),-(SP)      :PICKUP THE MODE
(1) 012528 116637 000701 012751  MOVB    1(SP),SOFILL :LOAD ZERO FILL SWITCH
(1) 012530 112637 012753      MOVB    (SP)+,SOMODE+1 :NUMBER OF DIGITS TO TYPE
(1) 012532 062716 000002      ADD     #2,(SP)      :ADJUST RETURN ADDRESS
(1) 012534 000406      EX      STYPOC
(1) 012536 112737 000001 012751  STYPOC: MOVB    #1,SOFILL :SET THE ZERO FILL SWITCH
(1) 012538 112737 000006 012753  MOVB    #6,SOMODE+1 :SET FOR SIX(6) DIGITS
(1) 012540 112737 000005 012750  STYPON: MOVB    #5,COUNT :SET THE ITERATION COUNT
(1) 012542 010746      MOV     R3,-(SP)      :SAVE R3
(1) 012544 010446      MOV     R4,-(SP)      :SAVE R4
(1) 012546 010546      MOV     R5,-(SP)      :SAVE R5
(1) 012548 113704 012753      MOVB    #SOMODE+1,R4 :GET THE NUMBER OF DIGITS TO TYPE
(1) 012550 073404      NEG     R4
(1) 012552 06704 000006      ADD     #6,R4 :SUBTRACT IT FOR MAX. ALLOWED
(1) 012554 110437 012752      MOVB    R4,SOMODE :SAVE IT FOR USE
(1) 012556 113704 012751      MOVB    SOFILL,R4 :GET THE ZERO FILL SWITCH
(1) 012558 016605 000012      MOV     12(SP),R5 :PICKUP THE INPUT NUMBER
(1) 012560 07003      CLR     R3 :CLEAR THE OUTPUT WORD
(1) 012562 07105      15:    ROL     R5 :ROTATE MSB INTO "C"
(1) 012564 07104      25:    BR     R5 :GO DO MSB
(1) 012566 006105      35:    ROL     R5 :FORM THIS DIGIT
(1) 012568 006105      ROL     R5
(1) 012570 010503      MOV     R5,R3
(1) 012572 005103      35:    ROL     R3 :GET LSB OF THIS DIGIT
(1) 012574 10337 012752      DECB   SOMODE :TYPE THIS DIGIT?
(1) 012576 100016      BPL    #3 :IF NO
(1) 012578 042703 177770      BIC    #177770,R3 :GET RID OF JUNK
(1) 012580 001002      BNE    #4 :TEST FOR 0
(1) 012582 005704      TST    R4 :SUPPRESS THIS 0?
(1) 012584 001403      BEQ    #5 :BR IF YES
(1) 012586 005204      45:    INC    R4 :DON'T TYPE ANYMORE 0'S
(1) 012588 052703 000060      BIS    #'0,R3 :TEST FOR ASCII
(1) 012590 052703 000040      55:    BIS    #' ',R3 :TEST FOR ASCII IF NOT ALREADY
(1) 012702 110337 012746      MOVB   R3,R5 :SAVE FOR TYPING
(1) 012704 10470 012746      TYPE  #R5 :GO TYPE THIS DIGIT
(1) 012712 1037 012750      75:    DECB   #CNT :COUNT BY 1
(1) 012714 003347      EBT    #65 :IF MORE TO DO
(1) 012720 002402      EBT    #65
(1) 012722 005204      INC    R4
(1) 012724 000744      BR     #25
(1) 012726 010705      65:    MOV     (R4)+,R5
(1) 012730 010704      MOV     (R4)+,R4
(1) 012732 012603      MOV     (SP)+,R3
(1) 012734 016666 000002 000004      MOV     2(SP),4(SP)
(1) 012742 012616      MOV     (SP)+,(SP) :SET THE STACK FOR RETURNING

```

(1) 012744 000002
(1) 012746 000
(1) 012747 000
(1) 012750 000
(1) 012751 000
(1) 012752 000000

RTI RETURN
BS: .BYTE 0 STORAGE FOR ASCII DIGIT
.BYTE 0 TERMINATOR FOR TYPE ROUTINE
SOCNT: .BYTE 0 OCTAL DIGIT COUNTER
SOFILL: .BYTE 0 ZERO FILL SWITCH
SOMODE: .WORD 0 NUMBER OF DIGITS TO TYPE

5764

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNIFICANT DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDS ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*REPLACE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:

* MOV NUM, -(SP) ; PUT THE BINARY NUMBER ON THE STACK
* TYPDS ; GO TO THE ROUTINE

(2) 012754
(3) 012754 010046
(3) 012756 010146
(3) 012760 010246
(3) 012762 010346
(3) 012764 010546
(1) 012766 012746 020200
(1) 012772 016605 000020
(1) 012776 100004
(1) 013000 005405
(1) 013002 112766 000055 000001
(1) 013010 005000
(1) 013012 012703 013170
(1) 013016 112723 000040
(1) 013022 005002
(1) 013024 016001 013160
(1) 013030 160105
(1) 013032 002402
(1) 013034 005202
(1) 013036 000774
(1) 013040 050105
(1) 013042 005702
(1) 013044 001002
(1) 013046 105716
(1) 013050 100407
(1) 013052 106316
(1) 013054 103003
(1) 013056 116663 000001 177777
(1) 013064 052702 000060
(1) 013070 052702 000040
(1) 013074 110223
(1) 013076 005720
(1) 013100 020027 000010
(1) 013104 002746
(1) 013106 003002
(1) 013110 010502
(1) 013112 000764

STYPDS:
MOV R0, -(SP) ; PUSH R0 ON STACK
MOV R1, -(SP) ; PUSH R1 ON STACK
MOV R2, -(SP) ; PUSH R2 ON STACK
MOV R3, -(SP) ; PUSH R3 ON STACK
MOV R5, -(SP) ; PUSH R5 ON STACK
MOV #20200, -(SP) ; SET BLANK SWITCH AND SIGN
MOV 20(SP), R5 ; GET THE INPUT NUMBER
BPL 1\$; BR IF INPUT IS POS.
NEG R5 ; MAKE THE BINARY NUMBER POS.
MOVB #'-, 1(SP) ; MAKE THE ASCII NUMBER NEG.
1\$: CLR R0 ; ZERO THE CONSTANTS INDEX
MOV #SOBLK, R3 ; SETUP THE OUTPUT POINTER
MOVB #' ', (R3)+ ; SET THE FIRST CHARACTER TO A BLANK
2\$: CLR R2 ; CLEAR THE BCD NUMBER
MOV \$OTBL(R0), R1 ; GET THE CONSTANT
3\$: SUB R1, R5 ; FORM THIS BCD DIGIT
BLT 4\$; BR IF DONE
INC R2 ; INCREASE THE BCD DIGIT BY 1
BR 3\$
4\$: ADD R1, R5 ; ADD BACK THE CONSTANT
TST R2 ; CHECK IF BCD DIGIT=0
BNE 5\$; FALL THROUGH IF 0
TSTB (SP) ; STILL DOING LEADING 0'S?
BMI 7\$; BR IF YES
5\$: ASLB (SP) ; MSD
BCC 6\$; BR IF NO
MOVB 1(SP), -1(R3) ; YES--SET THE SIGN
6\$: BIS #'0, R2 ; MAKE THE BCD DIGIT ASCII
7\$: BIS #' ', R2 ; MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB R2, (R3)+ ; PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ; JUST INCREMENTING
CMP R0, #10 ; CHECK THE TABLE INDEX
BLT 2\$; GO DO THE NEXT DIGIT
BGT 8\$; GO TO EXIT
MOV R5, R2 ; GET THE LSD
BR 6\$; GO CHANGE TO ASCII

```

(1) 013114 105726          BS:   TSTB   (SP)+          ;; WAS THE LSD THE FIRST NON-ZERO?
(1) 013116 100003          BPL    9$              ;; BR IF NO
(1) 013120 116563 177777 177776  MOVB   -1(SP),-2(R3)   ;; YES--SET THE SIGN FOR TYPING
(1) 013126 100013          CLRB   (R3)           ;; SET THE TERMINATOR
(3) 013130 012605          MOV    (SP)+,R5       ;; POP STACK INTO R5
(3) 013132 012603          MOV    (SP)+,R3       ;; POP STACK INTO R3
(3) 013134 012602          MOV    (SP)+,R2       ;; POP STACK INTO R2
(3) 013136 012601          MOV    (SP)+,R1       ;; POP STACK INTO R1
(3) 013140 012600          MOV    (SP)+,R0       ;; POP STACK INTO R0
(1) 013142 104400 013170  TYPE   $OBLR          ;; NOW TYPE THE NUMBER
(1) 013146 016666 000002 000004  MOV    2(SP),4(SP)    ;; ADJUST THE STACK
(1) 013154 012616          MOV    (SP)+,(SP)
(1) 013156 000000          RTI
(1) 013160 023420          SOTBL: 10000.          ;; RETURN TO USER
(1) 013162 001750          1000.
(1) 013164 000144          100.
(1) 013166 000012          10.
(1) 013170 000004          SOBKR: .BLKW 4
5765 ;*****
(1) .SBTTL TTY INPUT ROUTINE
(1) ;; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;; CALL:
(1) ;; * ROCHR ;; INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;; * RETURN HERE ;; CHARACTER IS ON THE STACK
(1)
(1)
(1) 013200 011646          SROCHR: MOV   (SP)-,(SP) ;; PUSH DOWN THE PC
(1) 013202 016666 000004 000002  MOV   4(SP),2(SP) ;; SAVE THE PS
(1) 013210 105777 165726  IS:   TSTB   2$TKS      ;; WAIT FOR
(1) 013214 100375          BPL    1$              ;; A CHARACTER
(1) 013216 117766 165722 000004  MOVB   2$TKB,4(SP)   ;; READ THE TTY
(1) 013224 042766 177600 000004  BIC    8+C(177),4(SP) ;; GET RID OF JUNK IF ANY
(1) 013232 000002          RTI ;; GO BACK TO USER
(2) ;*****
(1) ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) ;; CALL:
(1) ;; * ROLIN ;; INPUT A STRING FROM THE TTY
(1) ;; * RETURN HERE ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1) ;; * ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1) 013234 010346          SROLIN: MOV   R3,-(SP) ;; SAVE R3
(1) 013236 012703 013342  IS:   MOV   8$TTYIN,R3 ;; GET ADDRESS
(1) 013242 022703 013352  2$:   CMP   8$TTYIN+8.,R3 ;; BUFFER FULL?
(1) 013246 101405          BLOS   4$              ;; BR IF YES
(1) 013250 104405          ROCHR  ;; GO READ ONE CHARACTER FROM THE TTY
(1) 013252 112613          MOVB   (SP)+,(R3)    ;; GET CHARACTER
(1) 013254 122713 00C177  CMPB   8$177,(R3)    ;; IS IT A RUBOUT
(1) 013260 001003          BNE    3$              ;; SKIP IF NOT
(1) 013262 104400 001170  4$:   TYPE  $QUES      ;; TYPE A '?'
(1) 013266 000763          BR     1$              ;; CLEAR THE BUFFER AND LOOP
(1) 013270 111337 013340  3$:   MOVB   (R3),9$      ;; ECHO THE CHARACTER
(1) 013274 104400 013340  TYPE   9$
(1) 013300 122723 000015  CMPB   8$15,(R3)+   ;; CHECK FOR RETURN
(1) 013304 001356          BNE    2$              ;; LOOP IF NOT RETURN

```

```

(1) 013306 105063 177777          CLRB    -1(R3)          ;; CLEAR RETURN (THE 15)
(1) 013312 104400 001172          TYPE    $LF            ;; TYPE A LINE FEED
(1) 013316 012603                 MOV     (SP)+,R3       ;; RESTORE R3
(1) 013320 011646                 MOV     (SP),-(SP)     ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 013322 016666 000004 000002   MOV     4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
(1) 013330 012766 013342 000004   MOV     $STTYIN,4(SP)
(1) 013336 000002                 RTI                    ;; RETURN
(1) 013340 000                    9$:     .BYTE    0          ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 013341 000                    .BYTE    0          ;; TERMINATOR
(1) 013342 000010                 STTYIN: .BLKB    8.    ;; RESERVE 8 BYTES FOR TTY INPUT
5766 ;*****
(1)                                     .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)                                     ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                     ;*CHANGE IT TO BINARY.
(1)                                     ;*CALL:
(1)                                     ;*   RDOCT
(1)                                     ;*   RETURN HERE
(1)                                     ;*   READ AN OCTAL NUMBER
(1)                                     ;*   LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                     ;*   HIGH ORDER BITS ARE IN SHIOCT
(1) 013352 011646 000004 000002   SRDOCT: MOV     (SP),-(SP)  ;; PROVIDE SPACE FOR THE
(1) 013354 016666                 MOV     4(SP),2(SP)    ;; INPUT NUMBER
(3) 013362 010046                 MOV     R0,-(SP)       ;; PUSH R0 ON STACK
(3) 013364 010146                 MOV     R1,-(SP)       ;; PUSH R1 ON STACK
(3) 013366 010246                 MOV     R2,-(SP)       ;; PUSH R2 ON STACK
(1) 013370 104406                 1$:     RDLIN          ;; READ AN ASCII LINE
(1) 013372 012600                 MOV     (SP)+,R0       ;; GET ADDRESS OF 1ST CHARACTER
(1) 013374 005001                 CLR     R1             ;; CLEAR DATA WORD
(1) 013376 005002                 CLR     R2
(1) 013378 112046                 2$:     MOVSB    (R0)+,-(SP)  ;; PICKUP THIS CHARACTER
(1) 013380 001412                 BEQ     3$             ;; IF ZERO GET OUT
(1) 013404 006301                 ASL    R1              ;; *2
(1) 013406 006102                 ROL    R2              ;; *4
(1) 013410 006301                 ASL    R1              ;; *8
(1) 013412 006102                 ROL    R2
(1) 013414 006301                 ASL    R1
(1) 013416 006102                 ROL    R2
(1) 013420 042716 177770         BIC     #C7,(SP)       ;; STRIP THE ASCII JUNK
(1) 013424 062601                 ADD     (SP)+,R1       ;; ADD IN THIS DIGIT
(1) 013426 000764                 BR     2$             ;; LOOP
(1) 013430 005726                 3$:     TST     (SP)+     ;; CLEAN TERMINATOR FROM STACK
(1) 013432 010166 000012         MOV     R1,12(SP)     ;; SAVE THE RESULT
(1) 013436 010237 013452         MOV     R2,SHIOCT
(3) 013442 012602                 MOV     (SP)+,R2       ;; POP STACK INTO R2
(3) 013444 012601                 MOV     (SP)+,R1       ;; POP STACK INTO R1
(3) 013446 012600                 MOV     (SP)+,R0       ;; POP STACK INTO R0
(1) 013450 000002                 RTI                    ;; RETURN
(1) 013452 000000                 SHIOCT: .WORD    0     ;; HIGH ORDER BITS GO HERE

```

```

5768 ;*****
(1)
(1) .SBTTL TRAP DECODER
(1)
(1) ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1) ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1) ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1) ;*GO TO THAT ROUTINE.
(1)
(1) 013454 010046 STRAP: MOV RO, -(SP) ;: SAVE RO
(1) 013456 016600 000002 MOV 2(SP), RO ;: GET TRAP ADDRESS
(1) 013462 005740 TST -(RO) ;: BACKUP BY 2
(1) 013464 111000 MOVB (RO), RO ;: GET RIGHT BYTE OF TRAP
(1) 013466 006300 ASL RO ;: POSITION FOR INDEXING
(1) 013470 016000 013476 MOV STRPAD(RO), RO ;: INDEX TO TABLE
(1) 013474 000200 RTS RO ;: GO TO ROUTINE
(1)
(3)
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) : -----
(3) $TRPAD:
(3) $TYPE ;: CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
(3) $TYPOC ;: CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) $TYPOS ;: CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) $TYPON ;: CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) $TYPOS ;: CALL=TYPOS TRAP+4(104404) TYPE DECIMAL NUMBER (WITH SIGN)
(3) $RDCHR ;: CALL=RDCHR TRAP+5(104405) TTY TYPEIN CHARACTER ROUTINE
(3) $RDLIN ;: CALL=RDLIN TRAP+6(104406) TTY TYPEIN STRING ROUTINE
(3) $RDOCT ;: CALL=RDOCT TRAP+7(104407) READ AN OCTAL NUMBER FROM TTY

```

5770	013516			
5778	013516	000400	000440	000167
5779	013534	102507	016005	000002
5780	013552	001100	120527	000101
5781	013570	126560	173064	000002
5782	013602	000	201	202
5783	013612	210	011	012
5784	013622	012700	175610	112710
5785	013640	004767	000142	020527
5786	013656	005705	001366	004767
5787	013674	004767	000174	010501
5788	013712	000240	003407	060205
5789	013730	000164	000733	004767
5790	013746	000772	105703	001403
5791	013764	000144	000723	004767
5792	014002	000126	010107	004767
5793	014020	042716	007760	004767
5794	014036	006305	042705	170017
5795	014054	032710	100200	001622
5796	014072	000207	004767	177706
5797	014110	105005	150405	000207
5798	014126	000102	000402	112705
5799	014144	110560	000006	000207
5800				
5801	014152	240	037	015
5802	014160	005004	012700	177500
5803	014176	012702	000375	112103
5804	014214	105202	100772	116012
5805	014232	005002	120312	001377
5806				

ROMMAP:

.WORD	000400,000440,000167,000432,132710,100200,001775
.WORD	102507,016005,000002,032705,177560,001406,005703
.WORD	001100,120527,000101,001075,000207,042705,177760
.WORD	126560,173064,000002,001066,000207
.BYTE	000,201,202,003,204,005,006,207
.BYTE	210,011,012,213,014,215,216,017
.WORD	012700,175610,112710,000006,12706,077776,005003
.WORD	004767,000142,020527,040001,001372,004767,00130
.WORD	005705,001366,004767,000206,010502,162702,000004
.WORD	004767,000174,010501,005702,001431,160705,022705
.WORD	000240,003407,060205,022705,177704,003003,004767
.WORD	000164,000733,004767,000046,005702,100402,110521
.WORD	000772,105703,001403,004767,000144,000720,004767
.WORD	000144,000723,004767,000012,105703,001366,004767
.WORD	000126,010107,004767,177514,010546,111666,000001
.WORD	042716,007760,004767,177476,006305,006305,006305
.WORD	006305,042705,170017,052605,060503,005302,000207
.WORD	032710,100200,001622,100734,016005,000002,100731
.WORD	000207,004767,177706,010504,004767,177700,000305
.WORD	105005,150405,000207,112705,000117,000405,112705
.WORD	000102,000402,112705,000015,105760,000004,100375
.WORD	110560,000006,000207
.BYTE	240,037,015,005,024,224
.WORD	005004,012700,177500,000005,010410,012701,173434
.WORD	012702,000375,112103,112110,100407,130310,001776
.WORD	105202,100772,116012,000002,000771,005710,100756
.WORD	005002,120312,001377,00112

DM00A.P11 TRAP TABLE

5812	014516	017746	164422
5813	014517	042716	000200
5814	014518	022716	000024
5815	014519	001004	
5816	014520	012706	001100
5817	014521	000137	004412
5818	014522	022726	000007
5819	014523	001016	
5820	014524	104400	014614
5821	014525	017746	164354
5822	014526	104401	
5823	014527	104400	014625
5824	014528	104401	
5825	014529	104400	
5826	014530	105737	012342
5827	014531	001400	
5828	014532	011677	164332
5829	014533	000000	
5830	014534	000000	
5831	014535	020040	000040
5832	014536	042000	052020
5833	014537	000000	000000
5834	014538	000000	027473
5835	014539	000000	042524
5836	014540	000000	000040
5837	014541	052200	000040
5838	014650		
5839	014650	165250	000004
5840	014651	172524	000004
5841	014660	173000	001000
5842	014664	177546	000002
5843	014670	177550	000010
5844	014674	177560	000010
5845	014700	175610	000010
5846	014704	000000	000000
5847	014710	000000	000000
5848	014714	000000	000000
5849	014720	000000	000000
5850	014724	000000	000000
5851	014730	000000	000000
5852	014734	011637	015006
5853	014740	162737	000002
5854	014746	017737	000034
5855	014754	022737	106400
5856	014762	001401	
5857	014764	000000	
5858	014766	011637	015006
5859	014772	062716	000002
5860	014776	017766	000004
5861	015004	000002	000002
5862	015006	000000	
5863	015010	000000	
5864	015012	000000	
5865	015012	000000	
5866	015014	004470	

024520
000
052123

```

KATSR: MOV 2$TKB, -(SP)
        BIC 2$BIT7, (SP)
        CMP 2$24, (SP)
        BNE 3$
        MOV 2$STACK, SP
        JNB X1$
        CMP 2$7, (SP)+
        BNE 1$
        TYPE X$SR
        MOV 2$SR, -(SP)
        TYPEPC ,EQUALS
        BDOCT $TTYIN
        BEQ 2$
        MOV (SP), 2$SR
        TST (SP)+
        RTI
X$SR: .ASCIZ /
X$SR: .ASCIZ <200>*(SR)=/
EQUALS: .ASCIZ =/
XTSTN: .ASCIZ <200>/TEST NO: /
XTST: .ASCIZ <200>/T /
.EVEN

```

```

:SAVE CHAR
:CLEAR BIT SEVEN
:WAS IT (IT) (TEST NO.)
:BR IF NO
:SET STACK
:GO TO ROUTINE
:WAS IT (IG)?
:BR IF NO
:SR=
:
:WAS JUST (CR) HIT?
:BR IF YES
:SET UP SR
:SET STACK
:EXIT

```

BRPLY. TABLE:

165250	004
172524	004
173000	512
177546	002
177550	010
177560	010
175610	010
027000	000
000000	000
000000	000
000000	000
000000	000
000000	000
000000	000
000000	000

```

:PRINT
DEBUG: .DUBG PC
        MOV (SP), 1$
        SUB 2$, 1$
        MOV 2$, 2$
        CMP 2$MTPS, 2$
        BEQ .+4
        HALT
        MOV (SP), 1$ ;REAL TRAP TO ADDRESS '4'.
        ADD 2$, (SP)
        MOV 2$, 2$(SP)
        RTI

```

```

1$: 0
2$: 0
TEST. TABLE:
0
TST1

```

;PC FOR TEST NUMBER 1

DISPRE	000174	4717#	4895																	
DSMR	= 177570	4716#	4720																	
DT1	003650	4724	4736	4747	4753	4759	4771	4777	4789	4795	4879#									
DT10	003716	4836#																		
DT2	003654	4730	4742	4765	4783	4807	4813	4880#												
DT3	003704	4801	4881#																	
DT4	003716	4832#																		
DT5	003716	4833#																		
DT6	003716	4834#																		
DT7	003716	4885#																		
EMVEC	= 000030	4716#	4895#																	
EM1	001442	4722	4843#																	
EM10	001741	4763	4850#																	
EM11	001771	4769	4851#																	
EM12	002026	4775	4852#																	
EM13	002063	4781	4853#																	
EM14	002113	4787	4854#																	
EM15	002164	4793	4855#																	
EM16	002220	4799	4856#																	
EM17	002237	4805	4857#																	
EM2	001500	4728	4844#																	
EM20	002261	4811	4858#																	
EM21	002277	4859#	5151																	
EM3	001533	4734	4845#																	
EM4	001572	4740	4846#																	
EM5	001624	4745	4847#																	
EM6	001651	4751	4848#																	
EM7	001701	4757	4849#																	
EQALS	014625	5823	5835#																	
EMVEC	= 000004	4716#	5759#																	
EMS	= ##### U	4717	4896	5768																
IOTVEC	= 000020	4716#	4895#	4905#																
KBISR	014516	4907	5812#																	
LASTN	015140	4918	5759	5888#																
MEMSIZ	001374	4817#	5148	5679																
MSKTST	001434	4836#	4879	4880	4881	5761#														
PC	= 000007	4716#	5052#	5521#	5758#	5760#	5761#	5762#												
PIR0	= 177772	4716#																		
PIR0VE	= 000240	4716#																		
PR0	= 000000	4716#																		
PR1	= 000040	4716#																		
PR2	= 000100	4716#																		
PR3	= 000140	4716#																		
PR4	= 000200	4716#																		
PR5	= 000240	4716#																		
PR6	= 000300	4716#																		
PR7	= 000340	4716#																		
PS	= 177776	4716#																		
PSM	= 177776	4716#																		
PIRVEC	= 000024	4716#																		
ROCHR	= 104405	5765	5768#																	
ROLIN	= 104406	5766	5768#																	
RODOCT	= 104407	4915	5768#	5824																
RESVEC	= 000010	4716#																		
ROMMAP	013516	4990	5770#	5776	5808															
RO	= 000000	4716#	4916#	4918	4920	4922#	4923	4936#	4942	4943	4944#	4962#	4968#	4988#						

000000	000100	47160			
000001	000100	47160			
000002	000100	47160			
000003	000100	47160			
000004	000100	47160			
000005	000100	47160			
000006	000100	47160			
000007	000100	47160			
000008	000100	47160			
000009	000100	47160			
000010	000100	47160			
000011	000100	47160			
000012	000100	47160			
000013	000100	47160			
000014	000100	47160			
000015	000100	47160			
000016	000100	47160			
000017	000100	47160			
000018	000100	47160			
000019	000100	47160			
000020	000100	47160			
000021	000100	47160			
000022	000100	47160			
000023	000100	47160			
000024	000100	47160			
000025	000100	47160			
000026	000100	47160			
000027	000100	47160			
000028	000100	47160			
000029	000100	47160			
000030	000100	47160			
000031	000100	47160			
000032	000100	47160			
000033	000100	47160			
000034	000100	47160			
000035	000100	47160			
000036	000100	47160			
000037	000100	47160			
000038	000100	47160			
000039	000100	47160			
000040	000100	47160			
000041	000100	47160			
000042	000100	47160			
000043	000100	47160			
000044	000100	47160			
000045	000100	47160			
000046	000100	47160			
000047	000100	47160			
000048	000100	47160			
000049	000100	47160			
000050	000100	47160			
000051	000100	47160			
000052	000100	47160			
000053	000100	47160			
000054	000100	47160			
000055	000100	47160			
000056	000100	47160			
000057	000100	47160			
000058	000100	47160			
000059	000100	47160			
000060	000100	47160			
000061	000100	47160			
000062	000100	47160			
000063	000100	47160			
000064	000100	47160			
000065	000100	47160			
000066	000100	47160			
000067	000100	47160			
000068	000100	47160			
000069	000100	47160			
000070	000100	47160			
000071	000100	47160			
000072	000100	47160			
000073	000100	47160			
000074	000100	47160			
000075	000100	47160			
000076	000100	47160			
000077	000100	47160			
000078	000100	47160			
000079	000100	47160			
000080	000100	47160			
000081	000100	47160			
000082	000100	47160			
000083	000100	47160			
000084	000100	47160			
000085	000100	47160			
000086	000100	47160			
000087	000100	47160			
000088	000100	47160			
000089	000100	47160			
000090	000100	47160			
000091	000100	47160			
000092	000100	47160			
000093	000100	47160			
000094	000100	47160			
000095	000100	47160			
000096	000100	47160			
000097	000100	47160			
000098	000100	47160			
000099	000100	47160			
000100	000100	47160			
TBITVE	000014	47160			
TGALF	000200	57620			
TEST.T	015012	4923	5760	58710	
THT	000011	57620			
TKVEC	000060	47160			
TPVEC	000064	47160			
TRAPVE	000034	47160	4895*		
TRTVEC	000014	47160			
TSTFLG	001436	48370	4894*	4911	4913*
TST1	004470	4912	49350	5758	5883
TST10	005544	51780	5883		
TST11	005632	52170	5883		
TST12	005674	52170	5883		
TST13	005736	52170	5883		
TST14	006000	52170	5883		
TST15	006042	52170	5883		
TST16	006104	52170	5883		
TST17	006146	52170	5883		
TST2	004564	49610	5883		
TST20	006210	52170	5883		
TST21	006252	52170	5883		
TST22	006214	52170	5883		
TST23	006356	52170	5883		
TST24	006420	52170	5883		
TST25	006462	52170	5883		
TST26	006524	52170	5883		
TST27	006566	52170	5883		
TST3	004662	49870	5883		
TST30	006630	52170	5883		
TST31	006672	52250	5883		
TST32	006726	52420	5883		
TST33	006770	52640	5883		
TST34	007104	53030	5883		
TST35	007144	53200	5883		
TST36	007232	53610	5883		
TST37	007300	53610	5883		
TST4	004720	50070	5883		

STYPOS	012526	5763#	5768											
SXTSTR	011362	5759#												
SOFILL	012751	5763##												
S4OCAT=	***** U	5759	5760											
.	= 015144	4717#	4718#	4720#	4895	4896#	4994	5050	5055	5060	5075	5112	5115	5149
		5217	5231	5249	5276	5283	5292	5309	5361	5457	5524	5575	5578	5588
		5591	5594	5597	5605	5608	5611	5614	5638	5758	5759	5760	5761#	5762
		5764#	5765#	5776#	5808#	5863								

.SAPTH	4420#		
.SAPTY	4583#		
.SASTA	4467#		
.SCATC	478#	4702#	4717
.SCHTA	580#	4702#	4720
.SD820	3736#		
.SD820	3860#		
.SDIV	3631#		
.SEOP	1584#	4702#	5758
.SERR0	2002#	4702#	5760
.SERRT	2195#	4702#	5761
.SMULT	3567#		
.SPOWE	3280#		
.SRAND	3343#		
.SR00E	2973#		
.SR00C	2881#	4704#	5766
.SREAO	2665#	4704#	5765
.SF2A2	4004#		
.SSAVE	3049#		
.SS820	3821#		
.SS820	3923#		
.SSCOP	1794#	4703#	5759
.SSIZE	3405#		
.SSUPR	3961#		
.STRAP	3149#	4703#	5768
.STYPB	2580#		
.STYPD	2502#	4703#	5764
.STYPE	2283#	4703#	5762
.STYPO	2405#	4703#	5763
.S40CA	509#		

MOS

ADD	4925	4944	5110	5158	5166	5418	5457	5476	5515	5523	5555	5563	5637	5760	5761
	5762	5763	5764	5766	5866										
ASL	5761	5766	5768												
ASLB	5764														
BCC	5764														
BCS	5696														
BEQ	4896	4912	4917	4973	4994	5050	5060	5107	5121	5149	5217	5231	5249	5276	5283
	5286	5292	5309	5361	5457	5568	5584	5657	5690	5758	5759	5760	5761	5762	5763
	5766	5826	5863												
BGE	5759	5762													
BGT	5758	5759	5763	5764											
BHI	4919	5759													
BIC	5056	5123	5217	5361	5382	5457	5479	5648	5662	5665	5686	5758	5763	5765	5766
	5813														
BIS	5101	5307	5380	5383	5386	5477	5480	5483	5520	5635	5649	5666	5760	5763	5764
BISB	5761														
BIT	5759	5760													
BITB	5762														
BLOS	5115	5765													
BLT	5762	5763	5764												
BMI	5392	5489	5528	5553	5561	5698	5764								
BNE	4895	4896	4946	4998	5016	5055	5063	5075	5103	5157	5162	5168	5187	5234	5329
	5423	5524	5556	5564	5571	5573	5594	5601	5603	5614	5638	5661	5684	5701	5759
	5760	5761	5762	5763	5764	5765	5815	5819							
BPL	5108	5112	5253	5575	5578	5588	5591	5597	5605	5608	5611	5760	5762	5763	5764
	5765														
BR	4895	4896	4947	5017	5064	5077	5105	5117	5146	5159	5164	5188	5294	5330	5389
	5399	5424	5486	5496	5526	5644	5759	5761	5762	5763	5764	5765	5766		
CLC	4921														
CLR	4895	4898	4902	4904	4905	4906	4913	4968	5058	5070	5098	5119	5142	5143	5227
	5244	5267	5306	5378	5390	5396	5474	5487	5493	5516	5519	5521	5532	5548	5549
	5559	5606	5609	5612	5634	5636	5758	5759	5761	5763	5764	5766			
CLRB	5759	5762	5764	5765											
CMP	4895	4896	4918	4971	4977	4993	4996	5049	5078	5114	5124	5147	5148	5156	5165
	5167	5217	5230	5248	5275	5282	5291	5361	5395	5457	5492	5531	5554	5562	5567
	5583	5593	5656	5660	5683	5689	5700	5759	5760	5764	5765	5814	5818	5862	
CMPB	5759	5762	5765												
COM	5251														
DEC	4945	4997	5015	5054	5062	5106	5120	5186	5328	5422	5572	5600	5602	5758	5761
DECB	4972	5762	5763												
EHT	4716														
HALT	4717	5760	5762	5864											
INC	4894	4896	5074	5102	5233	5270	5582	5599	5613	5659	5758	5759	5760	5763	5764
INCB	5252	5271	5273	5287	5289	5391	5488	5527	5570	5759	5760	5762			
IOT	4716														
JMP	4717	4719	4927	5758	5760	5817									
JSR	5758	5760	5762												
MOV	4895	4897	4899	4900	4901	4903	4907	4908	4909	4916	4923	4926	4936	4937	4938
	4939	4940	4941	4949	4951	4952	4962	4963	4964	4965	4966	4967	4974	4978	4979
	4988	4989	4990	4991	4992	5008	5009	5010	5011	5012	5013	5019	5021	5022	5038
	5040	5041	5042	5043	5044	5045	5046	5047	5048	5052	5057	5065	5067	5068	5071
	5072	5079	5080	5092	5093	5094	5095	5096	5097	5100	5109	5118	5125	5126	5138
	5139	5140	5141	5153	5155	5169	5170	5179	5180	5181	5182	5183	5184	5190	5192
	5193	5217	5225	5226	5228	5229	5242	5243	5245	5246	5247	5265	5266	5268	5269
	5278	5281	5303	5304	5305	5308	5321	5322	5323	5324	5325	5326	5332	5334	5335
	5361	5374	5375	5376	5377	5379	5385	5393	5397	5398	5400	5401	5403	5414	5415

	5416	5417	5419	5420	5426	5428	5429	5457	5470	5471	5472	5473	5475	5482	5490
	5494	5495	5497	5498	5500	5512	5513	5514	5517	5518	5529	5533	5534	5545	5546
	5547	5566	5580	5581	5585	5586	5595	5624	5625	5626	5627	5628	5629	5630	5631
	5632	5633	5640	5641	5642	5643	5650	5663	5667	5678	5679	5681	5682	5685	5687
	5688	5758	5759	5760	5761	5762	5763	5764	5765	5766	5768	5812	5816	5821	5827
	5859	5861	5865	5867											
MOV8	4895	4920	5272	5274	5279	5280	5288	5290	5558	5576	5579	5589	5592	5598	5647
MTPS	5655	5759	5760	5761	5762	5763	5764	5765	5766	5768					
	4910	4976	5039	5073	5081	5099	5127	5361	5373	5404	5457	5469	5501	5511	5535
	5760	5862													
NEG	5113	5763	5764												
NOP	4969	5295	5381	5384	5387	5478	5481	5484	5654	5758					
RESET	5307	5758	5760												
ROL	4922	5693	5695	5763	5766										
RTI	4950	4975	5020	5069	5122	5191	5333	5394	5427	5491	5530	5651	5664	5668	5759
	5760	5762	5763	5764	5765	5766	5829	5868							
RTS	5761	5762	5768												
SEC	5692	5694													
SUB	5111	5680	5760	5764	5860										
TRAP	5768														
TST	4895	4911	4942	4943	5014	5059	5144	5145	5154	5160	5161	5185	5217	5327	5361
	5421	5457	5550	5551	5615	5697	5759	5760	5761	5762	5763	5764	5766	5768	5828
TSTB	5285	5552	5560	5574	5577	5587	5590	5596	5604	5607	5610	5759	5761	5762	5764
	5765	5825													
.ASCII	4720	4859	4860	4861	4862										
.ASCII2	4720	4843	4844	4845	4846	4847	4848	4849	4850	4851	4852	4853	4854	4855	4856
	4857	4858	4864	4865	4866	4867	4868	4869	4870	4871	4872	4873	4874	4875	4876
	4896	5758	5761	5833	5834	5835	5836	5837							
.BLKB	5765														
.BLKW	5764														
.BYTE	4720	4887	5758	5763	5765	5782	5783	5801							
.ENABL	4	4701													
.END	5890														
.ENOC	4713	4715	4716	4717	4720	4893	4895	4896	4912	4935	4961	4997	5007	5038	5091
	5138	5178	5217	5225	5242	5264	5303	5320	5361	5372	5413	5457	5468	5510	5545
	5624	5644	5678	5758	5759	5760	5761	5762	5763	5764	5765	5766	5768		
.EQUIV	4716														
.EVEN	4877	4888	4896	5761	5838										
.IF	4713	4715	4716	4717	4720	4893	4895	4896	4912	4935	4961	4987	5007	5038	5091
	5138	5178	5217	5225	5242	5264	5303	5320	5361	5372	5413	5457	5468	5510	5545
	5624	5644	5678	5758	5759	5760	5761	5762	5763	5764	5765	5766	5768		
.IFF	4715	4716	4720	4895	4912	4935	4961	4987	5007	5038	5091	5138	5178	5217	5225
	5242	5264	5303	5320	5361	5372	5413	5457	5468	5510	5545	5624	5644	5678	5758
	5759	5760	5761	5762	5763	5764	5765	5766	5768						
.IFT	4896	5759	5760	5765	5766										
.IFTF	4896	5759	5760	5765	5766										
.IIF	4713	4715	4717	4720	4895	4896	4994	5050	5060	5112	5115	5217	5231	5249	5276
	5283	5292	5309	5361	5457	5758	5759	5760	5761	5762	5765	5768			
.IRP	4893	4935	4961	4987	5007	5038	5091	5138	5178	5195	5217	5225	5242	5264	5303
	5320	5337	5361	5372	5413	5431	5457	5468	5510	5545	5624	5678	5759	5760	5764
	5766	5773	5883	5886											
.LIST	2	4693	4697	4700	4716	4717	4720	4842	4891	4892	4893	4895	4896	4935	4961
	4987	5007	5038	5091	5138	5178	5217	5225	5242	5264	5303	5320	5361	5372	5413
	5457	5468	5510	5545	5624	5678	5758	5759	5760	5765	5768	5777	5809	5810	5832
	5840	5841	5875	5883											
.MACRO	39	81	166	301	478	509	580	737	783	797	825	918	944	975	1023

	1036	1079	1113	1146	1159	1180	1193	1276	1275	1321	1358	1405	1438	1449	1524
	1533	1584	1794	2003	2195	2283	2435	2703	2760	2825	2801	2973	3049	3143	3210
	3343	3405	3529	3567	3631	3736	3821	3923	3923	3961	4004	4102	4150	4420	4467
	4507	4583	4705	4709	4715	4720	4929	4929	4981	5000	5024	5093	5129	5171	5217
	5219	5236	5255	5297	5313	5361	5363	5406	5457	5459	5503	5537	5617	5670	5704
	5714	5743	5768												
.MCALL	4702	4703	4704	4716	4895										
.MLIST				4695	4696	4717	4720	4840	4841	4890	4893	4895	4896	4935	4961
	4987	5007	5038	5091	5138	5178	5217	5255	5242	5264	5303	5320	5361	5372	5413
	5457	5468	5510	5545	5624	5678	5758	5759	5760	5765	5768	5771	5772	5807	5830
	5831	5839	5873	5883											
.PAGE	4720	4816	4899	5767	5769	5811									
.REPT	4717	4720	5876												
.SBTTL	4715	4716	4717	4720	4935	4961	4987	5007	5038	5091	5138	5178	5217	5225	5242
	5264	5303	5320	5361	5372	5413	5457	5468	5510	5545	5624	5678	5758	5759	5760
	5761	5762	5763	5764	5765	5766	5768								
.TITLE	4713														
.WORD	4717	4720	4879	4880	4881	5758	5761	5762	5763	5766	5775	5778	5779	5780	5781
	5784	5785	5786	5787	5788	5789	5790	5791	5792	5793	5794	5795	5796	5797	5798
	5799	5802	5803	5804	5805										

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

* DWQAA/CRF=SYSMAC.B1 DWQAA.P11
RUN-TIME: 38 46 4 SECONDS
RUN-TIME RATIO: 326/89=3.6
CORE USED: 29K (57 PAGES)

C06

Special printing in 10.000000 00 000 000 0000. 2 digit 00100, 15 page

