



The main body of the document is a large, dense grid of data. It consists of approximately 15 columns and 25 rows of small, illegible text or symbols. The content is too faint to be transcribed accurately, but it appears to be a structured table or a series of data points organized in a regular pattern.



.SBTTL USER DOCUMENTATION

.REM 8

IDENTIFICATION

PRODUCT CODE: AC-S918A-MC
PRODUCT NAME: CVMEMAO NON-VOLATILE DATA RETENTION DIAGNOSTIC
PRODUCT DATE: 5-AUGUST-81
MAINTAINER: STORAGE SYSTEMS DIAGNOSTICS
AUTHOR: KENNETH LANGLAIS

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.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1981 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

TABLE OF CONTENTS

- 1.0 OVERVIEW OF DIAGNOSTIC PRODUCT
 - 1.1 PRODUCT DESCRIPTION
 - 1.2 PRODUCT USERS
 - 1.2.1 ENGINEERING USAGE
 - 1.2.2 MANUFACTURING USAGE
 - 1.2.3 FIELD SERVICE USAGE
 - 1.2.4 CUSTOMER USAGE
- 2.0 PRODUCT GOALS
 - 2.1 ASSUMPTIONS
 - 2.2 PERFORMANCE GOALS
 - 2.3 COMPATIBILITY GOALS
 - 2.4 FAILSOFT GOALS
 - 2.5 RESTRICTIONS
 - 2.6 NON-GOALS
- 3.0 REQUIREMENTS
 - 3.1 RUN TIME ENVIRONMENT REQUIREMENTS
 - 3.2 DEVELOPMENT ENVIRONMENT REQUIREMENTS
- 4.0 FUNCTIONAL DESCRIPTION
 - 4.1 MENU QUESTIONS
 - 4.2 POWER FAIL
 - 4.3 POWER UP
 - 4.4 ERROR WITHIN MEMORY MAP
 - 4.5 PROGRAM FLOW FOR AN UNMAPPED SYSTEM
 - 4.5.1
 - 4.5.2
 - 4.5.3
 - 4.6 PROGRAM FLOW FOR A MAPPED SYSTEM
 - 4.6.1
 - 4.6.2

- 4.7 HELP FILE
- 4.8 ERROR INFORMATION
- 4.9 SCHEDULE

- 5.0 INTERFACES

- 6.0 BIBLIOGRAPHY

- 7.0 GLOSSARY

1.0 OVERVIEW OF DIAGNOSTIC PRODUCT

1.1 PRODUCT DESCRIPTION

MEMORY DIAGNOSTICS WAS CONTRACTED BY TOEM MARKETING GROUP TO GIVE DIGITAL CUSTOMERS, FINAL ASSEMBLY AND TEST (F.A. AND T.), AND FIELD SERVICE A MEANS OF VERIFYING THAT THE NON-VOLATILE MEMORY AND IT'S BATTERIES ARE OPERATIONAL. THIS DIAGNOSTIC WILL TEST NON-VOLATILE MEMORY IN ANY QBUS SYSTEM PROVIDING THE RESTRICTION CRITERIA SET FORTH IN SECTION 2 OF THIS MANUAL IS ADHERED TO.

1.2 PRODUCT USERS

1.2.1 ENGINEERING USAGE -

- DESIGN VERIFICATION OF BATTERY BACKUP CIRCUITRY
- AVAILABLE FOR DMT USE

1.2.2 MANUFACTURING USAGE -

VOLUME MANUFACTURING
A MEANS OF VERIFYING DATA DETECION OF NON-VOLATILE MEMORIES.

F A AND T
THIS DIAGNOSTIC WILL GIVE F.A. AND T. A MEANS OF CHECKING THE NON-VOLATILITY OF CMOS RAMS AND THE ABILITY OF THE BATTERIES TO MAINTAIN DATA RETENTION.

1.2.3 FIELD SERVICE USAGE -

VERIFICATION OF CUSTOMER INSTALLATION
SERVICE CALLS: FAILURE ISOLATION, REPAIR AND VERIFICATION
VERIFICATION OF ECO INSTALLATION

1.2.4 CUSTOMER USAGE -

MEANS OF VERIFYING DATA RETENTION OF NON-VOLATILE MEMORY AND IT'S BATTERIES.

2.0 PRODUCT GOALS

2.1 ASSUMPTIONS

IT WILL BE ASSUMED THAT PRIOR TO EXECUTING THIS DIAGNOSTIC ALL APPROPRIATE CPU, MEMORY, AND PERIPHERAL DIAGNOSTICS HAVE BEEN SUCCESSFULLY RUN.

THE DIAGNOSTIC WILL REPORT ANY BAD LOCATIONS FOUND WITHIN ANY AREA OF MEMORY THAT NON-VOLATILE EXISTS.

THE OPERATOR MUST HAVE THE KNOWLEDGE OF THE STARTING ADDRESS OF A NON-VOLATILE MEMORY WITHIN THE OPERATOR'S SYSTEM.

THIS DIAGNOSTIC TREATS ALL MEMORY MODULES CONTAINING ROM, NON-BATTERY BACKED UP MOS RAM, AND NON-BATTERY BACKED UP CMOS RAM AS VOLATILE MEMORY. ALL MEMORY MODULES CONTAINING CORE, BATTERY BACKED UP MOS RAM, AND BATTERY BACKED UP CMOS RAM AS NON-VOLATILE MEMORY.

THE Q-BUS CPU MUST HAVE EITHER A HALT/ENABLE SWITCH OR THE CPU MODULE MUST BE STRAPPED TO HALT WHEN POWERED UP (WIRE JUMPER CONFIGURATION MUST BE W5-IN, W6-OUT)

THE OPERATOR OF THIS DIAGNOSTIC IS ASSUMED TO HAVE THE MEANS OF VERIFYING THAT THE MEMORY MAP THIS DIAGNOSTIC WILL PRODUCE CORRECTLY CONFIGURES THE SYSTEM UNDER TEST.

2.2 PERFORMANCE GOALS

THE GOAL OF THIS DIAGNOSTIC IS TO DETERMINE THE DATA RETENTION OF ALL NON-VOLATILE MEMORY CONTAINED WITHIN A GIVEN SYSTEM. THIS PROGRAM WILL REQUIRE INFORMATION FROM THE OPERATOR DURING IT'S EXECUTION. THE INFORMATION WILL BE GATHERED USING A MENU TECHNIQUE.

2.3 COMPATIBILITY GOALS

THE INITIAL DIAGNOSTIC IS EXPECTED TO RUN ON ANY QBUS SYSTEM PROVIDING THAT THE TRAP LOCATIONS HAVE READ/WRITE ACCESS.

2.4 FAILSOFT GOALS

ANY PARITY ERROR CAUSED BY ACCESSING VOLATILE MEMORY AFTER POWER DOWN WILL BE IGNORED.

2.5 RESTRICTIONS

THIS DIAGNOSTIC IS DESIGNED TO RUN ON ALL QBUS SYSTEM EXCEPT FOR THOSE SYSTEMS THAT DO NOT ALLOW READ/WRITE ACCESS IN THE TRAP VECTOR SECTION OF MEMORY (ADDRESS 0 THROUGH 400 OCTAL). THE SYSTEM MUST HAVE A TERMINAL IN ORDER FOR THE OPERATOR TO ANSWER THE QUESTIONS THE DIAGNOSTIC WILL ASK. THE OPERATOR MUST KNOW AT WHICH LOCATIONS IN MEMORY THE NON-VOLATILE MEMORY EXISTS.

THIS DIAGNOSTIC IS TO ONLY WORK WITH CPU'S THAT SUPPPORT POWER-DOWN DATA RETENTION CAPABILITY.

2.6 NON-GOALS

THIS DIAGNOSTIC'S ONLY PURPOSE IS TO CHECK THAT NON-VOLATILE IS WORKING AND THAT THE BATTERIES ARE OK. THIS DIAGNOSTIC PERFORMS NO OTHER TESTING.

- NOT APT COMPATIBLE
- NOT IN THE SCOPE OF THIS PROJECT TO MODIFY THIS PROGRAM TO MEET UNIQUE NEEDS OF MANUFACTURING.

3.0 REQUIREMENTS

3.1 RUN TIME ENVIRONMENT REQUIREMENTS

QBUS CPU
CONSOLE TERMINAL
AT LEAST 1 OF ANY OF THE FOLLOWING MEMORY MODULES:
 MCV11 CMOS RAM MEMORY MODULES WITH BATTERY BACKUP
 MSV11 MOS RAM MEMORY WITH BATTERY BACKUP
 MMV11 CORE MEMORY MODULES
XXDP+ LOADING MEDIA

3.2 DEVELOPMENT ENVIRONMENT REQUIREMENTS

FOR PROGRAM DEVELOPMENT DIAGNOSTIC ENGINEERING WILL BE ABLE TO USE OUR OWN EQUIPMENT, BUT FOR INTIAL DEBUG, IT IS ESSENTIAL THAT WE HAVE ACCESS TO A PROPERLY EQUIPPED RUN TIME SYSTEM AS DESCRIBED IN THE PREVIOUS SECTION.

THIS DIAGNOSTIC WILL BE VERIFIED ON A VARIETY OF Q-BUS SYSTEMS.

4.0 FUNCTIONAL DESCRIPTION

THE PURPOSE OF THIS PROGRAM IS TO TEST THE DATA RETENTION OF NON-VOLATILE MODULES IN ANY QBUS SYSTEM. THIS WILL BE ACCOMPLISHED BY ASKING THE OPERATOR A SERIES OF QUESTIONS (MENU), GENERATING A RESTART HELP FILE, WRITE A BACKGROUND PATTERN (125252) THROUGHOUT MEMORY, DO A CHECKSUM OF THE PROGRAM AND THE TRAP VECTOR SPACE (10-376), TELLING THE OPERATOR TO POWERDOWN THE SYSTEM, POWER IT BACK UP, RESTART THE DIAGNOSTIC, VERIFY THE CHECKSUMS, CHECK ENTIRE MEMORY FOR THE BACKGROUND PATTERN (125252), AND GIVE THE OPERATOR A MEMORY MAP OF ALL VOLATILE AND NON VOLATILE MEMORY IN THE SYSTEM. THE OPERATOR MUST COMPARE THIS MAP WITH THE ONE LEFT BY THE INSTALLER IN ORDER TO DETERMINE THAT PROPER DATA RETENTION EXISTS.

4.1 MENU QUESTIONS

THE DIAGNOSTIC WILL PROMPT THE OPERATOR TO ANSWER THE FOLLOWING QUESTIONS A <CR> AS AN ANSWER TO THE FOLLOWING QUESTIONS WILL CAUSE THE CONDITION STATED DIRECTLY TO THE LEFT OF THE QUESTION MARK TO BE IN EFFECT.

1. DO YOU WANT A HELP FILE (L) N?

A 'Y' ANSWER TO THIS WILL CAUSE GENERAL HELP FILE TO BE PRINTED BEFORE CONTINUING WITH THE QUESTIONS.

2. IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y?

A 'Y' OR <CR> ANSWER WILL INHIBIT THE ASKING OF QUESTION 3.

3. WHAT IS THE STARTING ADDRESS OF ONE OF THE NON-VOLATILE MEMORY MODULES (O) O?

THE PROGRAM WILL NOW MOVE ITSELF TO THAT AREA IN MEMORY. IF THE ADDRESS IS NOT A LEGAL 4K BOUNDARY ADDRESS THE DIAGNOSTIC WILL PRINT THE FOLLOWING ERROR MESSAGE AND REPEAT QUESTION 3.

ILLEGAL ADDRESS - NOT A 4K BOUNDARY

IF THE ADDRESS IS OUTSIDE THE MEMORY AREA OF THE SYSTEM THE FOLLOWING ERROR MESSAGE WILL BE PRINTED AND QUESTION 3 IS REPEATED.

4K ADDRESS OR NUMBER OF 4K BANKS ARE OUT OF ADDRESS RANGE OF THIS SYSTEM.

NOTE

* IF THE ADDRESS TYPED IN IS REALLY VOLATILE THEN WHEN THE PROGRAM IS RESTARTED THE SYSTEM WILL NOT OPERATE PROPERLY.

4. DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM (L) Y?

A 'Y' <CR> ANSWER WILL CAUSE NO MORE QUESTIONS TO BE PROMPT AND THE PROGRAM FLOW WILL GO TO THE SECTION ON POWERFAIL 4.2

5. WHAT IS THE STARTING ADDRESS OF THE NON-VOLATILE MEMORY UNDER TEST (O) 0?

THIS ADDRESS MAY OR MAY NOT BE THE SAME AS THE ADDRESS TYPED IN QUESTION 3. IF IT IS THEN THE PROGRAM WILL RESIDE WITHIN THE QUESTIONED NON-VOLATILE MEMORY MODULE. IF NOT THEN THE PROGRAM WILL BE AT EITHER ADDRESS 0 IF QUESTION 2 WAS A 'Y' OR AT THE ADDRESS QUESTION 3 ASKED FOR.

6. HOW MANY 4K BANKS OF MEMORY TO BE TESTED (D) 0?

THE OPERATOR WILL INPUT THE NUMBER OF 4K BANKS TO BE TESTED FROM THE START ADDRESS, A '0' OR <CR> ANSWER WILL MEAN TEST ALL THE BANKS FROM THE START ADDRESS TO END OF MEMORY.

4.2 POWER FAIL

BEFORE THIS DIAGNOSTIC INSTRUCTS THE OPERATOR TO POWER DOWN THE SYSTEM. THE DIAGNOSTIC WILL PRODUCE A RESTART HELP FILE ON THE CONSOLE TERMINAL. THIS HELP FILE IS NEEDED TO GUIDE THE OPERATOR IN A STEP BY STEP MENU THAT MUST BE PERFORMED IN ORDER TO RESTART THE DIAGNOSTIC AFTER POWER IS RETURNED. SECTION 4.5 GIVES AS PART OF THE EXAMPLES TYPICAL RESTART HELP FILES FOR VARIOUS TYPES OF SYSTEMS.

THE OPERATOR WILL THEN BE INSTRUCTED TO POWER DOWN THE SYSTEM FOR NO LESS THAN 2 MINUTES AND NO MORE THAN 100 HOURS. THE ACTUAL MESSAGE IS:

PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO LONGER THAN 100 HOURS, EXECUTE THE RESTART HELP FILE.

4.3 POWER UP

AFTER THE POWER WAS RETURNED TO THE SYSTEM THE OPERATOR MUST FOLLOW THE STEP BY STEP PROCEDURE ON HOW TO RESTART THE DIAGNOSTIC GIVEN BY THE PROGRAM BEFORE POWER DOWN. THE DIAGNOSTIC WILL START CHECKING ALL MEMORY FOR DATA RETENTION. ON THE CONSOLE TERMINAL A MEMORY MAP WILL BE PRINTED AT ALL AREAS WITHIN THE SYSTEM THAT CONTAIN VOLATILE AND NON-VOLATILE MEMORY.

AN EXAMPLE OF A MEMORY MAP GIVEN BY THIS DIAGNOSTIC:

MEMORY MAP -----

START ADR.	END ADR.	MEMORY TYPE	WORD ERROR	PARITY ERROR
0	37776	NON-VOLATILE MEMORY	0	0
40000	157776	VOLATILE MEMORY		

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST

4.4 ERROR WITHIN MEMORY MAP

- A) WORD ERROR - THE NUMBER OF WORD FAILURES FOUND WITHIN A GIVEN 4K CHUNK OF MEMORY IF THE DIAGNOSTIC FEELS THAT 4K IS NON-VOLATILE.
- B) PARITY ERROR IS THE NUMBER OF PARITY ERRORS FOUND WITHIN A 4K PART OF MEMORY. THIS WILL ONLY PRINT OUT ERRORS FOUND IN NON-VOLATILE MEMORY. DUE TO THE EXCESSIVE RUNTIME OF THE DIAGNOSTIC WHEN SERVICING EVERY PARITY ERROR (WHICH INCLUDES ALL PARITY ERRORS WITHIN VOLATILE MEMORY) A DECISION WAS MADE TO LIMIT THE PARITY SERVICING CAPABILITY TO 10 PARITY ERRORS PER 4K OF MEMORY. IF THE MEMORY THE PARITY ERRORS WERE IN WERE NON-VOLATILE THEN A MESSAGE OF "10 OR MORE" WOULD BE PRINTED UNDER PARITY ERRORS.
- C) PROGRAM CHECK SUM ERROR - BEFORE THE MEMORY MAP HEADER IS PRINTED THE PROGRAM DOES A CHECKSUM OF ITSELF TO MAKE SURE THAT THE MEMORY THE PROGRAM IS IN IS OK. IF NOT THEN THE FOLLOWING ERROR MESSAGE WILL NOTIFY THE OPERATOR.

CHECKSUM ERROR. THE MEMORY THE PROGRAM IS RESIDENT IN HAS VOLATILE LOCATIONS.

THE PROGRAM WILL CONTINUE BUT MAY NOT OPERATE PROPERLY.
- D) TRAP VECTOR CHECKSUM. ANOTHER CHECKSUM WAS ALSO DONE WITH THE VECTOR PAGE (ADDRESS 0-376) OF BANK 0. THIS TIME THE ERROR WILL ONLY BE REPORTED IF BANK 0 WAS DETERMINED TO BE NON-VOLATILE.

ERROR MESSAGE:

TRAP LOCATIONS (ADDRESS 0-376) HAVE CHECK SUM ERROR IN
NON-VOLATILE MEMORY.

4.5 PROGRAM FLOW FOR AN UNMAPPED SYSTEM

4.5.1 -

THE FOLLOWING IS AN EXAMPLE OF AN UN-MAPPED SYSTEM WITH NON-VOLATILE
MEMORY IN THE LOWEST PART OF MAIN MEMORY. AFTER LOADING THE DIAGNOS-
TIC, THE FOLLOWING SEQUENCY OF EVENTS WILL APPEAR ON THE CONSOLE TER-
MINAL.

CVMEMAO DATA RETENTION DIAGNOSTIC FOR NON-VOLATILE MEMORIES.

THIS IS AN UN-MAPPED SYSTEM (NO MEMORY MANAGEMENT) WITH 28K OF
MEMORY.

DO YOU WANT A HELP FILE (L) N? N

IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y? Y

DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM
(L) Y? Y

RESTART HELP FILE

SET ENABLE/HALT FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER,
SET ENABLE/HALT FRONT PANEL SWTICH TO ENABLE POSITION, AND TYPE:

@777707/506<CR>

@P

PLEASE STAND BY!

PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO LONGER
THAN 100 HOURS. EXECUTE THE RESTART HELP FILE.

THE OPERATOR WILL NOW POWER DOWN, WAIT 2 MINUTES AND RETURN POWER TO
THE SYSTEM. THE FOLLOWING IS A TYPICAL MEMORY MAP FOR AN UN-MAPPED
SYSTEM.

MEMORY MAP

START ADR.	END ADR.	MEMORY TYPE	WORD ERROR	PARITY ERROR
0	37776	NON-VOLATILE MEMORY	0	0
40000	137776	VOLATILE MEMORY		
140000	157776	NON-VOLATILE MEMORY	0	0

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST.

4.5.2 -

THE FOLLOWING IS AN EXAMPLE OF AN UN-MAP
PED SYSTEM WITH NON-VOLATILE
MEMORY SOMEWHERE OTHER THAN LOWEST PART OF MAIN MEMORY.

AFTER LOADING THE DIAGNOSTIC, THE FOLLOWING SEQUENCE OF EVENTS WILL
APPEAR ON THE CONSOLE TERMINAL.

CVMEMAO DATA RETENTION DIAGNOSTIC FOR NON-VOLATILE MEMORIES

THIS IS AN UN-MAPPED SYSTEM (NO MEMORY MANAGEMENT) WITH 24K OF
MEMORY.

DO YOU WANT A HELP FILE (L) N? N

IS THERE NON VOLATILE MEMORY AT ADDRESS 0 (L) Y? N

WHAT IS THE STARTING ADDRESS OF ONE OF THE NON-VOLATILE MEMORY
MODULES L (0) 0? 20000

DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM
(L) Y? Y

RESTART HELP FILE

SET ENABLE/HALT FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER,
SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE POSITION, AND TYPE:

@\$7/XXXXXX 20176<CR>
@P

PLEASE STAND BY!
PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO LONGER
THAN 100 HR. EXECUTE THE RESTART HELP FILE.

THE OPERATOR WILL NOW POWERDOWN, WAIT 2 MINUTES, AND RETURN POWER TO
THE SYSTEM. NEXT THE OPERATOR MUST FOLLOW THE RESTART HELP FILE.
AFTER COMPLETION OF THE HELP FILE THE DIAGNOSTIC WILL PRINT A TYPICAL
MEMORY MAP FOR AN UN-MAPPED SYSTEM.

MEMORY MAP

START ADR.	END ADR.	MEMORY TYPE	WORD ERROR	PARITY ERROR
0	17776	VOLATILE MEMORY		
20000	77776	NON-VOLATILE MEMORY	0	0
100000	177776	VOLATILE MEMORY		

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST.

4.5.3 -

THE FOLLOWING IS AN EXAMPLE OF EXECUTING THE DIAGNOSTIC IN AN UN-MAPPED SYSTEM THAT CONTAIN 2 NON-VOLATILE ARRAYS. THE OPERATOR WANTS TO EXECUTE THE TEST ON THE NON-VOLATILE ARRAY THAT RESIDES IN THE LOWEST PART OF MAIN MEMORY AND HAVE THE PROGRAM RESIDE IN THE OTHER NON-VOLATILE ARRAY. EACH ARRAY HAS 4K OF MEMORY.

CVMEMAO DATA RETENTION DIAGNOSTIC FOR NON-VOLATILE MEMORIES

THIS IS AN UN-MAPPED SYSTEM (NO MEMORY MANAGEMENT) WITH 28K OF MEMORY.

DO YOU WANT A HELP FILE (L) N? N

IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y? N

WHAT IS THE STARTING ADDRESS OF ONE OF THE NON-VOLATILE MEMORY MODULES (O) 0? 20000

DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM (L) Y? N

WHAT IS THE STARTING ADDRESS OF THE NON-VOLATILE MEMORY UNDER TEST (O) 0? 0

HOW MANY 4K BANKS OF MEMORY ARE TO BE TESTED (D) 0? 1

RESTART HELP FILE

SET ENABLE/HALT FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER, SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE POSITION, AND TYPE:

@S7/XXXXXX 20204<CR>
@P

PLEASE STAND BY!
PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO LONGER THAN 100 HR. EXECUTE THE RESTART HELP FILE.

THE OPERATOR WILL NOW POWER DOWN, WAIT 2 MINUTES, AND RETURN POWER TO THE SYSTEM. NEXT THE OPERATOR MUST FOLLOW THE RESTART HELP FILE. AFTER COMPLETION OF THE HELP FILE THE DIAGNOSTIC WILL PRINT A MEMORY MAP FOR THAT PART OF MEMORY ONLY.

MEMORY MAP

START ADR.	END ADR.	MEMORY TYPE	WORD ERROR	PARITY ERROR
0	1776	NON-VOLATILE MEMORY	0	0

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST.

4.6 PROGRAM FLOW FOR A MAPPED SYSTEM

4.6.1 -

THE FOLLOWING IS AN EXAMPLE OF A MAPPED SYSTEM WITH NON-VOLATILE MEMORY IN THE LOWEST PART OF MAIN MEMORY.

AFTER LOADING THE DIAGNOSTIC, THE FOLLOWING SEQUENCE OF EVENTS WILL APPEAR ON THE CONSOLE TERMINAL.

CVMEMAO DATA RETENTION DIAGNOSTIC FOR NON-VOLATILE MEMORIES.

THIS IS A MAPPED SYSTEM (MEMORY MANAGEMENT) WITH 124K OF MEMORY.

DO YOU WANT A HELP FILE (L) N? N

IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y? Y

DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM (L) Y? Y

RESTART HELP FILE

SET ENABLE/HALT FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER, TYPE:

```
@772344/XXXXXX 0<CR>  
@772356/XXXXXX 177600<CR>  
@772304/XXXXXX 77406<CR>  
@772316/XXXXXX 77406<CR>  
@777572/XXXXXX 1<CR>
```

SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE POSITION AND TYPE:

```
@S7/XXXXXX 40510<CR>  
@P
```

PLEASE STAND BY!
PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO LONGER THAN 100 HOURS. EXECUTE THE RESTART HELP FILE.

THE OPERATOR WILL NOW POWER DOWN, WAIT 2 MINUTES AND RETURN POWER TO THE SYSTEM. THE FOLLOWING IS A TYPICAL MEMORY MAP FOR A MAPPED SYSTEM.

MEMORY MAP

START ADR.	END ADR.	MEMORY TYPE	WORD ERROR	PARITY ERROR
0	37776	NON-VOLATILE MEMORY	0	0
40000	757776	VOLATILE MEMORY		

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST.

4.6.2 -

THE FOLLOWING IS AN EXAMPLE OF A MAPPED SYSTEM WITH NON-VOLATILE MEMORY SOMEWHERE OTHER THAN LOWEST PART OF MAIN MEMORY.

AFTER LOADING THE DIAGNOSTIC, THE FOLLOWING SEQUENCE OF EVENTS WILL APPEAR IN THE CONSOLE TERMINAL.

CVMEAO DATA RETENTION DIAGNOSTIC FOR NON-VOLATILE MEMORIES

THIS IS A MAPPED SYSTEM (MEMORY MANAGEMENT) WITH 2043K OF MEMORY.

DO YOU WANT A HELP FILE (L) N? N

IS THERE NON-VOLATILE AT ADDRESS 0 (L) Y? N

WHAT IS THE STARTING ADDRESS OF THE NON-VOLATILE MEMORY MODULES
(O) 0? 1000000

DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY IN THIS SYSTEM
(L) Y? Y

RESTART HELP FILE

SET ENABLE/HALT FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER,
TYPE:

```
@772344/XXXXXX 10000<CR>
@772356/XXXXXX 177600<CR>
@772304/XXXXXX 77406<CR>
@772316/XXXXXX 77406<CR>
@777572/XXXXXX 1<CR>
@772516/XXXXXX 20<CR>
```

SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE POSITION AND TYPE:

```
@S7/XXXXXX 41034<CR>
@P
```

PLEASE STAND BY!

PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT NO MORE THAN 100
HR. EXECUTE THE RESTART HELP FILE.

THE OPERATOR WILL NOW POWER DOWN, WAIT 2 MINUTES, AND RETURN POWER TO
THE SYSTEM. NEXT THE OPERATOR MUST FOLLOW THE RESTART HELP FILE.
AFTER COMPLETION OF THE HELP FILE THE DIAGNOSTIC WILL PRINT A TYPICAL
MEMORY MAP FOR A MAPPED SYSTEM.

MEMORY MAP

START ADR.	END ADR.	MEMORY TYPE	WORD ERRGR	PARITY ERROR
0	3777776	VOLATILE MEMORY	0	
4000000	4077776	NON VOLATILE MEMORY	0	0
4100000	7777776	VOLATILE MEMORY	0	
1000000	1017776	NON-VOLATILE MEMORY	0	0
1020000	17757776	VOLATILE MEMORY	0	

THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST.

4.7 HELP FILE

FUNCTIONAL DESCRIPTION

THE PURPOSE OF THIS PROGRAM IS TO TEST THE DATA RETENTION OF NON-VOLATILE MODULES IN ANY QBUS SYSTEM. THIS IS NOT A MEMORY DIAGNOSTIC. PLEASE RUN CZKMA OR VMSA DIAGNOSTICS BEFORE RUNNING THIS PROGRAM. THE FOLLOWING IS A BRIEF DISCRIPTION OF THE PROGRAM FLOW:

1. ASK THE OPERATOR A SERIES OF QUESTIONS (MENU)
2. RELOCATE PROGRAM TO NON-VOLATILE AREA (IF BANK 0 IS VOLATILE)
3. GENERATE AND PRINT A RESTART HELP FILE
4. WRITE A BACKGROUND PATTERN (125252) THROUGHOUT MEMORY
5. DO A CHECKSUM OF THE PROGRAM
6. DO A CHECKSUM OF THE TRAP VECTOR SPACE (ADDRESS 0-376)
7. TELL THE OPERATOR TO POWERDOWN THE SYSTEM

THE OPERATOR WILL NOW FOLLOW THE RESTART HELP FILE SOME TIME BETWEEN 2 MINUTES AND 100 HOURS. THE FOLLOWING IS THE FLOW OF THE PROGRAM AFTER POWER UP:

1. DO ANOTHER CHECKSUM OF THE PROGRAM AND CHECK IT AGAINST THE ONE DONE BEFORE POWER DOWN.
2. DO ANOTHER CHECKSUM OF THE TRAP VECTOR SPACE AND CHECK IT AGAINST THE ONE DONE BEFORE POWER DOWN. (NO TRAP CHECKSUM ERROR WILL BE REPORTED UNLESS BANK 0 WAS DETERMINED NON-VOLATILE)
3. CHECK ENTIRE MEMORY FOR THE BACKGROUND PATTERN (125252)
4. PRINT A MEMORY MAP OF ALL VOLATILE AND NON VOLATILE MEMORY IN THE SYSTEM

THE OPERATOR MUST COMPARE THIS MAP WITH THE ONE LEFT BY THE INSTALLER IN ORDER TO DETERMINE THAT PROPER DATA RETENTION EXISTS.

PREREQUISITE

1. CZKMA OR VMSA MUST SUCCESSFULLY COMPLETE
2. OPERATOR MUST KNOW STARTING ADDRESSES OF ALL VOLATILE AND NON-VOLATILE MEMORY IN THE SYSTEM
3. VIDEO TERMINALS MUST EITHER REMAIN POWERED WHEN PROCESSOR IS POWERED DOWN OR A COPY OF THE RESTART HELP FILE MUST BE MADE BEFORE POWERING DOWN THE SYSTEM
4. IF THE CPU DOES NOT HAVE AN ENABLE/HALT SWITCH, CHECK THAT THE CPU IS STRAPPED TO HALT WHEN POWERED UP. OTHERWISE THE DIAGNOSTIC WILL NOT OPERATE PROPERLY.

4.8 ERROR INFORMATION

IN THE EVENT THAT THE DIAGNOSTIC DID NOT CONTINUE WHEN RESTART HELP FILE WAS COMPLETED, THE FOLLOWING EXAMPLES ARE OPERATOR ERROR THAT COULD CAUSE THE DIAGNOSTIC NOT TO RESTART.

1. STARTING ADDRESS WAS VOLATILE MEMORY
2. ERROR WHEN SETTING UP RESTART REGISTERS WHEN USING CONSOLE ODT.
3. ENABLE/HALT FRONT PANEL SWITCH SET TO ENABLE WHEN POWER IS RETURNED.
4. CPU WITHOUT ENABLE/HALT FRONT PANEL WAS NOT STRAPPED TO HALT WHEN POWERED UP (W5-IN, W6-OUT)

IF NO OPERATOR ERROR WAS DETERMINED THEN THE DATA RETENTION OF THE NON-VOLATILE MEMORY THE PROGRAM WAS IN IS NOW BAD. CHECK THE BATTERIES.

4.9 SCHEDULE

THE PROGRAM IS EXPECTED TO TAKE 8 WEEKS FROM DESIGN THROUGH DEBUG AT A COST OF 11K.

THE COST AND TIME WILL FLUCTUATE IF THERE ARE ANY CHANGES TO THE FUNCTIONALITY OF THIS PROGRAM.

5.0 INTERFACE

THE DIAGNOSTIC WILL BE DESIGNED TO CONFORM WITH THE FOLLOWING INTERFACES.

1. SYSMAC
2. XXDP+

6.0 BIBLIOGRAPHY

THE FOLLOWING LISTS THE LITERATURE USED IN THE DESIGN OF THIS DIAGNOSTIC.

1. SYSMAC.MAN
2. SPMACJ.DOC
3. XXDPPLUS.DOC

7.0 GLOSSARY

&

```

000000      .ENABLE ABS
              .DSABLE GBL

889          .LIST SEQ,9IN,CND
890
891
892          .MACRO .SSIZE MMERR
893          .SBTTL ROUTINE TO SIZE MEMORY
894          STARS
895          : *CALL:
896          : *      JSR      PC,$SIZE
897          : *      RETURN
898          .IF DF KIPARO
899          : *$LSTAD WILL CONTAIN:
900          : *      WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
901          : *      WITHOUT KT11 OPTION   -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
902          : *$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
903          : *$KT11 IS THE MEMORY MANAGEMENT KEY
904          : *BIT07 = 0 DON'T USE MEMORY MANAGEMENT
905          : *      MUST BE SETUP BEFORE THE CALL
906          : *BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
907          : *      DETERMINED BY ROUTINE
908          .IFF
909          : *$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
910          .IFTF
911          $SIZE:  MOV      R0,-(SP)      ;;SAVE R0 ON THE STACK
912                  MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
913          .IFT
914                  MOV      R2,-(SP)      ;;SAVE R2 ON THE STACK
915                  MOV      R3,-(SP)      ;;SAVE R3 ON THE STACK
916                  MOV      R4,-(SP)      ;;SAVE R4 ON THE STACK
917          .IFTF
918                  MOV      @#114,-(SP)   ;;SAVE MEMORY ERROR VECTOR PS & PC
919                  MOV      @#116,-(SP)
920                  MOV      #116,@#114    ;;IGNORE PARITY ERRORS WHILE SIZING
921                  MOV      #RTI,@#116
922                  MOV      @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
923                  MOV      @#ERRVEC+2,-(SP)
924                  MOV      SP,R0        ;;SAVE THE STACK POINTER
925          ;;SET THE ERRVEC PS TO THE PRESENT PS
926                  GETPRI @#ERRVEC+2
927          .IFT
928                  MOV      #3776,R1      ;;SETUP ADDRESS
929                  TSTB     (PC)+        ;;USE MEMORY MANAGEMENT?
930          $KT11:  .WORD     200          ;;SET TO USE MEMORY MANAGEMENT
931                  BPL      $CORE       ;;BR IF NO
932                  MOV      # $KTNEX,@#ERRVEC ;;SET FOR TIMEOUT
933                  TST      @#SRO       ;;KT11 ARE YOU THERE?
934                  BIS      #100000,$KT11 ;;YES--SET KT11 KEY
935                  MOV      #100$,@#ERRVEC ;;SET FOR TIMEOUT
936                  TST      @#172516    ;;Q-BUS MAP ARE YOU THERE?
937                  MOV      #200,$MAP    ;;TURN ON MAP INDICATOR
938                  MOV      #176200,$STOP ;;END OF 2M OF MEMORY
939                  BR       $MAPRG       ;;GO SET UP MAP REGISTERS
940          100$:  MOV      #6200,@#$STOP  ;;COMPARISON VALUE FOR 18 BIT MAPPING
941                  CMP      (SP)+,(SP)+  ;;CLEAN OFF STACK
942                  CLR      @#$MAP      ;;MAKE SURE MAP INDICATOR TURNED OFF
943                  BR       $NOMAP      ;;
  
```

BK001

BK001
 BK001
 BK001
 BK001
 BK001
 BK001
 BK001
 BK001


```

944 $MAP: .WORD 0 ;;=200 IF MAP PRESENT BK001
945 $STOP: .WORD 0 ;;FILLED WITH APPROPRIATE COMPARISON VALUE BK001
946 $MAPRG:
947 100$:
948 $NOMAP:
949 .IF NB MMERR
950 MOV #MMERR,@MMVEC ;;SET IN CASE OF ERROR
951 MOV #340,@MMVEC+2
952 .ENDC
953 CLR -(SP) ;;INITIALIZE FOR 'PAR' LOADING
954 MOV #KIPAR0,R2 ;;ADDRESS OF FIRST 'PAR'
955 MOV #^D8,R3 ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
956 1$: MOV #77406,-40(R2) ;;PDR = 4K, UP, READ/WRITE
957 MOV (SP),(R2)+ ;;LOAD 'PAR'
958 ADD #200,(SP) ;;UPDATE FOR NEXT 'PAR'
959 SOB R3,1$ ;;LOOP UNTIL ALL EIGHT ARE LOADED
960 MOV #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
961 CLR -(R2) ;;SETUP KIPAR6 FOR TESTING
962 MOV #2$,@ERRVEC ;;CATCH TIMEOUT IF NO SR3
963 MOV #20,@SR3 ;;ENABLE 22 BIT MODE AND UNIBUS MAP BK001
964 BR 3$ ;;THIS PDP-11 HAS A SR3 REGISTER
965 2$: CMP (SP)+,(SP)+ ;;CLEAN OFF THE STACK--NO SR3
966 3$: INC @SR0 ;;TURN ON MEMORY MANAGEMENT
967 MOV #SKTOUT,@ERRVEC ;;SET FOR TIME OUT
968 TSTB @SMAP ;;IS MAP THERE? BK001
969 BPL 4$ ;;NO-SKIP BK001
970 MOV #SMMOUT,@#114 ;;SET UP MEMORY ERROR VECTOR BK001
971 MOV @ERRVEC+2,@#116 ;;LOCK OUT INTERRUPTS BK001
972 4$: TST @#143776 ;;TRAP ON NON-EX-MEM
973 ADD #40,(R2) ;;MAKE A 1K STEP
974 CMP @SSTOP,(R2) ;;LAST ONE?
975 BHI 4$ ;;NO--TRY IT
976 SKTOUT: MOV (R2),R2 ;;GET LAST BANK+1
977 CLR @SR0 ;;TURN OFF MEMORY MANAGEMENT
978 TSTB @SMAP ;;IS MAP THERE? BK001
979 BPL $SIZEX ;;NO-SKIP BK001
980 CLR @SR3 ;;TURN OFF MAP BK001
981 BR $SIZEX
982 SMMOUT: MOV @#177744,R4 ;;SAVE MEMORY ERROR REGISTER BK001
983 MOV R4,@#177744 ;;CLEAR BITS IN REGISTER BK001
984 BIT #1,R4 ;;MEMORY TIMEOUT? BK001
985 BNE SKTOUT ;;YES-EXIT BK001
986 RTI ;;MUST BE PARITY ERROR-IGNORE IT BK001
987 SKTNEX: BIC #100000,SKT11 ;;KT11 NON-EXISTENT
988 SCORE: MOV #SCROUT,@ERRVEC ;;SET FOR TIMEOUT
989 CLR R2 ;;SET UP BANK
990 1$: ADD #4000,R1 ;;INCREMENT BY 1K
991 ADD #40,R2 ;;1K STEP
992 TST (R1) ;;TRAP ON TIME OUT
993 CMP #177776,R1 ;;LAST ONE
994 BNE 1$ ;;NO--TRY AGAIN
995 SCROUT: SUB #4000,R1
996 $SIZEX: SUB #40,R2 ;;DROP BACK
997 .IFF
998 MOV #2$,@ERRVEC ;;SET FOR TIMEOUT
999 MOV #20000,R1 ;;FIRST ADDRESS
1000 1$: TST (R1) ;;TEST THIS ADDRESS
  
```

```

1001          TST      (R1)+          ;;STEP TO NEXT ADDRESS
1002          BR       1$             ;;TRY ANOTHER
1003          2$:     SUB      #2,R1    ;;DROP BACK
1004          .IFTF
1005          MOV      R0,SP           ;;RESTORE THE STACK
1006          MOV      (SP)+,@ERRVEC+2 ;;RESTORE ERROR VECTOR
1007          MOV      (SP)+,@ERRVEC
1008          MOV      (SP)+,@116      ;;RESTORE MEMORY ERROR VECTOR
1009          MOV      (SP)+,@114
1010          MOV      R1,$LSTAD      ;;LAST ADDRESS
1011          .IFT
1012          MOV      R2,$LSTBK      ;;LAST BANK
1013          MOV      (SP)+,R4        ;;RESTORE R4
1014          MOV      (SP)+,R3        ;;RESTORE R3
1015          MOV      (SP)+,R2        ;;RESTORE R2
1016          .IFTF
1017          MOV      (SP)+,R1        ;;RESTORE R1
1018          MOV      (SP)+,R0        ;;RESTORE R0
1019          RTS      PC
1020          $LSTAD: .WORD 0          ;;CONTAINS THE LAST ADDRESS
1021          .IFT
1022          $LSTBK: .WORD 0          ;;CONTAINS THE LAST BANK
1023          .ENDC
1024          .ENDM  .$SIZE
  
```

BK001

```

1026 .MACRO .STYPE
1027 .SBTTL TYPE ROUTINE
1028 STARS
1029 ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1030 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1031 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1032 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1033 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1034 ;*
1035 ;*CALL:
1036 ;*1) USING A TRAP INSTRUCTION
1037 ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1038 ;*OR
1039 ;* TYPE
1040 ;* MESADR
1041 ;*
1042 $TYPE: TSTB $TPFLG1 ;;IS THERE A TERMINAL?
1043 BPL 1$ ;;BR IF YES
1044 HALT ;;HALT HERE IF NO TERMINAL
1045 BR 3$ ;;LEAVE
1046 1$: MOV R0,-(SP) ;;SAVE R0
1047 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
1048 .IF DF 'MAIL
1049 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
1050 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
1051 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1052 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
1053 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
1054 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
1055 61$: .WORD 0 ;;MESSAGE ADDRESS
1056 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
1057 BNE 60$ ;;YES,SKIP TYPE OUT
1058 .ENDC
1059 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1060 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
1061 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
1062 60$: MOV (SP)+,R0 ;;RESTORE R0
1063 3$: ADD #2,(SP) ;;ADJUST RETURN PC
1064 RTI ;;RETURN
1065 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
1066 BEQ 8$
1067 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
1068 BNE 5$
1069 TST (SP)+ ;;POP <CR><LF> EQUIV
1070 MOV PC,-(SP)
1071 ADD #CRLF1-,(SP)
1072 MOV (SP)+,20$
1073 TYPE ;;TYPE A CR AND LF
1074 20$: .WORD $CRLF1
1075 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
1076 BR 2$ ;;GET NEXT CHARACTER
1077 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
1078 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
1079 BNE 2$ ;;IF NO GO GET NEXT CHAR.
1080 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
1081 ;;AND THE NULL CHAR.
1082 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?

```

```

1083          BLT      6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
1084          JSR      PC,$TYPEC  ;;GO TYPE A NULL
1085          DECB     $CHARCNT    ;;DO NOT COUNT AS A COUNT
1086          BR       7$          ;;LOOP
1087          ;HORIZONTAL TAB PROCESSOR
1088      8$:      MOVB     #' ,(SP)  ;;REPLACE TAB WITH SPACE
1089      9$:      JSR      PC,$TYPEC  ;;TYPE A SPACE
1090          BITB     #7,$CHARCNT    ;;BRANCH IF NOT AT
1091          BNE      9$           ;;TAB STOP
1092          TST      (SP)+        ;;POP SPACE OFF STACK
1093          BR       2$          ;;GET NEXT CHARACTER
1094          $TYPEC: TSTB    @STKS1    ;;CHAR IN KYBD BUFFER?           :MJD001
1095          BPL      10$         ;;BR IF NOT                       :MJD001
1096          MOV      @STKB1,-(SP)   ;;GET CHAR                       :MJD001
1097          BIC      #177600,(SP)  ;;STRIP EXTRANEIOUS BITS        :MJD001
1098          CMPB     #$XOFF,(SP)   ;;WAS CHAR XOFF                 :MJD001
1099          BNE      102$        ;;BR IF NOT                       :MJD001
1100      101$:    TSTB    @STKS1    ;;WAIT FOR CHAR                   :MJD001
1101          BPL      101$        ;;                               :MJD001
1102          MOVB     @STKB1,(SP)   ;;GET CHAR                       :MJD001
1103          BIC      #177600,(SP)  ;;STRIP IT                       :MJD001
1104          CMPB     #$XON,(SP)   ;;WAS IT XON?                   :MJD001
1105          BNE      101$        ;;BR IF NOT                       :MJD001
1106      102$:    TST      (SP)+        ;;FIX STACK                       :MJD001
1107      10$:      TSTB    @STPS1    ;;WAIT UNTIL PRINTER IS READY    :MJD001
1108          BPL      10$         ;;                               :MJD001
1109          MOVB     2(SP),@STPB1   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1110          CMPB     #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
1111          BNE      1$           ;;BRANCH IF NO
1112          CLRB     $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
1113          BR       $TYPEX       ;;EXIT
1114      1$:      CMPB     #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
1115          BEQ      $TYPEX       ;;BRANCH IF YES
1116          INCB     (PC)+        ;;COUNT THE CHARACTER
1117          $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
1118          $TYPEX:  RTS      PC
1119          .IIF NDF HT,HT= 11      ;;CODE FOR HORIZONTAL TAB
1120          .IIF NDF $TKS1,$TKS1:   ;;                               :MJD001
1121          .IIF EQ .-$TKS1,$TKS1: .WORD 177560      ;;TTY KDB STATUS           :MJD001
1122          .IIF NDF $TKB1,$TKB1:   ;;                               :MJD001
1123          .IIF EQ .-$TKB1,$TKB1: .WORD 177562      ;;TTY KBD BUFFER           :MJD001
1124          .IIF NDF $XON,$XON = 21 ;;                               :MJD001
1125          .IIF NDF $XOFF,$XOFF = 23 ;;                               :MJD001
1126          .IIF NDF LF,LF= 12      ;;CODE FOR LINE FEED
1127          .IIF NDF CR,CR= 15      ;;CODE FOR CARRIAGE RETURN
1128          .IIF NDF CRLF,CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
1129          .IIF NDF $TPS1,$TPS1:   ;;                               :MJD001
1130          .IIF EQ .-$TPS1,$TPS1: .WORD 177564      ;;TTY PRINTER STATUS REG. ADDRESS
1131          .IIF NDF $TPB1,$TPB1:   ;;                               :MJD001
1132          .IIF EQ .-$TPB1,$TPB1: .WORD 177566      ;;TTY PRINTER BUFFER REG. ADDRESS
1133          .IIF NDF $NULL,$NULL:   ;;                               :MJD001
1134          .IIF EQ .-$NULL,$NULL: .BYTE 0          ;;CONTAINS NULL CHARACTER FOR FILLS
1135          .IIF NDF $FILLS,$FILLS: ;;                               :MJD001
1136          .IIF EQ .-$FILLS,$FILLS: .BYTE 2        ;;CONTAINS # OF FILLER CHARACTERS RE
1137          .IIF NDF $FILLC,$FILLC: ;;                               :MJD001
1138          .IIF EQ .-$FILLC,$FILLC: .BYTE 12       ;;INSERT FILL CHARS. AFTER A 'LINE F
1139          .IIF NDF $TPFLG1,$TPFLG1:

```

```
1140 .IIF EQ .-STPFLG1,$TPFLG1: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>
1141 .IIF NDF $QUES,$QUES:
1142 .IIF EQ .-$QUES,$QUES: .ASCII '?' ;:QUESTION MARK
1143 .IIF NDF $CRLF1,$CRLF1:
1144 .IIF EQ .-$CRLF1,$CRLF1: .ASCII <15> ;:CARRAIGE RETURN
1145 .IIF NDF $LF1,$LF1:
1146 .IIF EQ .-$LF1,$LF1: .ASCIIZ <12> ;:LINEFEED
1147 .IIF NE 18., .EVEN
1148 .ENDM .$TYPE
```

```

1150 .MACRO .$TYPOCT
1151 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
1152 STARS
1153 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1154 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1155 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1156 ;*CALL:
1157 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1158 ;*      TYPOS    ;;CALL FOR TYPEOUT
1159 ;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1160 ;*      .BYTE   M              ;;M=1 OR 0
1161 ;*                                     ;;1=TYPE LEADING ZEROS
1162 ;*                                     ;;0=SUPPRESS LEADING ZEROS
1163 ;*
1164 ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1165 ;*$TYPOS OR $TYPOC
1166 ;*CALL:
1167 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1168 ;*      TYPON    ;;CALL FOR TYPEOUT
1169 ;*
1170 ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1171 ;*CALL:
1172 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1173 ;*      TYPOC    ;;CALL FOR TYPEOUT
1174 $TYPOS: MOV      @(SP),-(SP)    ;;PICKUP THE MODE
1175         MOV      1(SP), $OFILL  ;;LOAD ZERO FILL SWITCH
1176         MOV      (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
1177         ADD      #2,(SP)       ;;ADJUST RETURN ADDRESS
1178         BR      $TYPON
1179 $TYPOC: MOV      #1, $OFILL     ;;SET THE ZERO FILL SWITCH
1180         MOV      #6, $OMODE+1  ;;SET FOR SIX(6) DIGITS
1181 $TYPON: MOV      #5, $OCNT     ;;SET THE ITERATION COUNT
1182         MOV      R3,-(SP)      ;;SAVE R3
1183         MOV      R4,-(SP)      ;;SAVE R4
1184         MOV      R5,-(SP)      ;;SAVE R5
1185         MOV      $OMODE+1,R4   ;;GET THE NUMBER OF DIGITS TO TYPE
1186         NEG      R4
1187         ADD      #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
1188         MOV      R4, $OMODE    ;;SAVE IT FOR USE
1189         MOV      $OFILL,R4    ;;GET THE ZERO FILL SWITCH
1190         MOV      12(SP),R5    ;;PICKUP THE INPUT NUMBER
1191         CLR      R3          ;;CLEAR THE OUTPUT WORD
1192 1$:     ROL      R5          ;;ROTATE MSB INTO 'C'
1193         BR      3$          ;;GO DO MSB
1194 2$:     ROL      R5          ;;FORM THIS DIGIT
1195         ROL      R5
1196         ROL      R5
1197         MOV      R5,R3
1198 3$:     ROL      R3          ;;GET LSB OF THIS DIGIT
1199         DECB    $OMODE      ;;TYPE THIS DIGIT?
1200         BPL    7$          ;;BR IF NO
1201         BIC    #177770,R3   ;;GET RID OF JUNK
1202         BNE    4$          ;;TEST FOR 0
1203         TST    R4          ;;SUPPRESS THIS 0?
1204         BEQ    5$          ;;BR IF YES
1205 4$:     INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S
1206         BIS    #'0,R3     ;;MAKE THIS DIGIT ASCII

```

```

1207          5$:  BIS      #' ,R3          ;;MAKE ASCII IF NOT ALREADY
1208          MOVB    R3,8$              ;;SAVE FOR TYPING
1209          MOV     PC,-(SP)           ;;FETCH ADDRESS OF NUMBER TO BE TYPED USING
1210                                     ;;POSITION INDEPENDENT CODE
1211          ADD     #8$-.,(SP)         ;;STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
1212                                     ;;PRINTED
1213          MOV     (SP)+,10$          ;;SET UP ADDRESS FOR TYPE COMMAND
1214          TYPE                                     ;;GO TYPE THIS DIGIT
1215          10$:  .WORD  8$              ;;ADDRESS OF CHARACTER TO BE PRINTED
1216          7$:  DECB   $OCNT           ;;COUNT BY 1
1217          BGT    2$                  ;;BR IF MORE TO DO
1218          BLT    6$                  ;;BR IF DONE
1219          INC    R4                  ;;INSURE LAST DIGIT ISN'T A BLANK
1220          BR     2$                  ;;GO DO THE LAST DIGIT
1221          6$:  MOV     (SP)+,R5        ;;RESTORE R5
1222          MOV     (SP)+,R4        ;;RESTORE R4
1223          MOV     (SP)+,R3        ;;RESTORE R3
1224          MOV     2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
1225          MOV     (SP)+,(SP)
1226          RTI                                     ;;RETURN
1227          8$:  .BYTE  0              ;;STORAGE FOR ASCII DIGIT
1228          .BYTE  0              ;;TERMINATOR FOR TYPE ROUTINE
1229          $OCNT: .BYTE  0              ;;OCTAL DIGIT COUNTER
1230          $OFILL: .BYTE  0           ;;ZERO FILL SWITCH
1231          $OMODE: .WORD  0           ;;NUMBER OF DIGITS TO TYPE
1232          .ENDM  .$TYPOCT
  
```

```

1234 .MACRO .STYPDEC
1235 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1236 STARS
1237 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1238 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1239 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1240 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1241 ;*REPLACED WITH SPACES.
1242 ;*CALL:
1243 ;*   MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1244 ;*   TYPDS  ;;GO TO THE ROUTINE
1245 $TYPDS: PUSH  <R0,R1,R2,R3,R4,R5>
1246   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1247   MOV     22(SP),R5       ;;GET THE INPUT NUMBER
1248   BPL    1$              ;;BR IF INPUT IS POS.
1249   NEG    R5              ;;MAKE THE BINARY NUMBER POS.
1250   MOVB   #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1251 1$:   CLR    R0           ;;ZERO THE CONSTANTS INDEX
1252   MOV    PC,R3          ;;FETCH PC FOR ADDRESSING
1253   ADD    #SDBLK-.,R3    ;;SETUP THE OUTPUT POINTER - R3 POINTS TO SDBLK
1254   ;;TABLE
1255   MOVB   #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
1256 2$:   CLR    R2           ;;CLEAR THE BCD NUMBER
1257   MOV    PC,R4          ;;USE PC FOR LOCATING TABLE
1258   ADD    #SDTBL-.,R4    ;;R4 NOW POINTS TO TABLE
1259   ADD    R0,R4          ;;INDEX INTO TABLE
1260   MOV    (R4),R1        ;;GET THE CONSTANT
1261 3$:   SUB    R1,R5       ;;FORM THIS BCD DIGIT
1262   BLT    4$            ;;BR IF DONE
1263   INC    R2            ;;INCREASE THE BCD DIGIT BY 1
1264   BR    3$
1265 4$:   ADD    R1,R5       ;;ADD BACK THE CONSTANT
1266   TST    R2            ;;CHECK IF BCD DIGIT=0
1267   BNE    5$           ;;FALL THROUGH IF 0
1268   TSTB   (SP)          ;;STILL DOING LEADING 0'S?
1269   BMI    7$           ;;BR IF YES
1270 5$:   ASLB   (SP)       ;;MSD?
1271   BCC    6$           ;;BR IF NO
1272   MOVB   1(SP),-1(R3)   ;;YES--SET THE SIGN
1273 6$:   BIS    #'0,R2      ;;MAKE THE BCD DIGIT ASCII
1274 7$:   BIS    #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1275   MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1276   TST    (R0)+         ;;JUST INCREMENTING
1277   CMP    R0,#10        ;;CHECK THE TABLE INDEX
1278   BLT    2$           ;;GO DO THE NEXT DIGIT
1279   BGT    8$           ;;GO TO EXIT
1280   MOV    R5,R2         ;;GET THE LSD
1281   BR    6$           ;;GO CHANGE TO ASCII
1282 8$:   TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
1283   BPL    9$           ;;BR IF NO
1284   MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
1285 9$:   CLRB   (R3)       ;;SET THE TERMINATOR
1286   POP    <R5,R4,R3,R2,R1,R0>
1287   MOV    PC,-(SP)      ;;FETCH ADDRESS OF NUMBER TO BE TYPED USING
1288   ;;POSITION INDEPENDENT CODE
1289   ADD    #SDBLK-.,(SP) ;;STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
1290   ;;PRINTED

```



```
1291          MOV      (SP)+,10$      ;;SET UP ADDRESS FOR TYPE COMMAND
1292          TYPE                                ;;NOW TYPE THE NUMBER
1293          10$: .WORD  $DBLK
1294          MOV      2(SP),4(SP)      ;;ADJUST THE STACK
1295          MOV      (SP)+,(SP)
1296          RTI                                ;;RETURN TO USER
1297          $DTBL: 10000.
1298                   1000.
1299                   100.
1300                   10.
1301          $DBLK: .BLKW  4
1302          .ENDM  . $TYPDEC
```

```

1304 .MACRO .STRAP X,Y,Z
1305 .SBTTL TRAP DECODER
1306 STARS
1307 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
1308 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1309 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1310 ;*GO TO THAT ROUTINE.
1311 .IF B X
1312 $TRAP: MOV R0,-(SP) ::SAVE R0
1313 MOV R1,-(SP) ::SAVE R1
1314 .IFF
1315 $TRAP: MOV 2(SP),-(SP) ::ASSUME THE STATUS OF
1316 BIC #20,(SP) ::THE CALLER--DO NOT ALLOW
1317 MOV #1$,-(SP) ::T-BIT TRAPS
1318 RTI ::SET THE NEW STATUS
1319 1$: MOV R0,-(SP) ::SAVE R0
1320 .ENDC
1321 MOV 4(SP),R0 ::GET TRAP ADDRESS
1322 TST -(R0) ::BACKUP BY 2
1323 MOVB (R0),R0 ::GET RIGHT BYTE OF TRAP
1324 .IF NB Z
1325 BPL $TRAP1 ::NON-USER TRAP,BELOW 200
1326 BIC #^C177,R0 ::STRIP AWAY THE JUNK
1327 JMP (PC) ::USER TRAP,ABOVE 177, GO TO
1328 .WORD Z ::USER TRAP HANDLER- Z
1329 $TRAP1:
1330 .ENDC
1331 .IF NB Y
1332 CMP # $TERM,R0 ::CHECK FOR OUT OF BOUNDS
1333 BGT .+6 ::BR IF OK
1334 HALT ::OUT OF BOUNDS
1335 BR .-2 ::HANGUP
1336 .ENDC
1337 ASL R0 ::POSITION FOR INDEXING
1338 MOV PC,R1 ::FETCH THIS PROGRAM POINTER FOR PIC
1339 ADD # $TRPAD-.,R1 ::POINT TO TABLE
1340 ADD R0,R1 ::R1 NOW POINTS TO ROUTINE TRAP CALL WANTED
1341 MOV (R1),R0 ::R0 CONTAINS ADDRESS FOR RTS
1342 MOV (SP)+,R1 ::RESTORE R1
1343 RTS R0 ::GO TO ROUTINE
1344 ::THIS IS USE TO HANDLE THE 'GETPRI' MACRO
1345 $TRAP2: MOV (SP),-(SP) ::MOVE THE PC DOWN
1346 MOV 4(SP),2(SP) ::MOVE THE PSW DOWN
1347 RTI ::RESTORE THE PSW
1348 .MACRO SETTRAP A,B,MSG
1349 $$SET A,B,\<TRAP+$TRP>,\$TRP,<MSG>
1350 .NLIST
1351 $TRP=$TRP+1
1352 .LIST
1353 .ENDM SETTRAP
1354 .MACRO $$SET A,B,C,D,COMNT
1355 .IF EQ $TRP-1
1356 .SBTTL TRAP TABLE
1357 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1358 ;*BY THE 'TRAP' INSTRUCTION.
1359 : ROUTINE
1360 : -----
  
```

```

1361 $TRPAD: .WORD $TRAP2
1362 .ENDC
1363 .IIF NDF GNS,.NLIST
1364 A= C
1365 .IIF NDF GNS,.LIST
1366 B ::CALL=A TRAP+D(C) COMNT
1367 .ENDM $$SET
1368 .MACRO TRMTRP
1369 $TERM=-$TRPAD
1370 .ENDM TRMTRP
1371 .NLIST
1372 $TRP=1
1373 .LIST
1374 .IF DF $TYPE
1375 SETTRAP TYPE,$TYPE,^/TTY TYPEOUT ROUTINE/
1376 .ENDC
1377 .IF DF $TYPOC
1378 SETTRAP TYPOC,$TYPOC,^/TYPE OCTAL NUMBER (WITH LEADING ZEROS)/
1379 SETTRAP TYPOS,$TYPOS,^/TYPE OCTAL NUMBER (NO LEADING ZEROS)/
1380 SETTRAP TYPON,$TYPON,^/TYPE OCTAL NUMBER (AS PER LAST CALL)/
1381 .ENDC
1382 .IF DF $TYPDS
1383 SETTRAP TYPDS,$TYPDS,^/TYPE DECIMAL NUMBER (WITH SIGN)/
1384 .ENDC
1385 .IF DF $TYPBN
1386 SETTRAP TYPBN,$TYPBN,^/TYPE BINARY (ASCII) NUMBER/
1387 .ENDC
1388 .IF DF $GTSWR
1389 SETTRAP GTSWR,$GTSWR,^/GET SOFT-SWR SETTING/
1390 .ENDC
1391 .IF DF $CKSWR
1392 SETTRAP CKSWR,$CKSWR,^/TEST FOR CHANGE IN SOFT-SWR/
1393 .ENDC
1394 .IF DF $RDCHR
1395 SETTRAP RDCHR,$RDCHR,^/TTY TYPEIN CHARACTER ROUTINE/
1396 .ENDC
1397 .IF DF $RDLIN
1398 SETTRAP RDLIN,$RDLIN,^/TTY TYPEIN STRING ROUTINE/
1399 .ENDC
1400 .IF DF $RDOCT
1401 SETTRAP RDOCT,$RDOCT,^/READ AN OCTAL NUMBER FROM TTY/
1402 .ENDC
1403 .IF DF $RDDEC
1404 SETTRAP RDDEC,$RDDEC,^/READ A DECIMAL NUMBER FROM TTY/
1405 .ENDC
1406 .IF DF $$SAVREG
1407 SETTRAP SAVREG,$SAVREG,^/SAVE R0-R5 ROUTINE/
1408 SETTRAP RESREG,$RESREG,^/RESTORE R0-R5 ROUTINE/
1409 .ENDC
1410 .IF DF $R2A
1411 SETTRAP R2AZ,$R2AZ
1412 SETTRAP R2AZ.,$R2AZ.
1413 SETTRAP R2AZQ,$R2AZQ
1414 .ENDC
1415 .ENDM . $TRAP
  
```

```

1417
1418 000000 .HEADER <CVMEMAO DIAGNOSTIC FOR NON-VOLATILE MEMORIES>,<1981>,<KEN LANGLAIS>
1419 000000 .EQUAT
1420 000000 .SWRHI
1421 000000 .SCATCH START1
1422 000204 .KT11 .X
1423
1424 000204 .SETUP .STRAP
1425 .NLIST BEX
1426 .LIST MEB
1427 000001 $LSTIN = 1
1428 000001 $LSTTAG = 1
1429 000400 . ;READ ROUTINES
1430
1431 000400 .SREAD 26
000400 177560 .IIF EQ .-$TKS,$TKS: .WORD 177560 ;;TTY KBD STATUS
000402 177562 .IIF EQ .-$TKB,$TKB: .WORD 177562 ;;TTY KBD BUFFER
000404 011646 $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
000406 016666 000004 000002 MOV 4(SP),2(SP) ;;SAVE THE PS
000414 105777 177760 1$: TSTB @TKS ;;WAIT FOR
000420 100375 BPL 1$ ;;A CHARACTER
000422 117766 177754 000004 MOVB @TKB,4(SP) ;;READ THE TTY
000430 042766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
000436 026627 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
000444 001013 BNE 3$ ;;BRANCH IF NO
000446 105777 177726 2$: TSTB @TKS ;;WAIT FOR A CHARACTER
000452 100375 BPL 2$ ;;LOOP UNTIL ITS THERE
000454 117746 177722 MOVB @TKB,-(SP) ;;GET CHARACTER
000460 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
000464 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
000470 001366 BNE 2$ ;;IF NOT DISCARD IT
000472 000750 BR 1$ ;;YES, RESUME
000474 026627 000004 000021 3$: CMP 4(SP),#$XON ;;IS IT A RANDOM XON? ;RAN001
000502 001744 BEQ 1$ ;;BRANCH IF YES ;RAN001
000504 026627 000004 000140 CMP 4(SP),#140 ;;IS IT UPPER CASE?
000512 002407 BLT 4$ ;;BRANCH IF YES
000514 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
000522 003003 BGT 4$ ;;BRANCH IF YES
000524 042766 000040 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
000532 000002 4$: RTI ;;GO BACK TO USER
000534 010346 $RDLIN: MOV R3,-(SP) ;;SAVE R3
000536 012703 000642 1$: MOV #TTYIN,R3 ;;GET ADDRESS
000542 022703 000670 2$: CMP #TTYIN+26,R3 ;;BUFFER FULL?
000546 101405 BLOS 4$ ;;BR IF YES
000550 104406 RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
000552 112613 MOVB (SP)+,(R3) ;;GET CHARACTER
000554 122713 000177 10$: CMPB #177,(R3) ;;IS IT A RUBOUT
000560 001003 BNE 3$ ;;SKIP IF NOT
000562 104401 000670 4$: TYPE ,SQUES ;;TYPE A '?'
000566 000763 BR 1$ ;;CLEAR THE BUFFER AND LOOP
000570 111367 000044 3$: MOVB (R3),9$ ;;ECHO THE CHARACTER
000574 104401 000640 TYPE ,9$
000600 122723 000015 CMPB #15,(R3)+ ;;CHECK FOR RETURN
000604 001356 BNE 2$ ;;LOOP IF NOT RETURN
000606 105063 177777 CLR B -1(R3) ;;CLEAR RETURN (THE 15)
000612 104401 000672 TYPE ,LF ;;TYPE A LINE FEED
000616 012603 MOV (SP)+,R3 ;;RESTORE R3
  
```

	000620	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
	000622	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
	000630	012766	000642	000004	MOV	#TTYIN,4(SP)	
	000636	000002			RTI		::RETURN
	000640	000			9\$: .BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
	000641	000			.BYTE	0	::TERMINATOR
	000670	077			.IIF EQ	.-\$QUES,\$QUES: .ASCII	::QUESTION MARK
	000671	015			.IIF EQ	.-\$CRLF,\$CRLF: .ASCII	<15> ::CARRIAGE RETURN
	000672	012	000		.IIF EQ	.-\$LF,\$LF: .ASCII	<12> ::LINE FEED
	000674	136	125	015	\$CNTLU:	.ASCIZ /^U/<15><12>	::CONTROL 'U'
	000701	136	107	015	\$CNTLG:	.ASCIZ /^G/<15><12>	::CONTROL 'G'
	000706	015	012	123	\$MSWR:	.ASCIZ <15><12>/SWR = /	
	000717	040	040	116	\$MNEW:	.ASCIZ / NEW = /	
1432							
1433	000730				.SRDOCT	BIG	
	000730	011646			\$RDOCT: MOV	(SP),-(SP)	::PROVIDE SPACE FOR THE
	000732	016666	000004	000002	MOV	4(SP),2(SP)	::INPUT NUMBER
	000740	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
	000742	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
	000744	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
	000746	104407			1\$: RDLIN		::READ AN ASCII LINE
	000750	012600			MOV	(SP)+,R0	::GET ADDRESS OF 1ST CHARACTER
	000752	010067	000100		MOV	R0,5\$::AND SAVE IT
	000756	005001			CLR	R1	::CLEAR DATA WORD
	000760	005002			CLR	R2	
	000762	112046			2\$: MOVB	(R0)+,-(SP)	::PICKUP THIS CHARACTER
	000764	001420			BEQ	3\$::IF ZERO GET OUT
	000766	122716	000060		CMPB	#'0,(SP)	::MAKE SURE THIS CHARACTER
	000772	003026			BGT	4\$::IS AN OCTAL DIGIT
	000774	122716	000067		CMPB	#'7,(SP)	
	001000	002423			BLT	4\$	
	001002	006301			ASL	R1	::*2
	001004	006102			ROL	R2	
	001006	006301			ASL	R1	::*4
	001010	006102			ROL	R2	
	001012	006301			ASL	R1	::*8
	001014	006102			ROL	R2	
	001016	042716	177770		BIC	^C7,(SP)	::STRIP THE ASCII JUNK
	001022	062601			ADD	(SP)+,R1	::ADD IN THIS DIGIT
	001024	000756			BR	2\$::LOOP
	001026	005726			3\$: TST	(SP)+	::CLEAN TERMINATOR FROM STACK
	001030	010166	000012		MOV	R1,12(SP)	::SAVE THE RESULT
	001034	010267	000026		MOV	R2,\$HIOCT	
	001040	012602			MOV	(SP)+,R2	::POP STACK INTO R2
	001042	012601			MOV	(SP)+,R1	::POP STACK INTO R1
	001044	012600			MOV	(SP)+,R0	::POP STACK INTO R0
	001046	000002			RTI		::RETURN
	001050	005726			4\$: TST	(SP)+	::CLEAN PARTIAL FROM STACK
	001052	105010			CLRB	(R0)	::SET A TERMINATOR
	001054	104401			TYPE		::TYPE UP THRU THE BAD CHAR.
	001056	000000			5\$: .WORD	0	
	001060	104401	000670		TYPE	,\$QUES	::'?' 'CR' & 'LF'
	001064	000730			BR	1\$::TRY AGAIN
	001066	000000			\$HIOCT: .WORD	0	::HIGH ORDER BITS GO HERE
1434							
1435	001070				.SRDDEC		
	001070	011646			\$RDDEC: MOV	(SP),-(SP)	::PROVIDE SPACE FOR

```

001072 016666 000004 000002      MOV      4(SP),2(SP)      ;;THE INPUT NUMBER
001100 010046                    MOV      R0,-(SP)        ;;PUSH R0 ON STACK
001102 010146                    MOV      R1,-(SP)        ;;PUSH R1 ON STACK
001104 010246                    MOV      R2,-(SP)        ;;PUSH R2 ON STACK
001106 104407      1$:      RDLIN                    ;;READ AN ASCIZ LINE
001110 012600                    MOV      (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
001112 010067 000120      MOV      R0,6$          ;;SAVE INCASE OF BAD INPUT
001116 005046                    CLR      -(SP)          ;;CLEAR DATA WORD
001120 005002                    CLR      R2             ;;SIGN SET POSITIVE
001122 122710 000055      CMPB    #'-',(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
001126 001001                    BNE     2$              ;;BR IF NO MINUS SIGN
001130 112002                    MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
001132 112001      2$:      MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
001134 001424                    BEQ     3$              ;;GET OUT IF ZERO
001136 122701 000060      CMPB    #'0,R1         ;;MAKE SURE THIS CHARACTER
001142 003032                    BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
001144 122701 000071      CMPB    #'9,R1
001150 002427                    BLT     5$
001152 032716 170000      BIT     #'^C7777,(SP)   ;;DON'T LET NUMBEP GET TO BIG
001156 001024                    BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
001160 006316                    ASL     (SP)            ;;*2
001162 011646                    MOV     (SP),-(SP)      ;;SAVE FOR LATER
001164 006316                    ASL     (SP)            ;;*4
001166 006316                    ASL     (SP)            ;;*8
001170 062616                    ADD     (SP)+,(SP)      ;;*10
001172 102416                    BVS     5$              ;;OVERFLOW ISN'T ALLOWED
001174 162701 000060      SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
001200 060116                    ADD     R1,(SP)         ;;ADD IN THIS DIGIT
001202 102412                    BVS     5$              ;;OVERFLOW ISN'T ALLOWED
001204 000752                    BR      2$              ;;LOOP
001206 005702      3$:      TST     R2             ;;CHECK IF NUMBER IS NEG
001210 001401                    BEQ     4$              ;;BR IF NO
001212 005416                    NEG     (SP)            ;;YES--NEGATE THE NUMBER
001214 012666 000012      4$:      MOV     (SP)+,12(SP)   ;;SAVE THE RESULT
001220 012602                    MOV     (SP)+,R2        ;;POP STACK INTO R2
001222 012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
001224 012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
001226 000002                    RTI
001230 005726      5$:      TST     (SP)+          ;;CLEAN PARTIAL NUMBER FROM STACK
001232 105010                    CLRE   (R0)            ;;SET A TERMINATOR
001234 104401                    TYPE
001236 000000      6$:      .WORD  0              ;;TYPE THE INPUT UP TO BAD CHAR.
001240 104401 000670      TYPE    ,QUES          ;;POINTER GOES HERE
001244 000720                    BR      1$              ;;'?' 'CR' & 'LF'
;;TRY AGAIN

1436
1437 001246      .SSIZE
001246 010046      $SIZE: MOV    R0,-(SP)        ;;SAVE R0 ON THE STACK
001250 010146      MOV    R1,-(SP)        ;;SAVE R1 ON THE STACK
001252 010246      MOV    R2,-(SP)        ;;SAVE R2 ON THE STACK
001254 010346      MOV    R3,-(SP)        ;;SAVE R3 ON THE STACK
001256 010446      MOV    R4,-(SP)        ;;SAVE R4 ON THE STACK
001260 013746 000114      MOV    @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
001264 013746 000116      MOV    @#116,-(SP)
001270 012737 000116 000114      MOV    #116,@#114     ;;IGNORE PARITY ERRORS WHILE SIZING
001276 012737 000002 000116      MOV    #RTI,@#116
001304 013746 000004      MOV    @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
001310 013746 000006      MOV    @#ERRVEC+2,-(SP)

```

BK001

Address	Offset	Value	Label	Instruction	Comments	Bank
001314	010600			MOV SP,R0	::SAVE THE STACK POINTER	
001316	104400			TRAP	::PUSH OLD PSW AND PC ON STACK	
001320	012637	000006		MOV (SP)+,@ERRVEC+2	::SAVE THE PSW IN @ERRVEC+2	
001324	012701	003776		MOV #3776,R1	::SETUP ADDRESS	
001330	105727			TSTB (PC)+	::USE MEMORY MANAGEMENT?	
001332	000200		\$KT11:	.WORD 200	::SET TO USE MEMORY MANAGEMENT	
001334	100135			BPL \$CORE	::BR IF NO	
001336	012737	001622	000004	MOV #SKTNEX,@ERRVEC	::SET FOR TIMEOUT	
001344	005737	177572		@SR0	::KT11 ARE YOU THERE?	
001350	052767	100000	177754	BIS #100000,\$KT11	::YES--SET KT11 KEY	
001356	012737	001406	000004	MOV #100,\$@ERRVEC	::SET FOR TIMEOUT	BK001
001364	005737	172516		TST @172516	::Q-BUS MAP ARE YOU THERE?	BK001
001370	012767	000200	000026	MOV #200,\$MAP	::TURN ON MAP INDICATOR	BK001
001376	012767	176200	000022	MOV #176200,\$STOP	::END OF 2M OF MEMORY	
001404	000411			BR \$MAPRG	::GO SET UP MAP REGISTERS	BK001
001406	012737	006200	001426	100\$: MOV #6200,@\$STOP	::COMPARISON VALUE FOR 18 BIT MAPPING	BK001
001414	022626			CMP (SP)+,(SP)+	::CLEAN OFF STACK	BK001
001416	005037	001424		CLR @\$MAP	::MAKE SURE MAP INDICATOR TURNED OFF	BK001
001422	000402			BR \$NOMAP	::	BK001
001424	000000		\$MAP:	.WORD 0	::=200 IF MAP PRESENT	BK001
001426	000000		\$STOP:	.WORD 0	::FILLED WITH APPROPRIATE COMPARISON VALUE	BK001
001430	005046			CLR -(SP)	::INITIALIZE FOR 'PAR' LOADING	
001432	012702	172340		MOV #KIPAR0,R2	::ADDRESS OF FIRST 'PAR'	
001436	012703	000010		MOV #8,R3	::LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'	
001442	012762	077406	177740	1\$: MOV #77406,-40(R2)	::PDR = 4K, UP, READ/WRITE	
001450	011622			MOV (SP),(R2)+	::LOAD 'PAR'	
001452	062716	000200		ADD #200,(SP)	::UPDATE FOR NEXT 'PAR'	
001456	077307			SOB R3,1\$::LOOP UNTIL ALL EIGHT ARE LOADED	
001460	012742	177600		MOV #177600,-(R2)	::SETUP KIPAR7 FOR I/O	
001464	005042			CLR -(R2)	::SETUP KIPAR6 FOR TESTING	
001466	012737	001504	000004	MOV #2\$,@ERRVEC	::CATCH TIMEOUT IF NO SR3	
001474	012737	000020	172516	MOV #20,@SR3	::ENABLE 22 BIT MODE AND UNIBUS MAP	BK001
001502	000401			BR 3\$::THIS PDP-11 HAS A SR3 REGISTER	
001504	022626		2\$:	CMP (SP)+,(SP)+	::CLEAN OFF THE STACK--NO SR3	
001506	005237	177572		3\$: INC @SR0	::TURN ON MEMORY MANAGEMENT	
001512	012737	001560	000004	MOV #SKTOUT,@ERRVEC	::SET FOR TIME OUT	
001520	105737	001424		TSTB @\$MAP	::IS MAP THERE?	BK001
001524	100006			BPL 4\$::NO-SKIP	BK001
001526	012737	001602	000114	MOV #SMMOUT,@114	::SET UP MEMORY ERROR VECTOR	BK001
001534	013737	000006	000116	MOV @ERRVEC+2,@116	::LOCK OUT INTERRUPTS	BK001
001542	005737	143776		4\$: TST @143776	::TRAP ON NON-EX-MEM	
001546	062712	000040		ADD #40,(R2)	::MAKE A 1K STEP	
001552	023712	001426		CMP @\$STOP,(R2)	::LAST ONE?	
001556	101371			BHI 4\$::NO--TRY IT	
001560	011202		\$KTOUT:	MOV (R2),R2	::GET LAST BANK+1	
001562	005037	177572		CLR @SR0	::TURN OFF MEMORY MANAGEMENT	
001566	105737	001424		TSTB @\$MAP	::IS MAP THERE?	BK001
001572	100034			BPL \$SIZEX	::NO-SKIP	BK001
001574	005037	172516		CLR @SR3	::TURN OFF MAP	BK001
001600	000431			BR \$SIZEX	::	
001602	013704	177744		\$MMOUT: MOV @177744,R4	::SAVE MEMORY ERROR REGISTER	BK001
001606	010437	177744		MOV R4,@177744	::CLEAR BITS IN REGISTER	BK001
001612	032704	000001		BIT #1,R4	::MEMORY TIMEOUT?	BK001
001616	001360			BNE \$KTOUT	::YES-EXIT	BK001
001620	000002			RTI	::MUST BE PARITY ERROR-IGNORE IT	BK001
001622	042767	100000	177502	\$KTNEX: BIC #100000,\$KT11	::KT11 NON-EXISTENT	
001630	012737	001660	000004	\$CORE: MOV #SCROUT,@ERRVEC	::SET FOR TIMEOUT	

```

001636 005002          CLR      R2          ;;SET UP BANK
001640 062701 004000 1$:  ADD      #4000,R1      ;;INCREMENT BY 1K
001644 062702 000040      ADD      #40,R2        ;;1K STEP
001650 005711          TST      (R1)          ;;TRAP ON TIME OUT
001652 022701 177776      CMP      #177776,R1     ;;LAST ONE
001656 001370          BNE      1$          ;;NO--TRY AGAIN
001660 162701 004000  $CROUT: SUB     #4000,R1
001664 162702 000040  $SIZEX: SUB     #40,R2        ;;DROP BACK
001670 010006          MOV      R0,SP         ;;RESTORE THE STACK
001672 012637 000006      MOV      (SP)+,@WERRVEC+2 ;;RESTORE ERROR VECTOR
001676 012637 000004      MOV      (SP)+,@WERRVEC
001702 012637 000116      MOV      (SP)+,@W116    ;;RESTORE MEMORY ERROR VECTOR
001706 012637 000114      MOV      (SP)+,@W114
001712 010167 000020      MOV      R1,$LSTAD     ;;LAST ADDRESS
001716 010267 000016      MOV      R2,$LSTBK     ;;LAST BANK
001722 012604          MOV      (SP)+,R4       ;;RESTORE R4
001724 012603          MOV      (SP)+,R3       ;;RESTORE R3
001726 012602          MOV      (SP)+,R2       ;;RESTORE R2
001730 012601          MOV      (SP)+,R1       ;;RESTORE R1
001732 012600          MOV      (SP)+,R0       ;;RESTORE R0
001734 000207          RTS      PC
001736 000000  $LSTAD: .WORD  0          ;;CONTAINS THE LAST ADDRESS
001740 000000  $LSTBK: .WORD  0          ;;CONTAINS THE LAST BANK
  
```

BK001


```
1439 ;THIS ROUTINE FETCHES A CHARACTER FROM THE TTY AND ECHOS IT
1440
1441 001742 ROUTINE FETCH
1442 001742
1443 001742 104406 RDCHR ;FETCH ANSWER
1444
1445 001744 POP @WECHO ;FETCH CHARACTER
001744 012637 010342 MOV (SP)+,@WECHO ;;POP STACK INTO @WECHO
1446
1447 001750 104401 010342 TYPE ,ECHO ;ECHO CHARACTER
1448
1449 001754 104401 016155 TYPE ,CRLF1 ;PRINT <CR><LF>
1450
1451 001760 ENDRTN
001760
001760
001760 000207
1452
1453
```

50000\$:
50001\$:
RTS PC

```

1455 ;THIS ROUTINE READS THE ADDRESS TYPED, DETERMINES THAT IT IS A LEGAL 4K
1456 ;ADDRESS (I.E. DOES NOT RESIDE OUT SIDE OF THE SYSTEM ADDRESS SPACE) AND
1457 ;RETURNS WITH R4 NE #0 IF ADDRESS IS LEGAL
1458
1459
1460 001762          ROUTINE FETADD
1461 001762
1462 001762 104410          RDOCT          ;READ OCTAL NUMBER
1463
1464 001764          POP @#ECHO          ;FETCH BITS 0 - 15
1465 001764 012637 010342  MOV (SP)+,@#ECHO  ;;POP STACK INTO @#ECHO
1466
1466 001770          IF #17777 SETIN @#ECHO THEN          ;CHECK FOR 4K
1467 001770 032737 017777 010342          BIT #17777,@#ECHO
1468 001776 001403          BEQ 50002$
1469
1468 002000 104401 010152          TYPE ,NOT4K          ;BOUNDARY
1469
1470 002004          ELSE
1471 002004 000432          BR 50003$
1472 002006          50002$:
1473 002006 013737 010342 010344  LET @#SAVLOW := @#ECHO          ;SAVE BITS 0 - 15 OF ADDRESS
1474 002014          MOV @#ECHO,@#SAVLOW
1475 002014 013737 001066 010346  LET @#SAVHI :- @#SHIOCT          ;SAVE BITS 16 - 22 OF ADDRESS
1476          MOV @#SHIOCT,@#SAVHI
1477
1478          ;BUMP TO NEXT 4K IN ORDER TO DETERMINE IF THERE IS ENOUGH
1479          ;MEMORY FOR THE PROGRAM AND TESTING
1479 002022          LET @#ECHO := @#ECHO + #20000          ;ADD 4K
1480 002022 062737 020000 010342          ADD #20000,@#ECHO
1481
1481 002030          IF #1 SETIN CONTROL THEN
1482 002030 032767 000001 014122          BIT #1,CONTROL
1483 002036 001402          BEQ 50004$
1484
1483 002040          LET @#SHIOCT :- @#SHIOCT + CARRY          ;ADD OVERFLOW
1484 002040 005537 001066          ADC @#SHIOCT
1485
1485 002044          ENDIF
1486 002044          50004$:
1487 002044          CALL ADDCHK
1488 002044 004767 000024          JSR PC,ADDCHK
1489
1489 002050          IF #1 SETIN CONTROL THEN          ;MAPPED?
1490 002050 032767 000001 014102          BIT #1,CONTROL
1491 002056 001403          BEQ 50005$
1492
1491 002060          LET R3 := R3 - #200          ;REMOVE 4K TEST INCREMENT
1492 002060 162703 000200          SUB #200,R3
1493
1493 002064          ELSE
1494 002064 000402          BR 50006$

```



```

1503          ;THIS ROUTINE CHECKS THAT THE 4K BOUNDARY IS WITHIN IN THE SYSTEM
1504          ;R3 WILL EITHER BE PAR COMPATIBLE ADDRESS FOR A MAPPED SYSTEM OR 4K ADDRESS
1505          ;FOR AN UNMAPPED SYSTEM
1506
1507 002074    ROUTINE ADDCHK
1508 002074
1509 002074    IF #1 SETIN CONTROL THEN
1510 002074 032767 000001 014056
1511 002104    INCR R3 FROM #1 TO #3 BY #1
1512 002104 012703 000001
1513 002110 000401
1514 002112
1515 002112 005203
1516 002114 020327 000003
1517 002120 003005
1518
1519
1520 002122    LET @#ECHO := @#ECHO ROTATE 1
1521 002122 006137 010342
1522
1523 002126    LET @#SHIOCT := @#SHIOCT ROTATE 1
1524 002126 006137 001066
1525
1526 002132    ENDINC
1527 002132 000767
1528
1529 002134    INCR R3 FROM #1 TO #7 BY #1
1530 002134 012703 000001
1531 002140 000401
1532 002142
1533 002142 005203
1534 002144 020327 000007
1535 002150 003003
1536
1537 002152    LET @#SHIOCT := @#SHIOCT SHIFT 1
1538 002152 006337 001066
1539
1540 002156    ENDINC
1541 002156 000771
1542 002160
1543
1544 002160    LET R3 := @#SHIOCT
1545 002160 013703 001066
1546
1547          ELSE
1548
1549          ;NO MEMORY MANAGEMENT
1550
1551 002164    LET R3 := @#ECHO
1552 002166 013703 010342
  
```

ADDCHK:

```

;DO WE HAVE MEM. MAN
BIT #1,CONTROL
BEQ 50002$
;YES
;MAKE ADDRESS IN ECHO AND SHIOCT
MOV #1,R3
BR 50003$
50004$:
INC R3
50005$:
CMP R3,#3
BGT 50005$
;COMPATIBLE WITH PAR
  
```

ROL @#ECHO

ROL @#SHIOCT

BR 50004\$
50005\$:

```

;NOW SHIOCT WILL TAKE LOWER BITS
MOV #1,R3
BR 50006$
50007$:
INC R3
50008$:
CMP R3,#7
BGT 50010$
;AND PUT THEM INTO UPPER BITS
  
```

ASL @#SHIOCT

BR 50007\$
50010\$:

MOV @#SHIOCT,R3

```

;NO MEMORY MANAGEMENT
BR 50011$
50002$:
  
```

```

;FETCH STARTING ADDRESS
MOV @#ECHO,R3
  
```



```

1557 002226 012 012 015 HELPF: .ASCII <12><12><15>' FUNCTIONAL DESCRIPTION'<12><12><15>
1558 002305 040 124 110 .ASCII ' THE PURPOSE OF THIS PROGRAM IS TO TEST THE DATA RETENTION OF'<12><15>
1559 002404 040 116 117 .ASCII ' NON-VOLATILE MODULES IN ANY QBUS SYSTEM. THIS IS NOT A MEMORY'<12><15>
1560 002504 040 104 111 .ASCII ' DIAGNOSTIC. PLEASE RUN CZKMA OR VMSA DIAGNOSTICS BEFORE RUNNING THIS'<12><15>
1561 002613 040 120 122 .ASCII ' PROGRAM. THE FOLLOWING IS A BRIEF DISCRIPTION OF THE PROGRAM FLOW:'<12><12><15>
1562 002721 040 040 040 .ASCII ' 1. ASK THE OPERATOR A SERIES OF QUESTIONS (MENU)'<12><15>
1563 003010 040 040 040 .ASCII ' 2. RELOCATE PROGRAM TO NON-VOLATILE AREA (IF BANK 0 IS VOLATILE)'<12><15>
1564 003117 040 040 040 .ASCII ' 3. GENERATE AND PRINT A RESTART HELP FILE'<12><15>
1565 003177 040 040 040 .ASCII ' 4. WRITE A BACKGROUND PATTERN (125252) THROUGHOUT MEMORY'<12><15>
1566 003276 040 040 040 .ASCII ' 5. DO A CHECKSUM OF THE PROGRAM'<12><15>
1567 003344 040 040 040 .ASCII ' 6. DO A CHECKSUM OF THE TRAP VECTOR SPACE (ADDRESS 0-376)'<12><15>
1568 003444 040 040 040 .ASCII ' 7. TELL THE OPERATOR TO POWERDOWN THE SYSTEM'<12><12><15>
1569 003530 124 110 105 .ASCII 'THE OPERATOR WILL NOW FOLLOW THE RESTART HELP FILE SOME TIME BFTWEEN'<12><15>
1570 003636 062 040 115 .ASCII '2 MINUTES AND 100 HOURS. THE FOLLOWING IS THE FLOW OF THE PROGRAM'<12><15>
1571 003741 101 106 124 .ASCII 'AFTER POWER UP:'<12><12><15>
1572 003763 040 040 040 .ASCII ' 1. DO ANOTHER CHECKSUM OF THE PROGRAM AND CHECK IT AGAINST THE'<12><15>
1573 004070 040 040 040 .ASCII ' ONE DONE BEFORE POWER DOWN.'<12><15>
1574 004135 040 040 040 .ASCII ' 2. DO ANOTHER CHECKSUM OF THE TRAP VECTOR SPACE AND CHECK IT AGAINST'<12><15>
1575 004250 040 040 040 .ASCII ' THE ONE DONE BEFORE POWER DOWN. (NO TRAP CHECKSUM ERROR WILL BE'<12><15>
1576 004361 040 040 040 .ASCII ' REPORTED UNLESS BANK 0 WAS DETERMINED NON-VOLATILE)'<12><15>
1577 004456 040 040 040 .ASCII ' 3. CHECK ENTIRE MEMORY FOR THE BACKGROUND PATTERN (125252)'<12><15>
1578 004557 040 040 040 .ASCII ' 4. PRINT A MEMORY MAP OF ALL VOLATILE AND NON VOLATILE MEMORY'<12><15>
1579 004663 040 040 040 .ASCII ' IN THE SYSTEM'<12><12><15>
1580 004713 124 110 105 .ASCII 'THE OPERATOR MUST COMPARE THIS MAP WITH THE ONE LEFT BY THE'<12><15>
1581 005010 111 116 123 .ASCII 'INSTALLER IN ORDER TO DETERMINE THAT PROPER DATA RETENTION EXISTS.'<12><12><15>
1582 005115 040 040 040 .ASCII ' PREREQUISITE'<12><12><15>
1583 005157 040 040 040 .ASCII ' 1. CZKMA OR VMSA MUST SUCCESSFULLY COMPLETE'<12><15>
1584 005241 040 040 040 .ASCII ' 2. OPERATOR MUST KNOW STARTING ADDRESSES OF ALL VOLATILE AND'<12><15>
1585 005344 040 040 040 .ASCII ' NON-VOLATILE MEMORY IN THE SYSTEM'<12><15>
1586 005417 040 040 040 .ASCII ' 3. VIDEO TERMINALS MUST EITHER REMAIN POWERED WHEN PROCESSOR IS'<12><15>
1587 005525 040 040 040 .ASCII ' POWERED DOWN OR A COPY OF THE RESTART HELP FILE MUST BE MADE'<12><15>
1588 005633 040 040 040 .ASCII ' BEFORE POWERING DOWN THE SYSTEM'<12><15>
1589 005704 040 040 040 .ASCII ' 4. IF THE CPU DOES NOT HAVE AN ENABLE/HALT SWITCH, CHECK THAT THE'<12><15>
1590 006014 040 040 040 .ASCII ' CPU IS STRAPPED TO HALT WHEN POWERED UP. OTHERWISE THE'<12><15>
1591 006114 040 040 040 .ASCII ' DIAGNOSTIC MAY NOT OPERATE PROPERLY.'<12><12><15>
1592 006174 012 015 104 HELP: .ASCIZ <12><15>'DO YOU WANT A HELP FILE (L) N? '
1593 006236 012 015 101 ILLADD: .ASCIZ <12><15>'ADDRESS NOT AT 4K BOUNDARY!!!!'<12><15>
1594 006302 012 012 012 HELLO: .ASCII <12><12><12><15>'CVMEMAO DATA RETENTION DIAGNOSTIC FOR '
1595 006354 116 117 116 .ASCIZ 'NON-VOLATILE MEMORIES'<12><15>
1596 006404 012 015 111 BANK0: .ASCIZ <12><15>'IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y? '
1597 006470 113 040 117 K: .ASCIZ 'K OF MEMORY'<12><15>
1598 006506 012 015 124 MAPPED: .ASCIZ <12><15>'THIS IS A MAPPED SYSTEM (MEMORY MANAGEMENT) WITH '
1599 006572 012 015 124 UNMAP: .ASCIZ <12><15>'THIS IS AN UNMAPPED SYSTEM (NO MEMORY MANAGEMENT) WITH '
1600 006664 012 015 127 WHERE: .ASCII <12><15>'WHAT IS THE STARTING ADDRESS OF ONE OF THE '
1601 006740 040 116 117 .ASCIZ ' NON-VOLATILE MEMORY'<12><15>'MODULES (O) 0? '
1602 007006 012 015 127 WHERE1: .ASCII <12><15>'WHAT IS THE STARTING ADDRESS OF THE NON-VOLATILE '
1603 007071 115 105 115 .ASCIZ 'MEMORY'<12><15>'UNDER TEST (O) 0? '
1604 007124 012 015 104 WHOLE: .ASCII <12><15>'DO YOU WANT THE DIAGNOSTIC TO VALIDATE ALL MEMORY '
1605 007210 111 116 040 .ASCIZ 'IN THIS SYSTEM (L) Y? '
1606 007237 012 015 110 NOBLK: .ASCIZ <12><15>'HOW MANY 4K BANKS OF MEMORY ARE TO BE TESTED (D) 0? '
1607 007326 012 012 012 RESHLP: .ASCII <12><12><12><12><12><15>
1608 007334 040 040 040 .ASCII ' RESTART HELP FILE'<12><12><15>
1609 007401 123 105 124 .ASCIZ 'SET ENABLE/FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER'<12><15>
1610 007477 074 103 122 CR1: .ASCIZ '<CR>'<12><15>
1611 007506 24 131 120 PAR0: .ASCIZ 'TYPE:'<12><12><15>'@772344/XXXXXX '
1612 007536 100 067 067 PAR7: .ASCII '@772356/XXXXXX 177600<CR>'<12><15>
1613 007571 100 067 067 .ASCII '@772304/XXXXXX 77406<CR>'<12><15>

```

ROUTINE TO SIZE MEMORY

1614	007624	100	067	067
1615	007657	100	067	067
1616	007713	100	067	067
1617	007747	012	123	105
1618	010025	120	117	123
1619	010052	100	044	067
1620	010066	074	103	122
1621	010074	100	120	012
1622	010100	040	040	040
1623	010152	012	012	015
1624	010224	012	012	015
1625	010302	040	101	104

```

.ASCII 'a772316/XXXXXX 77406<CR>'<12><15>
.ASCII 'a777572/XXXXXX 1<CR>'<12><15>
MMR3: .ASCIZ 'a772516/XXXXXX 20<CR>'<12><15>
ENABLE: .ASCII '<12>'SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE '
.ASCII 'POSITION AND TYPE:'<12><12><15>
.ASCII 'a$7/XXXXXX '
.ASCII '<CR>'<12><15>
RHELP: .ASCII 'aP'<12><15>
.ASCII '
.ASCII 'PLEASE STAND BY!'<12><15>
NOT4K: .ASCIZ '<12><12><15>'ILLEGAL ADDRESS - NOT AT 4K BOUNDARY'<12><15>
OUTMEM: .ASCII '<12><12><15>'4K ADDRESS OR NUMBER OF 4K BANKS ARE OUT OF '
.ASCII ' ADDRESS RANGE OF THIS SYSTEM'<12><15>

```

1626				
1627	010342	000000		
1628	010344	000000		
1629	010346	000000		

```

.EVEN
ECHO: .WORD 0
SAVLOW: .WORD 0
SAVHI: .WORD 0

```

1630				
1631		006014		

```

DIF1 = ENDP-RQ4 ;THE NUMBER TO BUMP THE POINTER TO BY PASS THE
;PROGRAM

```

```

;THIS ROUTINE BEGINS THE DIAGNOSTIC, IT WILL DETERMINE IF THIS MACHINE
;HAS MEMORY MANAGEMENT OR NOT, CALL THE SIZING ROUTINE, ASK THE MENU
;QUESTIONS, MOVE THE PROGRAM IF NEED BE, WRITE A BACKGROUND OF 125252 THROUGH
;DUT MEMORY, PRINT RESTART HELP FILE, IF APPLICABLE AND INSTRUCT
;OPERATOR TO POWER DOWN THE SYSTEM.

```

1642	010350			
	010350			

ROUTINE START1

START1:

1643				
1644	010350			
	010350	000005		

```

INLINE <RESET> ;CLEAR THE WORLD
RESET

```

1645				
1646	010352			
	010352	012706	017644	

```

LET SP :- #SPINIT ;FETCH STACK ADDRESS
MOV #SPINIT,SP

```

1647				
1648	010356			
	010356	004767	005646	

```

CALL TRAPCT ;TRAP CATCHER
JSR PC,TRAPCT

```

1649				
1650	010362			
	010362	005067	005572	

```

LET CONTROL :- #0 ;CLEAR OUT CONTROL WORD
CLR CONTROL

```

1651				
1652	010366			
	010366	005067	005576	

```

LET PPOINT : #0 ;PROGRAM POINTER.
CLR PPOINT

```

1653				
1654	010372			
	010372	104401	006302	

```

TYPE ,HELLO ;PRINT WHO WE ARE

```

1655				
1656	010376			
	010376	004767	170644	

```

CALL $SIZE ;THIS ROUTINE DETERMINES, MEMORY MANAGEMENT,
JSR PC,$SIZE ; 16, 18, AND 22 BIT ADDRESS AND MEMORY SIZE

```

1657				
1658				
1659	010402			
	010402	005767	170724	
	010406	002055		

```

IF $KT11 LT #0 THEN ;SET MEMORY MANAGEMENT BIT
TST $KT11
BGE 50002$

```

1660				
1661	010410			

```

LET CONTROL :- $MAP ;THIS SYSTEM HAS

```

1662	010410	016767	171010	005542		MOV	\$MAP,CONTROL
1663	010416				LET	CONTROL := CONTROL SET.BY #1	;INDICATE MEMORY
1664	010416	052767	000001	005534		BIS	#1,CONTROL
1665							;MANAGEMENT
1666	010424				LET	ENDADD := @#KIPAR6 CLR.BY #177	;ONLY WANT 4K BOUNDARY
1667	010424	013767	172354	005542		MOV	@#KIPAR6,ENDADD
1668	010432	042767	000177	005534		BIC	#177,ENDADD
1669							;FETCH LAST PAR ADDRESS IN SYSTEM
1670	010440				LET R1 := #KIPAR0		;FETCH PAR ADDRESS
1671	010440	012701	172340			MOV	#KIPAR0,R1
1672	010444				INCR R0 FROM #0 TO #6 BY #1		;CLEAR OUT PARS
1673	010444	005000				CLR	R0
1674	010446	000401				BR	50003\$
1675	010450						50004\$:
1676	010450	005200				INC	R0
1677	010452						50003\$:
1678	010452	020027	000006			CMP	R0,#6
1679	010456	003002				BGT	50005\$
1680					LET (R1)+ := #0		;4K
1681	010460	005021				CLR	(R1)+
1682	010462				ENDINC		
1683	010462	000772				BR	50004\$
1684	010464						50005\$:
1685	010464	012711	177600		LET (R1) := #177600		;SET UP I/O PAGE IN PAR7
1686	010464					MOV	#177600,(R1)
1687	010470				LET R1 := #KIPDR0		;FETCH PDR ADDRESS
1688	010470	012701	172300			MOV	#KIPDR0,R1
1689	010474				INCR R0 FROM #0 TO #7 BY #1		;SET UP FO 4K INTERVALS
1690	010474	005000				CLR	R0
1691	010476	000401				BR	50006\$
1692	010500						50007\$:
1693	010500	005200				INC	R0
1694	010502						50006\$:
1695	010502	020027	000007			CMP	R0,#7
1696	010506	003003				BGT	50010\$
1697					LET (R1)+ := #77406		;4K
1698	010510	012721	077406			MOV	#77406,(R1)+
1699	010514				ENDINC		
1700	010514	000771				BR	50007\$
1701	010516						50010\$:
1702	010516	012737	000001	177572	LET	@#SRO := #1	;SET MEMORY MANAGEMENT BIT
1703	010516					MOV	#1,@#SRO
1704	010524				IFB	\$MAP LT #0 THEN	;DO WE HAVE A 22 ADDRESS BIT
1705	010524	105767	170674			TSTB	\$MAP


```

010530 002003
1690 ;MACHINE? BGE 50011$
1691
1692 010532 LET @#SR3 := #20 ;SETUP FOR 22 BIT ADDRESSING
010532 012737 000020 172516 MOV #20,@#SR3
1693
1694 010540 ENDIF
010540 50011$:
1695
1696 010540 ELSE
010540 000406 BR 50012$
010542 50002$:
1697
1698 010542 LET ENDADD := $LSTAD + #2 ;FETCH LAST VIRTUAL ADDRESS
010542 016767 171170 005424 MOV $LSTAD,ENDADD
010550 062767 000002 005416 ADD #2,ENDADD
1699
1700 010556 ENDIF
010556 50012$:
1701
1702 010556 IF #1 SETIN CONTROL THEN ;TELL OPERATOR WHAT KIND OF SYSTEM
010556 032767 000001 005374 BIT #1,CONTROL
010564 001403 BEQ 50013$
1703
1704 010566 TYPE ,MAPPED ;TYPE THIS IS A MAPPED SYSTEM
1705 ;(MEMORY MANAGEMENT) WITH
1706
1707 010572 ELSE
010572 000402 BR 50014$
010574 50013$:
1708
1709 010574 TYPE ,UNMAP ;TYPE THIS IS AN UN-MAPPED SYSTEM
1710 ;(NO MEMORY MANAGEMENT) WITH
1711
1712 010600 ENDIF
010600 50014$:
1713
1714 010600 LET $LSTBK := $LSTBK + #40 ;FETCH LAST BANK ADDRESS WITH OUT MEMORY
010600 062767 000040 171132 ADD #40,$LSTBK
1715
1716 010606 LET $LSTBK := $LSTBK SHIFT -1 ;CORRECT LSTBK FOR PRINTING
010606 006267 171126 ASR $LSTBK
1717
1718 010612 LET $LSTBK := $LSTBK CLR.BY #100000 ;CLEAR SIGN BIT
010612 042767 100000 171120 BIC #100000,$LSTBK
1719
1720 010620 LET $LSTBK := $LSTBK SHIFT -4 ;CONTINUE CORRECTING
010620 006267 171114 ASR $LSTBK
010624 006267 171110 ASR $LSTBK
010630 006267 171104 ASR $LSTBK
010634 006267 171100 ASR $LSTBK
1721
1722 010640 TYPDEC $LSTBK ;TELL OPERATOR HOW MUCH MEMORY WE FOUND
010640 016746 171074 MOV $LSTBK,-(SP) ;;SAVE $LSTBK FOR TYPEOUT
010644 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
1723
1724 010646 TYPE ,K ;PRINT K OF MEMORY.

```

```

1725
1726 010652 104401 006174      TYPE      ,HELP      ;PRINT DO YOUR WANT A HELP FILE
1727
1728 010656 004737 001742      CALL      @WFETCH    ;FETCH ANSWER
                                JSR      PC,@WFETCH
1729
1730 010662 023727 010342 000131  IF @WECHO EQ #'Y THEN ;PRINT HELP FILE IF YES
                                CMP      @WECHO, #'Y
                                BNE      50015$
1731
1732 010672 104401 002226      TYPE      ,HELPF     ;PRINT HELP FILE
1733
1734 010676 010676      ENDIF
                                50015$:
1735
1736 010676 104401 006404      TYPE      ,BANKO     ;PRINT IS THERE NON-VOLATILE MEMORY AT ADDRESS 0
1737
1738 010702 004737 001742      CALL      @WFETCH    ;FETCH ANSWER
                                JSR      PC,@WFETCH
1739
1740 010706 023727 010342 000015  IF @WECHO EQ #15 THEN ;LOOK FOR DEFAULT CHARACTER
                                CMP      @WECHO, #15
                                BNE      50016$
1741
1742 010716 012737 000131 010342  LET @WECHO := # Y      ;CHANGE DEFAULT CHARACTER
                                MOV      #'Y,@WECHO
1743
1744 010724 010724      ENDIF
                                50016$:
1745
1746 010724 023727 010342 000131  IF @WECHO NE #'Y THEN ;IF NOT AT BANKO THEN WHERE
                                CMP      @WECHO, #'Y
                                BEQ      50017$
1747
1748 010734 005004      LET R4 := #0        ;CORRECT ADDRESS FLAG
                                CLR      R4
1749
1750 010736 010736      LOOP
                                50020$:
1751
1752 010736 104401 006664      TYPE      ,WHERE     ;PRINT WHAT IS THE STARTING ADDRESS
                                ;OF ONE OF THE NON-VOLATILE MEMORY
                                ;MODULES.
1753
1754
1755
1756 010742 004737 001762      CALL      @WFETADD   ;GO FETCH ADDRESS TO
                                JSR      PC,@WFETADD
1757
1758
1759 010746 005704 001001      EXIF      R4 NE #0   ;RELOCATE PROGRAM
                                TST      R4
                                BNE      50021$
1760
1761 010752 000771      ENDLOOP
                                BR      50020$
1762
1763 010754      IF R3 NE #0 THEN   ;DO NOT HAVE TO RELOCALE IF ADDRESS IS ZERO
  
```

```

010754 005703
010756 001426
1764
1765 010760
010760 032767 000001 005172
010766 001406
1766
1767 010770
010770 010337 172344
1768
1769
1770 010774
010774 012700 040400
1771
1772
1773
1774 011000
011000 010046
1775
1776 011002
011002 000404
011004
1777
1778 011004
011004 010300
011006 062700 000400
1779
1780
1781 011012
011012 010046
1782
1783 011014
011014
1784
1785 011014
011014 012701 011034
1786
1787 011020
011020
1788
1789 011020
011020 020127 017646
011024 001402
1790
1791
1792 011026
011026 012120
1793
1794 011030
011030 000773
011032
1795
1796
1797 011032 000136
1798
1799 011034
011034

TST R3
BEQ 50022$

IF #1 SETIN CONTROL THEN ;WE HAVE MEMORY MANAGEMENT?
BIT #1,CONTROL
BEQ 50023$

LET @#KIPAR2 := R3 ;LOAD PAR2 WITH POINTER TO THE
MOV R3,@#KIPAR2 ;4K CHUNK OF NON-VOLATILE MEMORY

LET R0 := #40400 ;POINT TO PROGRAM ADDRESS REG 2
MOV #40400,R0 ;WE START AT ADDRESS 400 OF
;4K BLOCK

MOV R0,-(SP) PUSH R0 ;:PUSH R0 ON STACK

ELSE ;NO MEMORY MANAGEMENT
BR 50024$
50023$:

LET R0 := R3 + #400 ;WE START AT ADDRESS 400
MOV R3,R0
ADD #400,R0 ;OF 4K BLOCK

MOV R0,-(SP) PUSH R0 ;:PUSH R0 ON STACK

ENDIF

50024$:

LET R1 := #Q1 ;FETCH STARTING ADDRESS TO BE MOVED
MOV #Q1,R1

LOOP

50025$:

EXIF R1 EQ #ENDPR ;LEAVE WHEN ENTIRE PROGRAM
CMP R1,#ENDPR
BEQ 50026$ ;HAS BEEN MOVED.

LET (R0)+ := (R1)+ ;MOVE PROGRAM
MOV (R1)+,(R0)+

ENDLOOP

BR 50025$
50026$:

JMP @ (SP)+ ;GO TO WHERE PROGRAM HAS MOVED TO

ENDIF

50022$:

```

1800				ENDIF	
1801	011034				50017\$:
	011034				
1802					
1803	011034	010706		Q1: LET SP := PC + #SPINIT-.	;FETCH NEW STACK
	011034	062706	006606		MOV PC,SP
	011036				ADD #SPINIT-.,SP
1804					
1805	011042			CALL TRAPCT	;RESET UP TRAP LOCATIONS
	011042	004767	005162		JSR PC,TRAPCT
1806					
1807	011046			LET R1 := @#34 + #STRPAD-\$TRAP	;POINT TO NEW LOCATION OF TRAP TABLE
	011046	013701	000034		MOV @#34,R1
	011052	062701	000046		ADD #STRPAD-\$TRAP,R1
1808					
1809	011056			LET R0 := #STRPAD	;FETCH OLD TABLE ADDRESS TO USE IN
	011056	012700	017474		MOV #STRPAD,R0
1810					;CALCULATING TABLE'S OFFSETS
1811					
1812	011062			LET R3 := R0 - #STRAP2	;CALCULATE OFFSET BETWEEN TABLE AND
	011062	010003			MOV R0,R3
	011064	162703	017462		SUB #STRAP2,R3
1813					;STRAP2 ROUTINE
1814					
1815	011070			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011070	005720			TST (R0)+
1816					
1817	011072			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011072	010102			MOV R1,R2
	011074	160302			SUB R3,R2
1818					
1819	011076			LET (R1)+ := R2	;LOAD TABLE
	011076	010221			MOV R2,(R1)+
1820					
1821	011100			LET R3 := R0 - #STYPE	;CALCULATE OFFSET BETWEEN TABLE AND
	011100	010003			MOV R0,R3
	011102	162703	016370		SUB #STYPE,R3
1822					;STYPE ROUTINE
1823					
1824	011106			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011106	005720			TST (R0)+
1825					
1826	011110			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011110	010102			MOV R1,R2
	011112	160302			SUB R3,R2
1827					
1828	011114			LET (R1)+ := R2	;LOAD TABLE
	011114	010221			MOV R2,(R1)+
1829					
1830	011116			LET R3 := R0 - #STYPOC	;CALCULATE OFFSET BETWEEN TABLE AND
	011116	010003			MOV R0,R3
	011120	162703	017212		SUB #STYPOC,R3
1831					;STYPOC ROUTINE
1832					
1833	011124			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011124	005720			TST (R0)+
1834					

1835	011126			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011126	010102			MOV R1,R2
	011130	160302			SUB R3,R2
1836					
1837	011132			LET (R1)+ := R2	;LOAD TABLE
	011132	010221			MOV R2,(R1)+
1838					
1839	011134			LET R3 := R0 - #STYPOS	;CALCULATE OFFSET BETWEEN TABLE AND
	011134	010003			MOV R0,R3
	011136	162703	017166		SUB #STYPOS,R3
1840					;STYPOS ROUTINE
1841					
1842	011142			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011142	005720			TST (R0)+
1843					
1844	011144			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011144	010102			MOV R1,R2
	011146	160302			SUB R3,R2
1845					
1846	011150			LET (R1)+ := R2	;LOAD TABLE
	011150	010221			MOV R2,(R1)+
1847					
1848	011152			LET R3 := R0 - #STYPON	;CALCULATE OFFSET BETWEEN TABLE AND
	011152	010003			MOV R0,R3
	011154	162703	017226		SUB #STYPON,R3
1849					;STYPON ROUTINE
1850					
1851	011160			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011160	005720			TST (R0)+
1852					
1853	011162			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011162	010102			MOV R1,R2
	011164	160302			SUB R3,R2
1854					
1855	011166			LET (R1)+ := R2	;LOAD TABLE
	011166	010221			MOV R2,(R1)+
1856					
1857	011170			LET R3 := R0 - #STYPDS	;CALCULATE OFFSET BETWEEN TABLE AND
	011170	010003			MOV R0,R3
	011172	162703	016714		SUB #STYPDS,R3
1858					;STYPDS ROUTINE
1859					
1860	011176			INLINE <TST (R0)+>	;BUMP OLD TABLE ADDRESS FOR NEXT CAL.
	011176	005720			TST (R0)+
1861					
1862	011200			LET R2 := R1 - R3	;R2 NOW CONTAINS NEW ADDRESS
	011200	010102			MOV R1,R2
	011202	160302			SUB R3,R2
1863					
1864	011204			LET (R1)+ := R2	;LOAD TABLE
	011204	010221			MOV R2,(R1)+
1865					
1866	011206	104401	007124	TYPE ,WHOLE	;PRINT DO YOU WANT THE DIAGNOSTIC TO VALIDATE
1867					;ALL MEMORY IN THIS SYSTEM.
1868					
1869	011212			CALL @WFETCH	;FETCH CHARACTER.
	011212	004737	001742		JSR PC,@WFETCH

```

1870
1871 011216          IF @#ECHO EQ #15 THEN          ;LOOK FOR DEFAULT CHARACTER
      011216 023727 010342 000015          CMP      @#ECHO,#15
      011224 001003          BNE      50027$
      .372
1873 011226          LET      @#ECHO := #'Y          ;CHANGE DEFAULT CHARACTER
      011226 012737 000131 010342          MOV      #'Y,@#ECHO
1874
1875 011234          ENDIF
      011234
      50027$:
1876
1877 011234          IF @#ECHO NE #'Y THEN          ;IF NOT ENTIRE SYSTEM FIND OUT WHERE
      011234 023727 010342 000131          CMP      @#ECHO,#'Y
      011242 001461          BEQ      50030$
1878
1879 011244          LET      R4 := #0          ;ILLEGAL ADDRESS FLAG
      011244 005004          CLR      R4
1880
1881 011246          LOOP
      011246
      50031$:
1882
1883 011246 104401 007006          TYPE      ,WHERE1 ;PRINT: WHAT IS THE ADDRESS OF THE
1884
1885
1886 011252          CALL      @#FETADD          ;GO FETCH ADDRESS TO BE TESTED
      011252 004737 001762          JSR      PC,@#FETADD
1887
1888 011256          EXIF      R4 NE #0
      011256 005704          TST      R4
      011260 001001          BNE      50032$
1889
1890 011262          ENDLOOP
      011262 000771          BR      50031$
      011264
      50032$:
1891
1892 011264          LET      R4 := #0          ;ILLEGAL ADDRESS FLAG
      011264 005004          CLR      R4
1893
1894 011266          LET      PPOINT := R3          ;NEW PROGRAM POINTER
      011266 010367 004676          MOV      R3,PPOINT
1895
1896 011272          LOOP
      011272
      50033$:
1897
1898 011272 104401 007237          TYPE      ,NOBLK          ;PRINT WHAT IS THE NUMBER OF 4K
1899
1900
1901 011276 104411          RDDEC          ;FETCH DECIMAL NUMBER
1902
1903 011300          POP      R0
      011300 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
1904
1905 011302          LET      R3 := #0          ;CLEAR ADD REG.
      011302 005003          CLR      R3
1906
1907 011304          LET      R2 := #0          ;CLEAR ADD REG.
      011304 005002          CLR      R2
    
```

```

1908
1909 011306                IF R0 NE #0 THEN                ;NO NEED TO CALCULATE END ADD.
      011306 005700                TST      R0
      011310 001426                BEQ      50035$
1910
1911 011312                INCR R1 FROM #1 TO R0 BY #1                ;MULTIPLY THE
      011312 012701 000001                MOV      #1,R1
      011316 000401                BR       50036$
      011320                50037$:
      011320 005201                INC      R1
      011322                50036$:
      011322 020100                CMP      R1,R0
      011324 003004                BGT      50040$
1912
1913
1914 011326                ;NUMBER OF BANKS TO BE TESTED BY 4K
      011326 062702 020000                LET R2 := R2 + #20000                ;ADD 4K
                                          ADD      #20000,R2
1915
1916 011332                LET R3 := R3 + CARRY                ;ADD OVERFLOW TO
      011332 005503                ADC      R3
                                          ;UPPER WORD
1917
1918
1919 011334                ENDINC
      011334 000771                BR       50037$
      011336                50040$:
1920
1921 011336                LET R2 := R2 + @#SAVLOW                ;ADD LOWER START ADDRESS
      011336 063702 010344                ADD      @#SAVLOW,R2
1922
1923 011342                LET R3 := R3 + CARRY                ;CARRY OVER ANY OVERFLOW
      011342 005503                ADC      R3
1924
1925 011344                LET R3 := R3 + @#SAVHI                ;ADD UPPER START ADDRESS BITS
      011344 063703 010346                ADD      @#SAVHI,R3
1926
1927 011350                LET @#ECHO := R2                ;STORE AWAY LOWER
      011350 010237 010342                MOV      R2,@#ECHO
                                          ;ADDRESS
1928
1929
1930 011354                LET @#SHIOCT := R3                ;STORE AWAY HIGHER
      011354 010337 001066                MOV      R3,@#SHIOCT
                                          ;ADDRESS
1931
1932
1933 011360                CALL @#ADDCHK
      011360 004737 002074                JSR      PC,@#ADDCHK
1934
1935 011364                ELSE
      011364 000401                BR       50041$
      011366                50035$:
1936
1937 011366                LET R4 : R4 * #1                ;INDICATE OK
      011366 005204                INC      R4
1938
1939 011370                ENDIF
      011370                50041$:
1940
1941 011370                EXIF R4 NE #0

```

```

ROUTINE TO SIZE MEMORY
011370 005704
011372 001001
1942
1943 011374
011374 000736
011376
1944
1945 011376
011376 005700
011400 001402
1946
1947 011402
011402 010367 004566
1948
1949 011406
011406
1950
1951 011406
011406
1952
1953 011406 104401 007326
1954
1955 011412
011412 032767 000001 004540
011420 001424
1956
1957 011422 104401 007506
1958
1959 011426
011426 013746 172344
011432 104403
011434 006
011435 000
1960
1961 011436 104401 007477
1962
1963 011442 104401 007536
1964
1965
1966
1967
1968 011446
011446 032767 000200 004504
011454 001406
1969
1970
1971 011456
011456 026727 004512 007600
011464 003402
1972
1973
1974 011466 104401 007713
1975
1976 011472
011472
1977
1978 011472

```

```

ENDLOOP
IF R0 NE #0 THEN
    LET ENDADD := R3
ENDIF
ENDIF
TYPE ,RESHLP
IF #1 SETIN CONTROL THEN
    TYPE ,PAR0
    TYPOCS @#KIPAR2 ;MEMORY MANAGEMENT LOCATION OF PROGRAM
    MOV @#KIPAR2,-(SP) ;;SAVE @#KIPAR2 FOR TYPEOUT
    TYPOS ;;GO TYPE--OCTAL ASCII
    .BYTE 6 ;;TYPE 6 DIGITS
    .BYTE 0 ;;SUPPRESS LEADING ZEROS
    TYPE ,CR1 ;PRINT '<CR>'<12><15>
    TYPE ,PAR7 ;PRINT @772356/777600<CR>
    ; @772300/77406<CR>
    ; @772316/77406<CR>
    ; @777572/1<CR>
    IF #200 SETIN CONTROL THEN ;IS THIS A 22 BIT ADDRESSING
        BIT #200,CONTROL
        BEQ 50044$
    ;MACHINE
    IF ENDADD GT #7600 THEN ;DOES THIS MACHINE CONTAIN MORE
        CMP ENDADD,#7600
        BLE 50045$
    ;THAN 124K?
    TYPE ,MMR3 ;YES PRINT @772516/20<CR>
ENDIF
50045$:
ENDIF

```

```

TST R4
BNE 50034$
BR 50033$
50034$:
;UPDATE END ADDRESS IF NECESSARY
TST R0
BEQ 50042$
;STORE AWAY LAST ADDRESS
MOV R3,ENDADD
50042$:
50030$:
;PRINT RESTART HELP FILE HEADER
;ARE WE A MAPPED SYSTEM
BIT #1,CONTROL
BEQ 50043$
;YES
;PRINT @772340/

```



```

ROUTINE TO SIZE MEMORY

011472                                50044$:
1979
1980 011472                            ENDIF                                50043$:
011472
1981
1982 011472 104401 007747              TYPE ,ENABLE ;PRINT SET SWITCH TO ENABLE AND TYPE
1983                                     ;@777707/
1984
1985 .ENABL LSB
1986
1987 011476                            LET R1 := PC + #PWRUP-. ;FETCH RESTART ADDRESS
011476 010701                          MOV PC,R1
011500 062701 000372                  ADD #PWRUP-.,R1
1988
1989 011504                            IF #1 SETIN CONTROL THEN ;ARE WE A MAPPED SYSTEM
011504 032767 000001 004446          BIT #1,CONTROL
011512 001404                          BEQ 50046$
1990                                     ;YES
1991
1992 011514                            LET R1 := R1 CLR.BY #160000 ;CLEAR OUT PROGRAM ADDRESS
011514 042701 160000                  BIC #160000,R1
1993                                     ;REGISTER PRINTERS.
1994
1995 011520                            LET R1 := R1 SET.BY #40000 ;LOOK AT PAR2
011520 052701 040000                  BIS #40000,R1
1996
1997 011524                            ENDIF
011524                                50046$:
1998
1999 011524                            TYPOCS R1 ;PRINT RESTART ADDRESS
011524 010146                          MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
011526 104403                          TYPOS ;;GO TYPE--OCTAL ASCII
011530 006 ;.BYTE 6 ;;TYPE 6 DIGITS
011531 000 ;.BYTE 0 ;;SUPPRESS LEADING ZEROS
2000
2001 011532 104401 010066              TYPE ,RHELP ;PRINT REST OF HELP FILE
2002
2003 .DSABL LSB
2004
2005 ;WRITE BACKGROUND THROUGH MEMORY
2006
2007 011536                            IF #1 SETIN CONTROL THEN ;ARE WE A MAPPED SYSTEM
011536 032767 000001 004414          BIT #1,CONTROL
011544 001414                          BEQ 50047$
2008
2009 011546                            LET @#KIPAR1 := PPOINT ;FETCH TEST STARTING POINT
011546 016737 004416 172342          MOV PPOINT,@#KIPAR1
2010
2011 011554                            IF PPOINT EQ #0 THEN ;ARE WE IN BANK 0
011554 005767 004410                  TST PPOINT
011560 001003                          BNE 50050$
2012
2013 011562                            LET R0 := #20400 ;POINT TO ADDRESS 400, PROGRAM
011562 012700 020400                  MOV #20400,R0
2014                                     ;ADDRESS REGISTER 1
2015
2016 011566                            ELSE

```

```

011566 000402                                BR      50051$
011570                                       50050$:
2017
2018 011570                                LET      RO := #20000 ;POINT TO ADDRESS 0, PROGRAM
011570 012700 020000                                MOV      #20000,RO
;ADDRESS REGISTER 1
2019
2020
2021 011574                                ENDIF
011574                                       50051$:
2022
2023 011574                                ELSE ;UNMAPPED SYSTEM
011574 000410                                       BR      50052$
011576                                       50047$:
2024
2025 011576                                IF PPOINT EQ #0 THEN ;ARE WE IN BANK 0
011576 005767 004366                                TST      PPOINT
011602 001003                                BNE      50053$
2026
2027 011604                                LET      RO := #400 ;START AFTER TRAP LOCATIONS
011604 012700 000400                                MOV      #400,RO
2028
2029 011610                                ELSE
011610 000402                                       BR      50054$
011612                                       50053$:
2030
2031 011612                                LET      RO := PPOINT ;START AT TEST BANK
011612 016700 004352                                MOV      PPOINT,RO
2032
2033 011616                                ENDIF
011616                                       50054$:
2034
2035 011616                                ENDIF
011616                                       50052$:
2036
2037 011616                                LET      R4 := #0 ;CLEAR EXIT FROM LOOP FLAG
011616 005004                                CLR      R4
2038
2039 .ENABL LSB
2040
2041 011620                                LET      R1 := PC + #ROMMD-. ;FETCH ROMMD ROUTINE
011620 010701                                MOV      PC,R1
011622 0052701 004360                                ADD      #ROMMD-.,R1
2042
2043 011626                                LET      @#4 := R1 ;LOAD ADDRESS INTO 4 INCASE WE
011626 010137 000004                                MOV      R1,@#4
;HIT A ROM MODULE
2044
2045
2046 011632                                Q4: LOOP
011632                                       50055$:
2047
2048 011632                                LET      (RO)+ := #125252 ;WRITE DATA PATTERN
011632 012720 125252                                MOV      #125252,(RO)+
;THROUGH OUT MEMORY
2049
2050
2051 011636                                IF #1 SETIN CONTROL THEN ;ARE WE A MAPPED SYSTEM
011636 032767 000001 004314                                BIT      #1,CONTROL
011644 001436                                BEQ      50057$

```

2052					
2053	011646				IF @#KIPAR1 EQ @#KIPAR2 THEN ;WE IN SAME BLOCK OF
	011646	023737	172342	172344	CMP @#KIPAR1,@#KIPAR2
	011654	001014			BNE 50060\$
2054					;MEMORY
2055					
2056	011656				LET R1 := PC - #-04 ;DISTANCE BETWEEN
	011656	010701			MOV PC,R1
	011660	162701	000026		SUB #-04,R1
2057					;BEGINNING OF PROGRAM AND HERE
2058					
2059	011664				LET R1 :- R1 CLR.BY #160000 ;CLEAR OUT PAR
	011664	042701	160000		BIC #160000,R1
2060					;POINTER
2061					
2062	011670				LET R3 := R0 CLR.BY #160000 ;CLEAR OUT PAR
	011670	010003			MOV R0,R3
	011672	042703	160000		BIC #160000,R3
2063					;POINTER
2064					
2065	011676				IF R3 EQ R1 THEN ;HAVE WE REACHED
	011676	020301			CMP R3,R1
	011700	001002			BNE 50061\$
2066					;BEGINNING OF PROGRAM
2067					
2068	011702				LET R0 :- R0 + #DIF1 ;HAS BUMP
	011702	062700	006014		ADD #DIF1,R0
2069					;POINTER TO END OF PROGRAM
2070					
2071	011706				ENDIF
	011706				50061\$:
2072	011706				ENDIF
	011706				50060\$:
2073					
2074	011706				IF R0 EQ #40000 THEN ;HAVE WE COMPLETED 4K CHUNK
	011706	020027	040000		CMP R0,#40000
	011712	001005			BNE 50062\$
2075					;OF MEMORY
2076					
2077	011714				LET R0 := #20000 ;YES RETURN PAR1 POINTER
	011714	012700	020000		MOV #20000,R0
2078					
2079					;BUMP TO NEXT 4K
2080	011720				LET @#KIPAR1 := @#KIPAR1 + #200
	011720	062737	000200	172342	ADD #200,@#KIPAR1
2081					
2082	011726				ENDIF
	011726				50062\$:
2083					
2084	011726				IF @#KIPAR1 EQ ENDADD THEN ;HAVE WE REACHED END OF
	011726	023767	172342	004240	CMP @#KIPAR1,ENDADD
	011734	001001			BNE 50063\$
2085					;MEMORY
2086					
2087	011736				LET R4 := R4 + #1 ;YES SET FINISHED FLAG
	011736	005204			INC R4
2088					

```

ROUTINE TO SIZE MEMORY

2089 011740          ENDIF
      011740          50063$:
2090
2091 011740          ELSE ;UNMAPPED SYSTEM
      011740 000413
      011742          BR      50064$
      011742          50057$:
2092
2093 011742          LET R1 := PC - #-Q4 ;FETCH BEGINNING ADDRESS OF
      011742 010701          MOV      PC,R1
      011744 162701 000112          SUB      #-Q4,R1
      ;THIS PROGRAM
2094
2095
2096 011750          IF R0 EQ R1 THEN ;HAVE WE REACHED BEGINNING OF
      011750 020001          CMP      R0,R1
      011752 001002          BNE      50065$
      ;THIS PROGRAM
2097
2098
2099 011754          LET R0 := R0 + #DIF1 ;YES BUMP POINTER TO
      011754 062700 006014          ADD      #DIF1,R0
      ;END OF PROGRAM
2100
2101
2102 011760          ENDIF
      011760          50065$:
2103
2104 011760          IF R0 HIS ENDADD THEN ;HAVE WE REACHED END OF MEMORY
      011760 020067 004210          CMP      R0,ENDADD
      011764 103401          BLO      50066$
2105
2106 011766          LET      R4 := R4 + #1 ;YES SET FINISHED FLAG
      011766 005204          INC      R4
2107
2108 011770          ENDIF
      011770          50066$:
2109
2110          ENDIF
      011770          50064$:
2111
2112 011770          EXIF   R4 NE #0 ;LEAVE IF FINISHED FLAG IS SET
      011770 005704          TST      R4
      011772 001001          BNE      50056$
2113
2114 011774          ENDLLOOP
      011774 000716          BR      50055$
      011776          50056$:
2115
2116          ;FETCH LOCATION OF POWER DOWN MESSAGE
2117 011776          LET R1 := PC + #PWRDWN-.
      011776 010701          MOV      PC,R1
      012000 062701 003776          ADD      #PWRDWN-. ,R1
2118
2119          ;LOAD ADDRESS INTO TRAP INSTRUCTION
2120 012004          LET 1$ := R1
      012004 010167 000054          MOV      R1,1$
2121
2122          ;CALCULATE CHECKSUM OF TRAP PAGE
2123
2124 012010          LET R1 :- #0 ;ADDRESS OF TRAP LOCATION

```

2125	012010	005001			CLR R1
2126	012012		LET R3 := #0		;CHECKSUM WORK REGISTER
	012012	005003			CLR R3
2127					
2128	012014		LOOP		50067\$:
	012014				
2129					
2130	012014		LET R3 := R3 + (R1)+		;CALCULATE CHECKSUM
	012014	062103			ADD (R1)+,R3
2131					
2132	012016		EXIF R1 EQ #400		;EXIF WHEN REACHED END OF TRAP PAGE
	012016	020127			CMP R1,#400
	012022	001401			BEQ 50070\$
2133					
2134	012024		ENDLOOP		BR 50067\$
	012024	000773			50070\$:
	012026				
2135					
2136	012026		LET CSTRPG := R3		;SAVE CHECKSUM
	012026	010367			MOV R3,CSTRPG
2137					
2138					
2139	012032		;CALCULATE CHECKSUM OF THIS PROGRAM		;R1 CONTAINS BEGINNING ADDRESS
	012032	010701	LET R1 := PC - #-Q4		MOV PC,R1
	012034	162701			SUB #-Q4,R1
2140					;OF PROGRAM
2141					
2142	012040		LET R2 := PC + #CHKSUM-.		;R2 CONTAINS END ADDRESS OF PROGRAM
	012040	010702			MOV PC,R2
	012042	062702			ADD #CHKSUM-.,R2
2143					
2144	012046		LET R3 := #0		;R3 CONTAINS CHECK SUM
	012046	005003			CLR R3
2145					
2146	012050		LOOP		50071\$:
	012050				
2147					
2148	012050		LET R3 := R3 + (R1)+		;ADD CONTENTS OF PROGRAM
	012050	062103			ADD (R1)+,R3
2149					
2150	012052		EXIF R1 EQ R2		;LEAVE WHEN REACHED END OF PROGRAM
	012052	020102			CMP R1,R2
	012054	001401			BEQ 50072\$
2151					
2152	012056		ENDLOOP		BR 50071\$
	012056	000774			50072\$:
	012060				
2153					
2154	012060		LET (R2) := R3		;LOAD CHECK SUM INTO CHECKSUM AREA
	012060	010312			MOV R3,(R2)
2155					
2156					
2157	012062	104401	;TELL OPERATOR TO POWER DOWN THE SYSTEM		
2158	012064	015776	TYPE		
2159			.WORD PWRDWN		
2160	012066		INLINE < BR . >		;

1\$:

2161 012066 000777
2162
2163
2164 012070
012070
012070
012070 000207

.DSABL LSB
ENDRTN

BR .

50000\$:
50001\$:
RTS PC

```

                .SBTTL POWER UP ROUTINE
2166
2167
2168                ;THIS ROUTINE CHECKS THE ENTIRE SYSTEM FOR DATA RETENTION AND OUTPUTS A MEMORY
2169                ;MAP OF THE SYSTEM.
2170
2171                ROUTINE PWRUP
                PWRUP:
2172
2173                LET SP := PC + #SPINIT-.                ;FETCH STACK BUFFER
                MOV PC,SP
                ADD #SPINIT-.,SP
2174
2175                IF #1 SETIN CONTROL THEN                ;IS THIS A MAPPED SYSTEM
                BIT #1,CONTROL
                BEQ 50002$
2176                ;YES
2177
2178                LET @#KIPARO := #0                ;CLEAR OUT PAR0
                CLR @#KIPARO
2179
2180                LET @#KIPAR1 := #0                ;CLEAR OUT PAR1
                CLR @#KIPAR1
2181
2182                LET R1 := #KIPAR3                ;FETCH MEMORY MANAGEMENT PAR
                MOV #KIPAR3,R1
                ;LOCATION (DO NOT TOUCH PAR2 FOR
                ;IT CONTROLS OUR PC
2183
2184
2185
2186                INCR R0 FROM #1 TO #4 BY #1                ;CLEAR THOSE REGISTERS
                MOV #1,R0
                BR 50003$
                50004$:
                INC R0
                50003$:
                CMP R0,#4
                BGT 50005$
2187
2188                LET (R1)+ := #0                ;
                CLR (R1)+
2189
2190                ENDINC
                BR 50004$
                50005$:
2191
2192                LET (R1) := #177600                ;SET UP I/O PAGE
                MOV #177600,(R1)
2193
2194                LET R1 := #KIPDR0                ;FETCH PDR LOCATION
                MOV #KIPDR0,R1
2195
2196                INCR R0 FROM #0 TO #7 BY #1                ;SET THEM UP TO DO 4K INTERVALS
                CLR R0
                BR 50006$
                50007$:
                INC R0
                50006$:
                CMP R0,#7
    
```

```

POWER UP ROUTINE

2197 012170 003003                                BGT      50010$
2198 012172                                LET (R1)+ := #77406 ;
012172 012721 077406                                MOV      #77406,(R1)+
2199
2200 012176                                ENDINC
012176 000771                                BR       50007$
012200                                50010$:
2201
2202 012200                                LET @#SRO := #1 ;SET MEMORY MANAGEMENT IF NOT
012200 012737 000001 177572                                MOV      #1,@#SRO
;ALREADY DONE SO
2203
2204
2205 012206                                IF #200 SETIN CONTROL THEN ;IS THIS A 22 ADDRESS BIT SYSTEM
012206 032767 000200 003744                                BIT      #200,CONTROL
012214 001403                                BEQ      50011$
2206
2207 012216                                LET @#SR3 := #20 ;YES, SET UP 22 BIT ADDRESSING
012216 012737 000020 172516                                MOV      #20,@#SR3
2208
2209 012224                                ENDIF
012224                                50011$:
2210
2211 012224                                IF PPOINT EQ #0 THEN ;ARE WE CHECKING ENTIRE SYSTEM?
012224 005767 003740                                TST     PPOINT
012230 001003                                BNE     50012$
;YES
2212
2213
2214 012232                                LET R0 := #20400 ;DO NOT CHECK TRAP LOCATIONS
012232 012700 020400                                MOV     #20400,R0
2215
2216 012236                                ELSE ;NO
012236 000405                                BR      50013$
012240                                50012$:
2217
2218 012240                                LET @#KIPAR1 := PPOINT ;SET UP POINTER TO CHECK THAT
012240 016737 003724 172342                                MOV     PPOINT,@#KIPAR1
;MODULE
2219
2220
2221 012246                                LET R0 := #20000 ;POINT TO PAR1 ADDRESS ZERO
012246 012700 020000                                MOV     #20000,R0
2222
2223 012252                                ENDIF
012252                                50013$:
2224
2225 012252                                ELSE
012252 000410                                BR      50014$
012254                                50002$:
2226
2227 012254                                IF PPOINT EQ #0 THEN ;ARE WE CHECKING ENTIRE SYSTEM?
012254 005767 003710                                TST     PPOINT
012260 001003                                BNE     50015$
;YES
2228
2229
2230 012262                                LET R0 := #400 ;SKIP TRAP LOCATIONS
012262 012700 000400                                MOV     #400,R0
2231

```



```

2232 012266          ELSE
      012266 000402          BR      50016$
      012270          50017$:
2233
2234 012270          LET R0 := PPCINT          ;FETCH POINTER ADDRESS
      012270 016700 003674          MOV      PPOINT,R0
2235
2236 012274          ENDIF
      012274          50016$:
2237
2238 012274          ENDIF
      012274          50014$:
2239
2240          ;CALCULATE CHECKSUM OF TRAP PAGE
2241
2242 012274          LET R1 := #0          ;ADDRESS OF TRAP LOCATION
      012274 005001          CLR      R1
2243
2244 012276          LET R3 := #0          ;CHECKSUM WORK REGISTER
      012276 005003          CLR      R3
2245
2246 012300          LOOP
      012300          50017$:
2247
2248 012300          LET R3 := R3 + (R1)+          ;CALCULATE CHECKSUM
      012300 062103          ADD      (R1)+,R3
2249
2250 012302          EXIF R1 EQ #400          ;EXIF WHEN REACHED END OF TRAP PAGE
      012302 020127 000400          CMP      R1,#400
      012306 001401          BEQ      50020$
2251
2252 012310          ENDLLOOP
      012310 000173          BR      50017$
      012312          50020$:
2253
2254 012312          IF R3 NE CSTRPG THEN          ;DO CHECKSUMS AGREE?
      012312 020367 005202          CMP      R3,CSTRPG
      012316 001403          BEQ      50021$
2255
2256          ;NO
2257 012320          MOV      PUSH #4000          ;INDICATE CHECKSUM ERROR
      012320 012746 004000          #4000,-(SP)  ;;PUSH #4000 ON STACK
2258
2259 012324          ELSE
      012324 000402          BR      50022$
      012326          50021$:
2260
2261 012326          MOV      PUSH #0          ;INDICATE NO ERROR
      012326 012746 000000          #0,-(SP)  ;;PUSH #0 ON STACK
2262
2263          ;CHECKSUM ERROR
2264 012332          ENDIF
      012332          50022$:
2265
2266 012332          CALL   TRAPCT          ;LOAD UP TRAP VECTORS WITH
      012332 004767 003672          JSR      PC,TRAPCT
2267          ;TRAP CATCHERS
    
```

```

2268
2269 012336          LET    R1 := PC - #-Q4          ;R1 CONTAINS BEGINNING ADDRESS
      012336 010701          MOV    PC,R1
      012340 162701 000506          SUB    #-Q4,R1
2270
2271
2272 012344          LET    R2 := PC + #CHKSUM-.      ;R2 CONTAINS END ADDRESS OF
      012344 010702          MOV    PC,R2
      012346 062702 005154          ADD    #CHKSUM-. ,R2
2273
2274
2275 012352          LET    R3 := #0                ;R3 CONTAINS CHECK SUM
      012352 005003          CLR    R3
2276
2277 012354          LOOP
      012354
2278
2279 012354          LET    R3 := R3 + (R1)+          ;ADD CONTENTS OF PROGRAM
      012354 062103          ADD    (R1)+,R3
2280
2281 012356          EXIF   R1 EQ R2                ;LEAVE WHEN REACHED END OF PROGRAM
      012356 020102          CMP    R1,R2
      012360 001401          BEQ    50024$
2282
2283 012362          ENDL0OP
      012362 000774          BR    50023$
      012364
2284
2285 012364          IF (R2) NE R3 THEN            ;DO WE HAVE A CHECKSUM ERROR
      012364 021203          CMP    (R2),R3
      012366 001407          BEQ    50025$
2286
2287
2288
2289 012370          ;FETCH LOCATION OF ERROR MESSAGE
      012370 010701          LET R1 := PC + #ERR-.
      012372 062701 002641          MOV    PC,R1
2290
2291
2292 012376          ;LOAD ADDRESS INTO TRAP INSTRUCTION
      012376 010167 000002          LET 2$ := R1
2293
2294
2295 012402          ;TELL OPERATOR CHECK SUM ERROR
      012402 104401          TYPE
2296 012404          2$: .WORD ERR
      012404 015233
2297
2298 012406          ENDF
      012406
2299
2300 012406          LET CONTROL := CONTROL SET.BY (SP)+ ;INDICATE RESULT OF CHECK SUM
      012406 052667 003546          BIS    (SP)+,CONTROL
2301
2302
2303 012412          PUSH   PC
      012412 010746          MOV    PC,-(SP)      ;;PUSH PC ON STACK
2304
2305 012414          LET    (SP) := (SP) + #HEAD-.    ;CALCULATE HEAD LOACTION
    
```

```

012414 062716 003112                                ADD    #HEAD-.,(SP)
2306
2307 012420                                POP    30$                                ;TO PRINT IT
012420 012667 000002                                MOV    (SP)+,30$                        ;;POP STACK INTO 30$
2308
2309 012424 104401                                TYPE
2310
2311 012426 015526                                30$: .WORD HEAD                        ;PRINT MEMORY MAP HEADER
2312
2313 012430                                LET START := PPOINT                      ;SAVE STARTING POINT
012430 016767 003534 003540                                MOV    PPOINT,START
2314
2315 012436                                LET L4KST := PPOINT                      ;WE ARE AT THIS 4K BANK
012436 016767 003526 003534                                MOV    PPOINT,L4KST
2316
2317 012444                                LET R3 := #172100                        ;FETCH CSR LOCATIONS
012444 012703 172100                                MOV    #172100,R3
2318
2319 012450                                LET TRCATCH := #240                      ;NOP TRAP CATCHER
012450 012767 000240 003706                                MOV    #240,TRCATCH
2320
2321 012456                                INCR R2 FROM #0 TO #16. BY #1           ;ENABLE PARITY FOR ALL CSRS
012456 005002                                CLR    R2
012460 000401                                BR     50026$
012462                                50027$:
012462 005202                                INC    R2
012464                                50026$:
012464 020227 000020                                CMP    R2,#16.
012470 003003                                BGT   50030$
2322
2323 012472                                LET (R3)+ := #1                          ;
012472 012723 000001                                MOV    #1,(R3)+
2324
2325 012476                                ENDINC
012476 000771                                BR     50027$
012500                                50030$:
2326
2327 012500                                LET TRCATCH := #0                        ;RESUME HALT IF TRAP
012500 005067 003660                                CLR    TRCATCH
2328
2329 012504                                LET R4 := #114                          ;LOAD PARITY LOCATION INTO R4
012504 012704 000114                                MOV    #114,R4
2330                                ;SO THAT NEXT INSTRUCTION WILL
2331                                ;WORK ON ALL MACHINES
2332
2333 012510                                LET (R4) := PC + #PARTRP-.              ;LOAD PARITY TRAP LOCATION WITH
012510 010714                                MOV    PC,(R4)
012512 062714 002140                                ADD    #PARTRP-.,(R4)
2334                                ;ADDRESS OF PARITY TRAP ROUTINE
2335
2336 012516                                IF #125252 EQ (R0)+ THEN                ;IS THIS LOCATION VOLATILE
012516 022720 125252                                CMP    #125252,(R0)+
012522 001016                                BNE   50031$
2337
2338                                ;IF PARITY ERROR THEN ERROR COUNT ALREADY DONE
2339 012524                                IF #2 SETIN CONTROL THEN
012524 032767 000002 003426                                BIT    #2,CONTROL

```

```

POWER UP ROUTINE

2340 012532 001404                                BEQ      50032$
2341 012534                                LET CONTROL := CONTROL CLR.BY #2      ;CLEAR PARITY
012534 042767 000002 003416                    BIC      #2,CONTROL                    ;ERROR FLAG
2342
2343
2344 012542                                ELSE
012542 000405                                BR       50033$
012544
2345
2346                                ;INDICATE NON-VOLATILE
2347 012544                                LET CONTROL := CONTROL SET.BY #100000
012544 052767 100000 003406                    BIS      #100000,CONTROL
2348
2349 012552                                LET NONVOL := NONVOL + #1             ;UPDATE NONVOL COUNT
012552 005267 003404                                INC      NONVOL
2350
2351 012556                                ENDIF
012556
2352
2353                                ELSE
012556 000412                                BR       50034$
012556 000412                                50031$:
012560
2354
2355                                ;IF PARITY ERROR THEN ERROR COUNT ALREADY DONE
2356 012560                                IF #2 SETIN CONTROL THEN
012560 032767 000002 003372                    BIT      #2,CONTROL
012566 001404                                BEQ      50035$
2357
2358 012570                                LET CONTROL := CONTROL CLR.BY #2      ;CLEAR PARITY
012570 042767 000002 003362                    BIC      #2,CONTROL                    ;ERROR FLAG
2359
2360
2361 012576                                ELSE
012576 000402                                BR       50036$
012600                                50035$:
2362
2363 012600                                LET VOL := VOL + #1                   ;UPDATE VOL COUNT
012600 005267 003360                                INC      VOL
2364
2365                                ENDIF
012604
2366                                ENDIF
012604
2367                                50034$:
012604
2368
2369 012604                                LET R4 := #0                          ;FLAG FOR NON VOLATILE MEMORY HITS
012604 005004                                CLR      R4
2370
2371                                ; FAKED OUT LOOP BECAUSE LOOP IS OUT SIDE OF ADDRESS RANGE
2372
2373 012606                                STLOOP: IF #1 SETIN CONTROL THEN      ;ARE WE MAPPED
012606 032767 000001 003344                    BIT      #1,CONTROL
012614 001407                                BEQ      50037$
2374
2375                                ;YES
2376 012616                                IF #KIPAR1 EQ ENDADD THEN            ;IS IT TIME TO LEAVE?

```

```

012616 023767 172342 003350          CMP      @#KIPAR1,ENDADD
012624 001002                          BNE      50040$
2377                                     ;YES
2378 012626          INLINE <JMP      EDLOOP> ;LEAVE LOOP
012626 000167 001356                    JMP      EDLOOP
2379
2380 012632          ENDIF
012632                          50040$:
2381
2382 012632          ELSE
012632 000405                          BR      50041$
012634                          50037$:
2383
2384 012634          IF RO EQ ENDADD THEN ;IS IT TIME TO LEAVE?
012634 020067 003334                    CMP      RO,ENDADD
012640 001002                          BNE      50042$
2385                                     ;YES
2386
2387 012642          INLINE <JMP      EDLOOP> ;LEAVE LOOP
012642 000167 001342                    JMP      EDLOOP
2388
2389 012646          ENDIF
012646                          50042$:
2390
2391 012646          ENDIF
012646                          50041$:
2392
2393 012646          IF #125252 EQ (RO)+ THEN ;DOES THIS LOCATION CONTAIN
012646 022720 125252                    CMP      #125252,(RO)+
012652 001013                          BNE      50043$
2394                                     ;CORRECT DATA
2395
2396                                     ;IF PARITY ERROR THEN ERROR COUNT ALREADY DONE
2397 012654          IF #2 SETIN CONTROL THEN
012654 032767 000002 003276            BIT      #2,CONTROL
012662 001404                          BEQ      50044$
2398
2399                                     ;CLEAR PARITY ERROR FLAG
2400 012664          LET CONTROL := CONTROL CLR.BY #2
012664 042767 000002 003266            BIC      #2,CONTROL
2401
2402 012672          ELSE
012672 000402                          BR      50045$
012674                          50044$:
2403
2404 012674          LET NONVOL := NONVOL + #1 ;YES, UPDATE
012674 005267 003262                    INC      NONVOL
2405                                     ;NON-VOLATILE COUNT
2406
2407 012700          ENDIF
012700                          50045$:
2408
2409 012700          ELSE
012700 000412                          BR      50046$
012702                          50043$:
2410
2411                                     ;IF PARITY ERROR THEN ERROR COUNT ALREADY DONE

```

```

2412 012702                IF #2 SETIN CONTROL THEN
      012702 032767 000002 003250                BIT      #2,CONTROL
      012710 001404                                BEQ      50047$
2413
2414                                ;CLEAR PARITY ERROR FLAG
2415 012712                LET CONTROL := CONTROL CLR.BY #2
      012712 042767 000002 003240                BIC      #2,CONTROL
2416
2417 012720                ELSE
      012720 000402                                BR       50050$
      012722                                50047$:
2418
2419 012722                LET VOL := VOL + #1      ;NO, UPDATE
      012722 005267 003236                INC      VOL
2420                                ;VOLATILE COUNT
2421
2422 012726                ENDIF
      012726                                50050$:
2423
2424 012726                ENDIF
      012726                                50046$:
2425
2426 012726                IF #1 SETIN CONTROL THEN      ;ARE WE MAPPED
      012726 032767 000001 003224                BIT      #1,CONTROL
      012734 001421                                BEQ      50051$
2427
2428 012736                IF @#KIPAR1 EQ @#KIPAR2 THEN      ;ARE PARS IN SAME BANK?
      012736 023737 172342 172344                CMP      @#KIPAR1,@#KIPAR2
      012744 001014                BNE      50052$
2429                                ;YES
2430
2431                                ;DISTANCE BETWEEN BEGINNING OF PROGRAM AND HERE
2432 012746                LET R1 := PC - #-Q4
      012746 010701                                MOV      PC,R1
      012750 162701 001116                SUB      #-Q4,R1
2433
2434                                ;CLEAR OUT PAR POINTER
2435 012754                LET R1 := R1 CLR.BY #160000
      012754 042701 160000                BIC      #160000,R1
2436
2437                                ;CLEAR OUT PAR POINTER
2438 012760                LET R3 := R0 CLR.BY #160000
      012760 010003                                MOV      R0,R3
      012762 042703 160000                BIC      #160000,R3
2439
2440                                ;HAVE WE REACHED BEGINNING OF PROGRAM
2441 012766                IF R3 EQ R1 THEN
      012766 020301                                CMP      R3,R1
      012770 001002                BNE      50053$
2442
2443                                ;BUMP POINTER TO END OF PROGRAM
2444 012772                LET R0 := R0 + #DIF1
      012772 062700 006014                ADD      #DIF1,R0
2445
2446 012776                ENDIF
      012776                                50053$:
2447

```

```

2448 012776          ENDIF
      012776          50052$:
2449
2450 012776          ELSE
      012776 000407          BR      50054$
      013000          50051$:
2451
2452          ;ADDRESS OF BEGINING OF PROGRAM
2453 013000          LET R1 := PC - #-Q4
      013000 010701          MOV     PC,R1
      013002 162701 001150          SUB     #-Q4,R1
2454
2455 013006          IF R0 EQ R1 THEN          ;POINTER REACHED PROGRAM?
      013006 020001          CMP     R0,R1
      013010 001002          BNE     50055$
2456
2457          ;BUMP POINTER TO END OF PROGRAM
2458 013012          LET R0 := R0 + #DIF1
      013012 062700 006014          ADD     #DIF1,R0
2459
2460 013016          ENDIF
      013016          50055$:
2461
2462 013016          ENDIF
      013016          50054$:
2463
2464 013016          IF #17777 SETIN R0 THEN          ;ARE WE AT 4K BOUNDARY?
      013016 032700 017777          BIT     #17777,R0
      013022 001402          BEQ     50056$
2465
2466          ;NO
2467 013024          INLINE <JMP STLOOP>          ;CONTINUE LOOP & BYPASS PRINT
      013024 000167 177556          JMP STLOOP
2468
2469          ;ROUTINES
2470 013030          ENDIF
      013030          50056$:
2471
2472          ;ENABLE PARITY
2473 013030          LET R3 := #172100          ;FETCH CSR LOCATIONS
      013030 012703 172100          MOV     #172100,R3
2474
2475 013034          LET TRCATCH := #240          ;NOP TRAP CATCHER
      013034 012767 000240 003322          MOV     #240,TRCATCH
2476
2477 013042          INCR R2 FROM #0 TO #16. BY #1          ;ENABLE PARITY FOR ALL CSRS
      013042 005002          CLR     R2
      013044 000401          BR      50057$
      013046          50060$:
      013046 005202          INC     R2
      013050          50057$:
      013050 020227 000020          CMP     R2,#16.
      013054 003003          BGT     50061$
2478
2479 013056          LET (R3)+ := #1
      013056 012723 000001          ;
      MOV     #1,(R3)+
2480
    
```

```

2481 013062          ENDINC
      013062 000771          BR      50060$
      013064          50061$:
2482 013064          LET TRCATCH := #0          ;RESUME HALT IF TRAP
2483 013064 005067 003274          CLR      TRCATCH
2484 013070          LET R4 := #0          ;CLEAR PRINT FLAG
2485 013070 005004          CLR      R4
2486 013072          IF PARCNT NE #0 THEN ;WE FOUND SOME PARITY ERRORS
2487 013072 005767 003074          TST     PARCNT
      013076 001405          BEQ     50062$
2488 013100          IF VOL LOS NONVOL THEN ;MAKE SURE IT IS
2489 013100 026767 003060 003054          CMP     VOL, NONVOL
      013106 101001          BHI     50063$
      ;STILL NON VOLATILE
2490 013110          LET R4 := R4 + #1          ;HITS
2491 013110 005204          INC     R4
2492 013112          ENDIF
2493 013112          50063$:
2494 013112          ENDIF
2495 013112          50062$:
2496 013112          ENDIF
2497 013112          IF VOL NE #0 THEN ;HAVE FOUND ANY HITS?
2498 013112 005767 003046          TST     VOL
      013116 001405          BEQ     50064$
      ;YES
2499 013120          IF VOL LE NONVOL THEN ;MIGHT BE VOL
2500 013120 026767 003040 003034          CMP     VOL, NONVOL
      013126 003001          BGT     50065$
      ;NO
2501 013130          LET R4 := R4 + #1          ;HITS
2502 013130 005204          INC     R4
2503 013132          ENDIF
2504 013132          50065$:
2505 013132          ENDIF
2506 013132          50064$:
2507 013132          IF R4 EQ #0 THEN ;WE HAVE HITS IN NON VOLATILE
2508 013132 005704          TST     R4
      013134 001002          BNE     50066$
      ;MEMORY
2509 013136          INLINE <JMP 20$> ;GO TO NORMAL ROUTINE (NO ERRORS)
2510 013136 000167 000450          JMP     20$
2511 013142          ENDIF
2512 013142          50066$:
2513 013142
2514 013142
2515 013142
    
```


2516						
2517	013142					IF L4KST EQ START THEN ;ONLY NEED TO PRINT ONE LINE
	013142	026767	003032	003026		CMP L4KST,START
	013150	001066				BNE 50067\$
2518						
2519						;TAKE CARE OF ANY PROBLEMS WITH START ADDRESS
2520						;BEING VOLATILE WITHIN NON-VOLATILE MEMORY
2521	013152					LET CONTROL := CONTROL SET.BY #100000
	013152	052767	100000	003000		BIS #100000,CONTROL
2522						
2523	013160					IF #1 SET IN CONTROL THEN ;MAPPED
	013160	032767	000001	002772		BIT #1,CONTROL
	013166	001404				BEQ 50070\$
2524						
2525						;FETCH LAST BANK ADDRESS
2526	013170					LET END := @#KIPAR1
	013170	013767	172342	002770		MOV @#KIPAR1,END
2527						
2528	013176					ELSE
	013176	000405				BR 50071\$
	013200					50070\$:
2529						
2530						;FETCH LAST BANK ADDRESS
2531	013200					LET END := R0 - #2
	013200	010067	002762			MOV R0,END
	013204	162767	000002	002754		SUB #2,END
2532						
2533	013212					ENDIF
	013212					50071\$:
2534						
2535	013212					CALL PRINT ;PRINT ADDRESSES
	013212	004767	001172			JSR PC,PRINT
2536						
2537	013216					CALL TAB ;PRINT TAB
	013216	004767	001142			JSR PC,TAB
2538						
2539	013222					PUSH R1
	013222	010146				;;PUSH R1 ON STACK
2540						
2541	013224					LET R1 := VOL ;FETCH VOL COUNT
	013224	016701	002734			MOV VOL,R1
2542						
2543	013230					TYPDEC R1 ;PRINT ERROR COUNT
	013230	010146				;;SAVE R1 FOR TYPEOUT
	013232	104405				;;GO TYPE--DECIMAL ASCII WITH SIGN
2544						
2545	013234					CALL TAB ;PRINT TAB
	013234	004767	001124			JSR PC,TAB
2546						
2547	013240					LET R1 := PARCNT ;FETCH PARITY COUNT
	013240	016701	002726			MOV PARCNT,R1
2548						
2549	013244					TYPDEC R1 ;PRINT PARITY ERROR CNT.
	013244	010146				;;SAVE R1 FOR TYPEOUT
	013246	104405				;;GO TYPE--DECIMAL ASCII WITH SIGN
2550						
2551	013250					IF PARCNT EQ #10. THEN ;DO WE HAVE 10 OR MORE

```

013250 026727 002716 000012          CMP      PARCNT,#10.
013256 001002          BNE      50072$
2552          ;PARITY ERRORS
2553
2554 013260 104401 015100          TYPE    ,MORE    ;PRINT OR MORE
2555
2556 013264          ENDIF
013264          50072$:
2557
2558 013264          POP      R1      ;RESTORE R1
013264 012601          MOV      (SP)+,R1    ;:POP STACK INTO R1
2559
2560 013266          CALL     CRLF2      ;PRINT <CR> <LF>
013266 004767 001046          JSR      PC,CRLF2
2561
2562          ;UP DATE START BANK TO NEXT LEVEL TO BE TESTED
2563 013272          IF #1 SETIN CONTROL THEN ;MAPPED
013272 032767 000001 002660          BIT      #1,CONTROL
013300 001407          BEQ      50073$
2564
2565 013302          LET START := @WKIPAR1 + #200
013302 013767 172342 002666          MOV      @WKIPAR1,START
013310 062767 000200 002660          ADD      #200,START
2566
2567 013316          ELSE
013316 000402          BR      50074$
013320          50073$:
2568
2569 013320          LET START := R0
013320 010067 002652          MOV      R0,START
2570
2571 013324          ENDIF
013324          50074$:
2572
2573 013324          ELSE
013324 000532          BR      50075$
013326          50067$:
2574
2575 013326          IF #1 SETIN CONTROL THEN ;MAPPED
013326 032767 000001 002624          BIT      #1,CONTROL
013334 001407          BEQ      50076$
2576
2577 013336          LET END := L4KST - #200 ;SUB 4K
013336 016767 002636 002622          MOV      L4KST,END
013344 162767 000200 002614          SUB      #200,END
2578
2579 013352          ELSE
013352 000406          BR      50077$
013354          50076$:
2580
2581 013354          LET END := L4KST - #2 ;LAST ADDRESS
013354 016767 002620 002604          MOV      L4KST,END
013362 162767 000002 002576          SUB      #2,END
2582
2583 013370          ENDIF
013370          50077$:
2584
    
```

POWER UP ROUTINE

```

2585 013370
      013370 004767 001014
2586
2587 013374
      013374 005767 002560
      013400 002012
2588
2589 013402
      013402 004767 000756
2590
2591
2592 013406
      013406 012746 000000
      013412 104405
2593
2594 013414
      013414 004767 000744
2595
2596 013420
      013420 012746 000000
      013424 104405
2597
2598 013426
      013426
2599
2600
2601 013426
      013426 052767 100000 002524
2602
2603 013434
      013434 004767 000700
2604
2605 013440
      013440 032767 000001 002512
      013446 001404
2606
2607
2608 013450
      013450 013767 172342 002510
2609
2610 013456
      013456 000405
      013460
2611
2612
2613 013460
      013460 010067 002502
      013464 162767 000002 002474
2614
2615 013472
      013472
2616
2617 013472
      013472 016767 002502 002476
2618
2619 013500
      013500 004767 000704
    
```

```

MOV #0,-(SP)
TYPDS
MOV #0,-(SP)
TYPDS
    
```

```

CALL PRINT ;PRINT ADDRESSES
      JSR PC,PRINT
IF CONTROL LT #0 THEN ;ARE WE CHECKING NON-VOL
      TST CONTROL
      BGE 50100$
CALL TAB ;PRINT TAB
      JSR PC,TAB
;NO ERRORS IN THIS PART
TYPDEC #0
;;SAVE #0 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN
CALL TAB ;PRINT TAB
      JSR PC,TAB
TYPDEC #0 ;NO PARITY EITHER
;;SAVE #0 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN
ENDIF
50100$:
;INDICATE THAT NOW THIS IS NON-VOLATILE
LET CONTROL := CONTROL SET BY #100000
      BIS #100000,CONTROL
CALL CRLF2 ;PRINT <CR><LF>
      JSR PC,CRLF2
IF #1 SET IN CONTROL THEN ;MAPPED
      BIT #1,CONTROL
      BEQ 50101$
;LAST 4K BANK
LET END := @#KIPAR1
      MOV @#KIPAR1,END
ELSE
      BR 50102$
50101$:
;LAST ADDRESS
LET END := RO - #2
      MOV RO,END
      SUB #2,END
ENDIF
50102$:
LET START := L4KST
      MOV L4KST,START
CALL PRINT ;PRINT ADDRESSES
      JSR PC,PRINT
    
```

POWER UP ROUTINE

2620									
2621	013504					CALL	TAB		:PRINT TAB
	013504	004767	000654						:JSR PC,TAB
2622									
2623	013510					PUSH	R1		:SAVE R1
	013510	010146			MOV		R1,-(SP)		::PUSH R1 ON STACK
2624									
2625	013512					LET	R1 := VOL		:FETCH VOL COUNT
	013512	016701	002446						MOV VOL,R1
2626									
2627	013516					TYPDEC	R1		:PRINT ANY ERRORS
	013516	010146			MOV		R1,-(SP)		::SAVE R1 FOR TYPEOUT
	013520	104405			TYPDS				::GO TYPE--DECIMAL ASCII WITH SIGN
2628									
2629	013522					CALL	TAB		:PRINT TAB
	013522	004767	000636						:JSR PC,TAB
2630									
2631	013526					LET	R1 := PARCNT		:FETCH PARITY COUNT
	013526	016701	002440						MOV PARCNT,R1
2632									
2633	013532					TYPDEC	R1		:PRINT PARITY ERRORS
	013532	010146			MOV		R1,-(SP)		::SAVE R1 FOR TYPEOUT
	013534	104405			TYPDS				::GO TYPE--DECIMAL ASCII WITH SIGN
2634									
2635	013536					IF	PARCNT EQ #10. THEN		:DO WE HAVE 10 OR MORE
	013536	026727	002430	000012					CMP PARCNT,#10.
	013544	001002							BNE 50103\$
									:PARITY ERRORS
2636									
2637									
2638	013546	104401	015100						
2639									
2640	013552								
	013552								50103\$:
2641									
2642	013552					POP	R1		:RESTORE R1
	013552	012601			MOV		(SP)+,R1		::POP STACK INTO R1
2643									
2644	013554					CALL	CRLF2		:PRINT <CR><LF>
	013554	004767	000560						:JSR PC,CRLF2
2645									
2646									
2647	013560								:UP DATE START BANK TO NEXT LEVEL TO BE TESTED
	013560	032767	000001	002372					IF #1 SETIN CONTROL THEN
	013566	001407							BNE #1,CONTROL
									50104\$
2648									
2649	013570								
	013570	013767	172342	002400		LET	START := @#KIPAR1 + #200		
	013576	062767	000200	002372					MOV @#KIPAR1,START
									ADD #200,START
2650									
2651	013604					ELSE			
	013604	000402							BR 50105\$
	013606								50104\$:
2652									
2653	013606								
	013606	010067	002364			LET	START := R0		
									MOV R0,START
2654									
2655	013612					ENDIF			

```

2656 013612                                50105$:
2657 013612                                ENDIF
2658 013612                                50075$:
2659 013612 20$:                            IF R4 EQ #0 THEN ;HAVE WE PRINTED ANYTHING UP TO NOW
    013612 005704                                TST R4
    013614 001132                                BNE 50106$
2660 013616                                IF START EQ L4KST THEN ;MAKE SURE THAT CONTROL IS
2661 013616 026767 002354 002354                CMP START,L4KST
    013624 001013                                BNE 50107$
    ;SET PROPERLY
2662 013626                                IF VOL GE NONVOL THEN
2663 013626 026767 002332 002326                CMP VOL, NONVOL
    013634 002404                                BLT 50110$
2665 013636                                LET CONTROL := CONTROL CLR.BY #100000
2666 013636 042767 100000 002314                BIC #100000,CONTROL
2667 013644 •                                ELSE
2668 013644 000403                                BR 50111$
    013646
2669 013646                                LET CONTROL := CONTROL SET.BY #100000
2670 013646 052767 100000 002304                BIS #100000,CONTROL
2671 013654                                ENDIF
2672 013654                                50111$:
2673 013654                                ENDIF
2674 013654                                50107$:
2675 013654                                IF VOL GE NONVOL THEN ;NO, ARE WE IN NON VOLATILE
2676 013654 026767 002304 002300                CMP VOL, NONVOL
    013662 002451                                BLT 50112$
    ;MEMORY?
    ; WERE WE LOOKING FOR NON-VOLATILE
    IF CONTROL LT #0 THEN
2677 013664                                TST CONTROL
2678 013664 005767 002270                                BGE 50113$
2679 013670 002045
    ;ARE WE A MAPPED SYSTEM?
    IF #1 SETIN CONTROL THEN
2681 013672                                BIT #1,CONTROL
2682 013672 032767 000001 002260                BEQ 50114$
    013700 001407
    ;FETCH LAST 4K BANK
    LET END := L4KST - #200
2684 013702                                MOV L4KST,END
2685 013702 016767 002272 002256                SUB #200,END
    013710 162767 000200 002250
2687 013716                                ELSE
2688 013716 000406                                BR 50115$
    013720                                50114$:
    
```

```

2689
2690
2691
2692 013720
      013720 016767 002254 002240
      013726 162767 000002 002232
      ;FETCH LAST ADDRESS OF
      ;OF NON-VOLATILE MEMORY
      LET END := L4KST - #2
      MOV     L4KST,END
      SUB     #2,END

2693
2694 013734
      013734
      ENDIF
      50115$:

2695
2696 013734
      013734 004767 000450
      CALL   PRINT      JSR     ;PRINT ADDRESSES
      ;PC,PRINT

2697
2698 013740
      013740 004767 000420
      CALL   TAB        JSR     ;PRINT TAB
      ;PC,TAB

2699
2700 013744
      013744 012746 000000
      013750 104405
      MOV     #0,-(SP)
      TYPDS
      ;:SAVE #0 FOR TYPEOUT
      ;:GO TYPE--DECIMAL ASCII WITH SIGN
      TYPDEC #0 ;NO ERRORS

2701
2702 013752
      013752 004767 000406
      CALL   TAB        JSR     ;PRINT TAB
      ;PC,TAB

2703
2704 013756
      013756 012746 000000
      013762 104405
      MOV     #0,-(SP)
      TYPDS
      ;:SAVE #0 FOR TYPEOUT
      ;:GO TYPE--DECIMAL ASCII WITH SIGN
      TYPDEC #0 ;NO PARITY ERRORS

2705
2706 013764
      013764 004767 000350
      CALL   CRLF2      JSR     ;PRINT <CR><LF>
      ;PC,CRLF2

2707
2708 013770
      013770 016767 002204 002200
      LET START := L4KST
      MOV     L4KST,START

2709
2710
2711 013776
      013776 042767 100000 002154
      ;INDICATE NOW LOOKING FOR VOLATILE
      LET CONTROL := CONTROL CLR.BY #100000
      BIC     #100000,CONTROL

2712
2713 014004
      014004
      ENDIF
      50113$:

2714
2715 014004
      014004 000436
      ELSE
      BR     50116$
      50112$:

2716
2717
2718 014006
      014006 005767 002146
      014012 002433
      ;YES, WERE WE LOOKING FOR VOLATILE
      IF CONTROL GE #0 THEN
      TST     CONTROL
      BLT     50117$

2719
2720
2721 014014
      014014 032767 000001 002136
      014022 001407
      ;YES, ARE WE MAPPED
      IF #1 SETIN CONTROL THEN
      BIT     #1,CONTROL
      BEQ     50120$

2722
2723
2724 014024
      014024 016767 002150 002134
      ;FETCH LAST 4K BANK BEFORE CHANGE
      LET END := L4KST - #200
      MOV     L4KST,END
    
```



```

2763 014132          LET VOL := #0          ;CLEAR VOLATILE FLAG
      014132 005067 002026          CLR          VOL
2764 014136          LET NONVOL := #0        ;CLEAR NON-VOLATILE FLAG
2765 014136 005067 002020          CLR          NONVOL
2766 014142          LET PARCNT := #0        ;CLEAR PARITY COUNT
2767 014142 005067 002024          CLR          PARCNT
2768 014146          IF #1 SETIN CONTROL THEN ;ARE WE MAPPED?
2769 014146 032767 000001 002004    BIT          #1,CONTROL
      014154 001411          BEQ          50123$
                                   ;YES
2770
2771
2772 014156          LET @#KIPAR1 := @#KIPAR1 + #200 ;GOTO NEXT BANK
      014156 062737 000200 172342    ADD          #200,@#KIPAR1
2773
2774 014164          LET R0 := #20000         ;RETURN TO ADDRESS 0 PAR1
      014164 012700 020000          MOV          #20000,R0
2775
2776 014170          LET L4KST := @#KIPAR1     ;FETCH NEXT BANK
      014170 013767 172342 002002    MOV          @#KIPAR1,L4KST
2777
2778 014176          ELSE
      014176 000402          BR          50124$
      014200          50123$:
2779
2780 014200          LET L4KST := R0           ;FETCH NEXT BANK
      014200 010067 001774          MOV          R0,L4KST
2781
2782 014204          ENDIF
      014204          50124$:
2783
2784 014204          INLINE <JMP STLOOP>       ;CONTINUE LOOP
      014204 000167 176376          JMP          STLOOP
2785
2786 014210          EDLOOP: IF R4 EQ #0 THEN   ;DO WE NEED TO PRINT LAST MEMORY UNDER TEST
      014210 005704          TST          R4
      014212 001041          BNE          50125$
                                   ;YES
2787
2788
2789 014214          IF #1 SETIN CONTROL THEN   ;ARE WE MAPPED?
      014214 032767 000001 001736    BIT          #1,CONTROL
      014222 001407          BEQ          50126$
2790
2791 014224          LET END := @#KIPAR1 - #200 ;FETCH LAST 4K BANK
      014224 013767 172342 001734    MOV          @#KIPAR1,END
      014232 162767 000200 001726    SUB          #200,END
2792
2793 014240          ELSE
      014240 000405          BR          50127$
      014242          50126$:
2794
2795 014242          LET END := R0 - #2         ;FETCH LAST ADDRESS
      014242 010067 001720          MOV          R0,END
      014246 162767 000002 001712    SUB          #2,END
2796
    
```



```

2797 014254          ENDIF
      014254
2798
2799 014254          CALL PRINT          ;PRINT ADDRESSES
      014254 004767 000130          JSR PC,PRINT
2800
2801 014260          IF CONTROL LT #0 THEN ;WAS LAST CHUNK OF MEMORY
      014260 005767 001674          TST CONTROL
      014264 002012          BGE 50130$
                                   ;NON-VOL - YES, PRINT NO ERRORS
2802
2803
2804 014266          CALL TAB          ;PRINT TAB
      014266 004767 000072          JSR PC,TAB
2805
2806 014272          TYPDEC #0          ;PRINT ANY ERRORS
      014272 012746 000000          MOV #0,-(SP) ;:SAVE #0 FOR TYPEOUT
      014276 104405          TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
2807
2808 014300          CALL TAB          ;PRINT TAB
      014300 004767 000060          JSR PC,TAB
2809
2810 014304          TYPDEC #0          ;PARITY ERRORS
      014304 012746 000000          MOV #0,-(SP) ;:SAVE #0 FOR TYPEOUT
      014310 104405          TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
2811
2812 014312          ENDIF
      014312
2813
2814 014312          CALL CRLF2         ;PRINT <CR><LF>
      014312 004767 000022          JSR PC,CRLF2
2815
2816 014316          ENDIF
      014316
2817
2818 ;FETCH LOCATION OF FINISHED MESSAGE
2819 014316          LET R1 := PC + #TAF-.
      014316 010701          MOV PC,R1
      014320 062701 001037          ADD #TAF-.,R1
2820
2821 ;LOAD ADDRESS INTO TRAP INSTRUCTION
2822 014324          LET 18$ := R1
      014324 010167 000002          MOV R1,18$
2823
2824 ;TELL OPERATOR WE ARE FINISHED
2825 014330 104401          TYPE
2826 014332 015357          18$: .WORD TAF
2827
2828 014334          INLINE <HALT>
      014334 000000          HALT
2829
2830 014336          ENDRTN
      014336
      014336          50000$:
      014336 000207          50001$:
                                   RTS PC
    
```

```
2832 014340          ROUTINE CRLF2
2833 014340
2834                ;SAVE R1
2835 014340 010146   PUSH    R1
2836                MOV     R1,-(SP)      ;;PUSH R1 ON STACK
2837                ;FETCH LOCATION OF <CR><LF> MESSAGE
2838 014342          LET R1 := PC + #CRLF1-.
2839 014342 010701   MOV     PC,R1
2840 014344 062701 001611 ADD     #CRLF1-.,R1
2841                ;LOAD ADDRESS INTO TRAP INSTRUCTION
2842 014350          LET 5$ := R1
2843                MOV     R1,5$
2844                ;TELL OPERATOR <CR><LF>
2845 014354 104401   TYPE
2846 014356 016155   5$: .WORD  CRLF1
2847                ;RESTORE R1
2848 014360          POP     R1
2849 014360 012601   MOV     (SP)+,R1      ;;POP STACK INTO R1
2850                ENDRTN
                50000$:
                50001$:
                RTS     PC
```

```
2852 014364          ROUTINE TAB
      014364
2853
2854                ;SAVE R1
2855 014364          PUSH    R1
      014364 010146  MOV     R1,-(SP)      ;;PUSH R1 ON STACK
2856
2857                ;FETCH LOCATION OF TAB MESSAGE
2858 014366          LET R1 := PC + #TAB1-.
      014366 010701
      014370 062701 001060
                                MOV     PC,R1
                                ADD     #TAB1-.,R1
2859
2860                ;LOAD ADDRESS INTO TRAP INSTRUCTION
2861 014374          LET 6$ := R1
      014374 010167 000002
                                MOV     R1,6$
2862
2863                ;TELL OPERATOR TAB
2864 014400 104401   TYPE
2865 014402 015450   6$: .WORD  TAB1
2866
2867                ;RESTORE R1
2868 014404          POP     R1
      014404 012601  MOV     (SP)+,R1      ;;POP STACK INTO R1
2869
2870          ENDRTN
                                50000$:
                                50001$:
                                RTS     PC
      014406
      014406
      014406 000207
```

```

2872          .SBTTL PRINT ADDRESSES ROUTINE
2873
2874          ;THIS ROUTINE PRINTS THE ADDRESSES IN THE MEMORY MAP
2875          ROUTINE PRINT
2876 014410
2877 014410          PRINT:
2878 014410 032767 000001 001542          IF #1 SETIN CONTROL THEN          ;ARE WE A MAPPED SYSTEM
2879 014416 001416          BIT          #1,CONTROL
2880 014420          LET R2 := START          ;YES, FETCH 4K BANK
2881 014420 016702 001552          MOV          START,R2
2882 014424          LET R1 := #0          ;ALWAYS AT 4K BOUNDARY
2883 014424 005001          CLR          R1
2884 014426          CALL P22BAD          ;PRINT START ADDRESS
2885 014426 004767 000366          JSR          PC,P22BAD
2886 014432          CALL TAB          ;PRINT TAB
2887 014432 004767 177726          JSR          PC,TAB
2888 014436          LET R2 :- END          ;FETCH LAST 4K BANK
2889 014436 016702 001524          MOV          END,R2
2890 014442          LET R1 := #17776          ;LAST ADDRESS IN 4K BANK
2891 014442 012701 017776          MOV          #17776,R1
2892 014446          CALL P22BAD
2893 014446 004767 000346          JSR          PC,P22BAD
2894 014452          ELSE
2895 014452 000442          BR          50003$
2896 014454          LET R1 := START          ;FETCH BEGINNING ADDRESS
2897 014454 016701 001516          MOV          START,R1
2898 014460          IF R1 GE #0 THEN
2899 014460 005701          TST          R1
2900 014462 002405          BLT          50004$
2901 014464          MOV          #0,-(SP)          TYPOCS #0,,4          ;TYPE UPPER BIT AS ZERO
2902 014464 012746 000000          TYPOS          ;:SAVE #0 FOR TYPEOUT
2903 014470 104403          .BYTE          4          ;:GO TYPE--OCTAL ASCII
2904 014472 004          .BYTE          0          ;:TYPE 4 DIGIT(S)
2905 014473 000          .BYTE          0          ;:SUPPRESS LEADING ZEROS
2906 014474          ELSE
2907 014474 000404          BR          50005$
2908 014476          MOV          #1,-(SP)          TYPOCS #1,,4          ;TYPE UPPER BIT AS ONE
2909 014476 012746 000001          TYPOS          ;:SAVE #1 FOR TYPEOUT
2910 014502 104403          .BYTE          4          ;:GO TYPE--OCTAL ASCII
2911 014504 004          .BYTE          4          ;:TYPE 4 DIGIT(S)
    
```

```

2905 014505 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2906 014506 ENDIF
2907 014506 50005$:
2908 014506 010146 TYPOCS R1,,5,X ;PRINT STARTING ADDRESS
014506 104403 MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
014510 104403 TYPOS ;;GO TYPE--OCTAL ASCII
014512 005 .BYTE 5 ;;TYPE 5 DIGIT(S)
014513 001 .BYTE 1 ;;TYPE LEADING ZEROS
2909 014514 CALL TAB ;TAB
2910 014514 004767 177644 JSR PC,TAB
2911 014520 LET R1 := END ;FETCH LAST ADDRESS
2912 014520 016701 001442 MOV END,R1
2913 014524 IF R1 GE #0 THEN
2914 014524 005701 TST R1
014526 002405 BLT 50006$
2915 014530 TYPOCS #0,,4 ;TYPE UPPER BIT AS ZERO
2916 014530 012746 000000 MOV #0,-(SP) ;;SAVE #0 FOR TYPEOUT
014534 104403 TYPOS ;;GO TYPE--OCTAL ASCII
014536 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
014537 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2917 014540 ELSE
2918 014540 000404 BR 50007$
014542 50006$:
2919 014542 TYPOCS #1,,4 ;TYPE UPPER BIT AS ONE
2920 014542 012746 000001 MOV #1,-(SP) ;;SAVE #1 FOR TYPEOUT
014546 104403 TYPOS ;;GO TYPE--OCTAL ASCII
014550 004 .BYTE 4 ;;TYPE 4 DIGIT(S)
014551 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2921 014552 ENDIF
2922 014552 50007$:
2923 014552 TYPOCS R1,,5,X ;PRINT LAST ADDRESS
2924 014552 010146 MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
014554 104403 TYPOS ;;GO TYPE--OCTAL ASCII
014556 005 .BYTE 5 ;;TYPE 5 DIGIT(S)
014557 001 .BYTE 1 ;;TYPE LEADING ZEROS
2925 014560 ENDIF
2926 014560 50003$:
2927 014560 IF CONTROL LT #0 THEN ;IS IT NON-VOLATILE
2928 014560 005767 001374 TST CONTROL
014564 002022 BGE 50010$
2929 014566 ;FETCH LOCATION OF NON VOLATILE MESSAGE
2930 014566 010701 LET R1 := PC + #NVMES-.
2931 014570 062701 000670 MOV PC,R1
ADD #NVMES-.,R1

```

```

2932
2933          ;LOAD ADDRESS INTO TRAP INSTRUCTION
          LET 5$ := R1
          MOV      R1,5$
          014574 010167 000002
2935
2936          ;TELL OPERATOR NON VOLATILE
2937 014600 104401
2938 014602 015460          5$:
          .WORD   NVMES
2939
2940 014604          IF START EQ #0 THEN          ;ARE WE IN PAGE 0
          014604 005767 001366          TST      START
          014610 001007          BNE      50011$
          ;YES
2941
2942
2943 014612          IF #4000 SETIN CONTROL THEN          ;WAS THERE TRAP PAGE
          014612 032767 004000 001340          BIT      #4000,CONTROL
          014620 001403          BEQ      50012$
          ;CHECKSUM ERROR
          ;YES
2944
2945
2946
2947          ;INDICATE PRINT ERROR MESSAGE
2948 014622          LET CONTROL := CONTROL SET.BY #40          ;
          014622 052767 000040 001330          BIS      #40,CONTROL
2949
2950          ENDIF          50012$:
          014630
          014630
2951
2952          ENDIF          50011$:
          014630
          014630
2953
2954          ELSE
          014630          BR      50013$
          014630 000407          50010$:
          014632
2955
2956          ;FETCH LOCATION OF VOLATILE MESSAGE
2957 014632          LET R1 := PC + #VMES-.
          014633 010701          MOV      PC,R1
          014634 062701 000647          ADD      #VMES-.,R1
2958
2959          ;LOAD ADDRESS INTO TRAP INSTRUCTION
2960 014640          LET 6$ := R1
          014640 010167 000002          MOV      R1,6$
2961
2962          ;TELL OPERATOR VOLATILE
2963 014644 104401
2964 014646 015503          6$:
          .WORD   VMES
2965
2966          ENDIF          50013$:
          014650
          014650
2967
2968          ENDRTN          50000$:
          014650          50001$:
          014650          RTS      PC
          014650 000207
    
```

```

2970          .SBTTL  PARITY CATCHER
2971
2972          ;THIS ROUTINE CATCHES PARITY ERRORS AND STORES THEN IN PARCNT
2973
2974 014652          IROUTINE PARTRP
2975 014652          PARTRP:
2976 014652 005267 001314          LET PARCNT := PARCNT + #1          ;INDICATE PARITY ERROR
2977                                     INC          PARCNT
2978          ;DISABLE PARITY
2979 014656 012767 000240 001500          LET TRCATCH := #240          ;IF CSR DOES NOT EXIST THEN JUST RTI
2980                                     MOV          #240,TRCATCH
2981 014664 012702 172100          LET R2 := #172100          ;FIRST CSR ADDRESS
2982                                     MOV          #172100,R2
2983 014670          INCR R1 FROM #0 TO #16. BY #1          ;CLEAR ALL CSRS
2984 014670 005001          CLR          R1
2985 014672 000401          BR          50002$
2986 014674 005201          50003$:
2987 014676          INC          R1
2988 014676 020127 000020          50002$:
2989 014702 003002          CMP          R1,#16.
2990                                     BGT          50004$
2991 014704          LET (R2)+ := #0
2992                                     CLR          (R2)+
2993 014706          ENDINC
2994 014706 000772          BR          50003$
2995 014710          LET TRCATCH := #0          ;RESUME HALT IF TRAP
2996 014710 005067 001450          CLR          TRCATCH
2997 014714          IF #125252 EQ -(R0) THEN          ;RECHECK ADDRESS WITH PARITY OFF
2998 014714 022740 125252          CMP          #125252,-(R0)
2999 014720 001003          BNE          50005$
3000          LET NONVOL := NONVOL + #1          ;WE ARE NON-VOLATILE
3001          INC          NONVOL
3002          ELSE
3003          BR          50006$
3004          LET VOL := VOL + #1          ;WE ARE VOLATILE
3005          INC          VOL
3006          ENDIF
3007          50006$:
3008          LET CONTROL :- CONTROL SET.BY #2          ;INDICATE PARITY ERROR WAS
3009          BIS          #2,CONTROL
3010          ;SERVICED
3011
3012
3013
    
```

```

PARITY CATCHER

3004 014742          LET R0 := R0 + #2          ;RESTORE R0 TO WHAT IT WAS AFTER
      014742 062700 000002          ADD      #2,R0
3005                                     ;TRAP
3006
3007 014746          IF PARCNT LT #10. THEN ;LEAVE PARITY DISABLE WHEN TO MANY ERRORS
      014746 026727 001220 000012    CMP      PARCNT,#10.
      014754 002020          BGE      50007$

3008                                     ;ENABLE PARITY ERRORS THROUGH THE CSRS
3009 LET TRCATCH := #240 ;IF CSR DOES NOT EXIST THEN JUST RTI
3010 014756          MOV      #240,TRCATCH
      014756 012757 000240 001400

3011
3012 014764          LET R2 := #172100          ;FIRST CSR ADDRESS
      014764 012702 172100          MOV      #172100,R2

3013
3014 014770          INCR R1 FROM #0 TO #16. BY #1 ;CLEAR ALL CSRS
      014770 005001          CLR      R1
      014772 000401          BR      50010$
      014774          50011$:
      014774 005201          INC      R1
      014776          50010$:
      014776 020127 000020          CMP      R1,#16.
      015002 003003          BGT      50012$

3015
3016 015004          LET (R2)+ := #1
      015004 012722 000001          MOV      #1,(R2)+

3017
3018 015010          ENDINC
      015010 000771          BR      50011$
      015012          50012$:

3019
3020 015012          LET TRCATCH := #0          ;RESUME HALT IF TRAP
      015012 005067 001346          CLR      TRCATCH

3021
3022 015016          ENDIF
      015016          50007$:

3023
3024 015016          ENDRTI
      015016          50000$:
      015016          50001$:
      015016 000002          RTI
  
```



```

3026                .SBTTL PRINT CONTIGUOUS 22 BIT ADDRESS ROUTINE
3027
3028                ;THIS ROUTINE TAKES THE ADDRESS STORED IN R1 AND THE MEMORY ADDRESS IN
3029                ;R2 AND CREATES A 22 BIT CONTIGUOUS ADDRESS WITH ADDRESS BITS 14 - 0
3030                ;IN R1 AND BITS 21 - 15 IN R2
3031
3032                ROUTINE P22BAD
3033
3034                P22BAD:
3035                PUSH <R1,R2,R2>                ;USE STACK AS SCRATCH PAD AREA
3036                MOV R1,-(SP)                ;;PUSH R1 ON STACK
3037                MOV R2,-(SP)                ;;PUSH R2 ON STACK
3038                MOV R2,-(SP)                ;;PUSH R2 ON STACK
3039
3040                LET (SP) := (SP) SHIFT -2        ;PUT BITS 8 & 7 INTO 6 & 5
3041                ASR (SP)
3042                ASR (SP)
3043
3044                LET (SP) := SWAP (SP)            ;PUT BITS 6 & 5 INTO 14 & 13
3045                SWAB (SP)
3046
3047                LET (SP) := (SP) CLR.BY #117777 ;CLEAR OUT UN-IMPORTANT BITS
3048                BIC #117777,(SP)
3049
3050                LET R1 := R1 CLR.BY #160000     ;CLEAR OUT MEMORY MAN. POINTERS
3051                BIC #160000,R1
3052
3053                LET R1 := R1 SET.BY (SP)+        ;TRANSFER BITS 8 & 7 IN R2 INTO BITS
3054                BIS (SP)+,R1                    ;14 & 13 IN R1
3055
3056                LET R2 := R2 SHIFT -1          ;FETCH ADDRESS BITS & PUT BIT 8 INTO
3057                ASR R2                          ;LOWER BYTE
3058
3059                LET R2 := SWAP R2                ;NOW ADDRESS BITS 21-15 ARE IN LSB
3060                SWAB R2
3061
3062                LET R2 := R2 CLR.BY #177600     ;CLEAR OUT ALL OTHER BITS
3063                BIC #177600,R2
3064
3065                TYPOCS R2,,4                    ;PRINT ADDRESS BITS 21 - 15
3066                MOV R2,-(SP)                    ;;SAVE R2 FOR TYPEOUT
3067                TYPOS                            ;;GO TYPE--OCTAL ASCII
3068                .BYTE 4                          ;;TYPE 4 DIGIT(S)
3069                .BYTE 0                          ;;SUPPRESS LEADING ZEROS
3070
3071                TYPOCS R1,,5,X                  ;PRINT ADDRESS BITS 14 - 0
3072                MOV R1,-(SP)                    ;;SAVE R1 FOR TYPEOUT
3073                TYPOS                            ;;GO TYPE--OCTAL ASCII
3074                .BYTE 5                          ;;TYPE 5 DIGIT(S)
3075                .BYTE 1                          ;;TYPE LEADING ZEROS
3076
3077                POP <R2,R1>
3078                MOV (SP)+,R2                    ;;POP STACK INTO R2
3079                MOV (SP)+,R1                    ;;POP STACK INTO R1
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
    
```

3060 015076
015076
015076
015076 000207

ENDRTN

50000\$:
50001\$:
RTS PC

```

3062          .SBTTL  DEFINITIONS
3063
3064 015100      040      117      122  MORE:  .ASCIZ  ' OR MORE '
3065 015111      124      122      101  TRAPER: .ASCII  'TRAP LOCATIONS (ADDRESS 0 - 376) HAVE CHECK SUM ERROR'
3066 015176      040      111      116      .ASCIZ  ' IN NON - VOLATILE MEMORY'<12><12><15>
3067 015233      012      012      015  ERR:    .ASCII  '<12><12><15>'CHECK SUM ERROR. THE MEMORY THE PROGRAM IS'
3068 015310      040      122      105      .ASCIZ  ' RESIDENT'<12><15>'IN HAS VOLATILE LOCATIONS'<12><15>
3069 015357      012      012      015  TAF:    .ASCIZ  '<12><12><15>'THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST'<12><15>
3070 015450      040      040      040  TAB1:   .ASCIZ  '
3071 015460      040      040      040  NVMES:  .ASCIZ  '          NON VOLATILE'
3072 015503      040      040      040  VMES:   .ASCIZ  '          VOLATILE'
3073 015526      012      012      015  HEAD:   .ASCII  '<12><12><15>
3074 015531      040      040      040      .ASCII  '
3075 015604      040      040      040      .ASCII  '          MEMORY MAP'<12><15>
3076 015660      123      124      101      .ASCII  '-----'<15><12><12>
3077 015756      120      101      122      .ASCIZ  'START ADR.          END ADR.          MEMORY TYPE          WORD ERROR'
3078 015776      120      114      105  PWRDWN: .ASCII  'PARITY ERROR'<12><12><15>
3079 016060      040      116      117      .ASCII  'PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT'
3080 016113      040      105      130      .ASCIZ  ' NO LONGER'<12><15>'THAN 100 HOURS.'
3081 016155      012      015      000  CRLF1:  .ASCIZ  ' EXECUTE THE RESTART HELP FILE.'<12><15>
3082          .EVEN
3083 016160      000000  CONTRO: .WORD  0
3084 016162      000000  NONVOL: .WORD  0
3085 016164      000000  VOL:    .WORD  0
3086 016166      000000  END:    .WORD  0
3087 016170      000000  PPOINT: .WORD  0
3088 016172      000000  PARCNT: .WORD  0
3089 016174      000000  ENDADD: .WORD  0
3090 016176      000000  START:  .WORD  0
3091 016200      000000  L4KST:  .WORD  0
    
```

```

3093 ;THIS ROUTINE BUMPS THE WRITE POINTER 4K IF A WRITE TO A ROM MODULE CAUSES A
3094 ;TRAP
3095
3096 016202 ;ROUTINE ROMMD
    016202
3097
3098 016202 ;ARE WE IN A MAPPED SYSTEM
    016202 032767 000001 177750 IF #1 SETIN CONTROL THEN
    016210 001403 ;YES
3099
3100
3101 016212 ;BUMP POINTER TO INDICATE
    016212 012700 040000 LET R0 := #40000 ;WF HAVE REACHED THE END
3102 ;OF THIS 4K CHUNK OF MEMORY
3103
3104
3105 016216 ;ELSE
    016216 000403 BR 50003$
    016220 50002$:
3106
3107 016220 ;BUMP POINTER BACK TO WHAT IT
    016220 005740 INLINE <TST -(R0)> ;WAS BEFORE TRAP
3108
3109
3110 016222 ;BUMP TO NEXT 4K BOUNDARY
    016222 062700 040000 LET R0 := R0 + #40000 ADD #40000,R0
3111
3112 016226 ;
    016226 ; ENDIF
3113
3114 016226 ;
    016226 ; ENDRTI
    016226 000002
    50000$:
    50001$:
    RTI
    
```



```

3149
3150 016322          LET (R0)+ := (R0)+ + #STRAP-.  ;' '
      016322 062720 001104          ADD      #STRAP-.,(R0)+
3151
3152 016326          LET (R0)+ := #340          ;LOAD PSW
      016326 012720 000340          MOV      #340,(R0)+
3153
3154 016332          ELSE
      016332 000403          BR      50010$
      016334          50007$:
3155
3156 016334          LET (R0)+ := R1          ;LOAD TRAP CATCHER
      016334 010120          MOV      R1,(R0)+
3157
3158 016336          LET (R0)+ := #340          ;LOAD PSW
      016336 012720 000340          MOV      #340,(R0)+
3159
3160 016342          ENDIF
      016342          50010$:
3161
3162 016342          ENDLLOOP
      016342 000752          BR      50004$
      016344          50005$:
3163
3164 016344          IF #1 SETIN CONTROL THEN  ;ARE WE IN A MAPPED SYSTEM
      016344 032767 000001 177606          BIT      #1,CONTROL
      016352 001402          BEQ      50011$
3165
3166 016354          MOV      POP @#KIPAR1      ;:POP STACK INTO @#KIPAR1
      016354 012637 172342          (SP)+,@#KIPAR1
3167
3168 016360          ENDIF
      016360          50011$:
3169
3170 016360          POP      R0
      016360 012600          MOV      (SP)+,R0      ;:POP STACK INTO R0
3171
3172 016362          ENDRTN
      016362          50000$:
      016362          50001$:
      016362 000207          RTS      PC
3173
3174          ;DO AN RTI WHENEVER IT TRAPS.
3175
3176 016364          IROUTINE TRCATCH
      016364          TRCATCH:
3177
3178 016364          INLINE <HALT> ;FOR DEBUG PURPOSE REPLACE WITH A HALT INSTRUCTION
      016364 000000          HALT
3179
3180 016366          ENDRTI
      016366          50000$:
      016366          50001$:
      016366 000002          RTI

```

```

3182
3183
3184 016370      .LIST  MEB
016370 105767 000313 $TYPE:  TSTB  $TPFLG1  ;; IS THERE A TERMINAL?
016374 100002      BPL  1$      ;; BR IF YES
016376 000000      HALT      ;; HALT HERE IF NO TERMINAL
016400 000407      BR  3$      ;; LEAVE
016402 010046      1$:  MOV  R0,-(SP)  ;; SAVE R0
016404 017600 000002  MOV  @2(SP),R0  ;; GET ADDRESS OF ASCIZ STRING
016410 112046      2$:  MOVB (R0)+,-(SP)  ;; PUSH CHARACTER TO BE TYPED ONTO STACK
016412 001005      BNE  4$      ;; BR IF IT ISN'T THE TERMINATOR
016414 005726      TST  (SP)+  ;; IF TERMINATOR POP IT OFF THE STACK
016416 012600      60$:  MOV  (SP)+,R0  ;; RESTORE R0
016420 062716 000002  3$:  ADD  #2,(SP)  ;; ADJUST RETURN PC
016424 000002      RTI      ;; RETURN
016426 122716 000011  4$:  CMPB #HT,(SP)  ;; BRANCH IF <HT>
016432 001435      BEQ  8$      ;; BRANCH IF NOT <CRLF>
016434 122716 000200  CMPB #CRLF,(SP)
016440 001013      BNE  5$      ;; POP <CR><LF> EQUIV
016442 005726      TST  (SP)+
016444 010746      MOV  PC,-(SP)
016446 062716 000242  ADD  #CRLF1-.,(SP)
016452 012667 000002  MOV  (SP)+,20$
016456 104401      TYPE
016460 016710      20$:  .WORD $CRLF1  ;; TYPE A CR AND LF
016462 105067 000202  CLRB $CHARCNT  ;; CLEAR CHARACTER COUNT
016466 000750      BR  2$      ;; GET NEXT CHARACTER
016470 004767 000056  5$:  JSR  PC,$TYPEC  ;; GO TYPE THIS CHARACTER
016474 126726 000206  6$:  CMPB $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
016500 001343      BNE  2$      ;; IF NO GO GET NEXT CHAR.
016502 016746 000176  MOV  $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
016506 105366 000001  7$:  DECB 1(SP)  ;; DOES A NULL NEED TO BE TYPED?
016512 002770      BLT  6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
016514 004767 000032  JSR  PC,$TYPEC  ;; GO TYPE A NULL
016520 105367 000144  DECB $CHARCNT  ;; DO NOT COUNT AS A COUNT
016524 000770      BR  7$      ;; LOOP
016526 112716 000040  8$:  MOVB #' ,(SP)  ;; REPLACE TAB WITH SPACE
016532 004767 000014  9$:  JSR  PC,$TYPEC  ;; TYPE A SPACE
016536 132767 000007 000124  BITB #7,$CHARCNT  ;; BRANCH IF NOT AT
016544 001372      BNE  9$      ;; TAB STOP
016546 005726      TST  (SP)+  ;; POP SPACE OFF STACK
016550 000717      BR  2$      ;; GET NEXT CHARACTER
016552 105777 000116  $TYPEC: TSTB @STKS1  ;; CHAR IN KYBD BUFFER?
016556 100022      BPL  10$     ;; BR IF NOT
016560 017746 000112  MOV  @STKB1,-(SP)  ;; GET CHAR
016564 042716 177600  BIC  #177600,(SP)  ;; STRIP EXTRANEIOUS BITS
016570 122716 000023  CMPB #XOFF,(SP)  ;; WAS CHAR XOFF
016574 001012      BNE  102$   ;; BR IF NOT
016576 105777 000072  101$: TSTB @STKS1  ;; WAIT FOR CHAR
016602 100375      BPL  101$
016604 117716 000066  MOVB @STKB1,(SP)  ;; GET CHAR
016610 042716 177600  BIC  #177600,(SP)  ;; STRIP IT
016614 122716 000021  CMPB #XON,(SP)  ;; WAS IT XON?
016620 001366      BNE  101$   ;; BR IF NOT
016622 005726      102$: TST  (SP)+  ;; FIX STACK
016624 105777 000050  10$:  TSTB @STPS1  ;; WAIT UNTIL PRINTER IS READY
016630 100375      BPL  10$
    
```

```

016632 116677 000002 000042      MOVB 2(SP),@STPB1      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
016640 122766 000015 000002      CMPB #CR,2(SP)        ;;IS CHARACTER A CARRIAGE RETURN?
016646 001003                      BNE 1$                ;;BRANCH IF NO
016650 105067 000014                      CLRB $CHARCNT         ;;YES--CLEAR CHARACTER COUNT
016654 000406                      BR $TYPEX             ;;EXIT
016656 122766 000012 000002 1$:  CMPB #LF,2(SP)        ;;IS CHARACTER A LINE FEED?
016664 001402                      BEQ $TYPEX            ;;BRANCH IF YES
016666 105227                      INCB (PC)+            ;;COUNT THE CHARACTER
016670 000000                      $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
016672 000207                      $TYPEX: RTS PC
016674 177560                      .IIF EQ .-$TKS1,$TKS1: .WORD 177560      ;;TTY KDB STATUS      :MJD001
016676 177562                      .IIF EQ .-$TKB1,$TKB1: .WORD 177562      ;;TTY KDB BUFFER      :MJD001
016700 177564                      .IIF EQ .-$TPS1,$TPS1: .WORD 177564      ;;TTY PRINTER STATUS REG. ADDRESS
016702 177566                      .IIF EQ .-$TPB1,$TPB1: .WORD 177566      ;;TTY PRINTER BUFFER REG. ADDRESS
016704 000                      .IIF EQ .-$NULL,$NULL: .BYTE 0           ;;CONTAINS NULL CHARACTER FOR FILLS
016705 002                      .IIF EQ .-$FILLS,$FILLS: .BYTE 2       ;;CONTAINS # OF FILLER CHARACTERS RE
016706 012                      .IIF EQ .-$FILLC,$FILLC: .BYTE 12      ;;INSERT FILL CHARS. AFTER A 'LINE F
016707 000                      .IIF EQ .-$TPFLG1,$TPFLG1: .BYTE 0      ;;'TERMINAL AVAILABLE' FLAG (BIT<07>
016710 015                      .IIF EQ .-$CRLF1,$CRLF1: .ASCII <15>   ;;CARRIAGE RETURN
016711 012 000                      .IIF EQ .-$LF1,$LF1: .ASCII <12>      ;;LINEFEED

```

3185
3186

```

016714                      .STYPDEC
016714 010046      MOV R0,-(SP)      ;;PUSH R0 ON STACK
016716 010146      MOV R1,-(SP)      ;;PUSH R1 ON STACK
016720 010246      MOV R2,-(SP)      ;;PUSH R2 ON STACK
016722 010346      MOV R3,-(SP)      ;;PUSH R3 ON STACK
016724 010446      MOV R4,-(SP)      ;;PUSH R4 ON STACK
016726 010546      MOV R5,-(SP)      ;;PUSH R5 ON STACK
016730 012746 020200  MOV #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
016734 016605 000022  MOV 22(SP),R5     ;;GET THE INPUT NUMBER
016740 100004      BPL 1$            ;;BR IF INPUT IS POS.
016742 005405      NEG R5           ;;MAKE THE BINARY NUMBER POS.
016744 112766 000055 000001 1$:  MOVB #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
016752 005000      CLR R0           ;;ZERO THE CONSTANTS INDEX
016754 010703      MOV PC,R3        ;;FETCH PC FOR ADDRESSING
016756 062703 000200  ADD #SDBLK-.,R3   ;;SETUP THE OUTPUT POINTER - R3 POINTS TO SDBLK
016762 112723 000040  MOVB #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
016766 005002      CLR R2           ;;CLEAR THE BCD NUMBER
016770 010704      MOV PC,R4        ;;USE PC FOR LOCATING TABLE
016772 062704 000154  ADD #SDTBL-.,R4  ;;R4 NOW POINTS TO TABLE
016776 060004      ADD R0,R4        ;;INDEX INTO TABLE
017000 011401      MOV (R4),R1      ;;GET THE CONSTANT
017002 160105      3$:  SUB R1,R5        ;;FORM THIS BCD DIGIT
017004 002402      BLT 4$          ;;BR IF DONE
017006 005202      INC R2           ;;INCREASE THE BCD DIGIT BY 1
017010 000774      BR 3$
017012 060105      4$:  ADD R1,R5        ;;ADD BACK THE CONSTANT
017014 005702      TST R2           ;;CHECK IF BCD DIGIT=0
017016 001002      BNE 5$          ;;FALL THROUGH IF 0
017020 105716      TSTB (SP)        ;;STILL DOING LEADING 0'S?
017022 100407      BMI 7$          ;;BR IF YES
017024 106316      5$:  ASLB (SP)        ;;MSD?
017026 103003      BCC 6$          ;;BR IF NO
017030 116663 000001 177777  MOVB 1(SP),-1(P3)  ;;YES--SET THE SIGN
017036 052702 000060  6$:  BIS #'0,R2        ;;MAKE THE BCD DIGIT ASCII
017042 052702 000040  7$:  BIS #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
017046 110223      MOVB R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER

```



```

017050 005720          TST      (R0)+          ;;JUST INCREMENTING
017052 020027 000010  CMP      R0,#10        ;;CHECK THE TABLE INDEX
017056 002743          BLT      2$                ;;GO DO THE NEXT DIGIT
017060 003002          BGT      8$                ;;GO TO EXIT
017062 010502          MOV      R5,R2          ;;GET THE LSD
017064 000764          BR       6$                ;;GO CHANGE TO ASCII
017066 105726          8$:   TSTB   (SP)+          ;;WAS THE LSD THE FIRST NON-ZERO?
017070 100003          BPL      9$                ;;BR IF NO
017072 116663 177777 177776 9$:   MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
017100 105013          CLRB   (R3)            ;;SET THE TERMINATOR
017102 012605          MOV      (SP)+,R5        ;;POP STACK INTO R5
017104 012604          MOV      (SP)+,R4        ;;POP STACK INTO R4
017106 012603          MOV      (SP)+,R3        ;;POP STACK INTO R3
017110 012602          MOV      (SP)+,R2        ;;POP STACK INTO R2
017112 012601          MOV      (SP)+,R1        ;;POP STACK INTO R1
017114 012600          MOV      (SP)+,R0        ;;POP STACK INTO R0
017116 010746          MOV      PC,-(SP)        ;;FETCH ADDRESS OF NUMBER TO BE TYPED USING
017120 062716 000036  ADD      #5DBLK-,(SP)    ;;STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
017124 012667 000002  MOV      (SP)+,10$       ;;SET UP ADDRESS FOR TYPE COMMAND
017130 104401          TYPE   .WORD           ;;NOW TYPE THE NUMBER
017132 017156          10$:  .WORD   $DBLK
017134 016666 000002 000004  MOV      2(SP),4(SP)    ;;ADJUST THE STACK
017142 012616          MOV      (SP)+,(SP)
017144 000002          RTI
017146 023420          $DTBL: 10000.
017150 001750          1000.
017152 000144          100.
017154 000012          10.

3187 017166          .STYPOCT
3188 017166 017646 000000  $TYPOS: MOV      @2(SP),-(SP)    ;;PICKUP THE MODE
017172 116667 000001 000223  MOVB   1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
017200 112667 000221          MOVB   (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
017204 062716 000002          ADD      #2,(SP)        ;;ADJUST RETURN ADDRESS
017210 000406          BR       $TYPON
017212 112767 000001 000203  $TYPOC: MOVB   #1,$OFILL    ;;SET THE ZERO FILL SWITCH
017220 112767 000006 000177  MOVB   #6,$OMODE+1    ;;SET FOR SIX(6) DIGITS
017226 112767 000005 000166  $TYPON: MOVB   #5,$OCNT    ;;SET THE ITERATION COUNT
017234 010346          MOV      R3,-(SP)        ;;SAVE R3
017236 010446          MOV      R4,-(SP)        ;;SAVE R4
017240 010546          MOV      R5,-(SP)        ;;SAVE R5
017242 116704 000157  MOVB   $OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
017246 005404          NEG      R4
017250 062704 000006          ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
017254 110467 000144          MOVB   R4,$OMODE        ;;SAVE IT FOR USE
017260 116704 000137          MOVB   $OFILL,R4        ;;GET THE ZERO FILL SWITCH
017264 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
017270 005003          CLR      R3            ;;CLEAR THE OUTPUT WORD
017272 006105          1$:   ROL      R5          ;;ROTATE MSB INTO 'C'
017274 000404          BR       3$            ;;GO DO MSB
017276 006105          2$:   ROL      R5          ;;FORM THIS DIGIT
017300 006105          ROL      R5
017302 006105          ROL      R5
017304 010503          MOV      R5,R3
017306 006103          3$:   ROL      R3          ;;GET LSB OF THIS DIGIT
017310 105367 000110  DECB   $OMODE        ;;TYPE THIS DIGIT?
017314 100023          BPL      7$            ;;BR IF NO
    
```

```

017316 042703 177770      BIC      #177770,R3      ;;GET RID OF JUNK
017322 001002      BNE      4$            ;;TEST FOR 0
017324 005704      TST      R4            ;;SUPPRESS THIS 0?
017326 001403      BEQ      5$            ;;BR IF YES
017330 005204      4$: INC      R4            ;;DON'T SUPPRESS ANYMORE 0'S
017332 052703 000060      BIS      #'0,R3        ;;MAKE THIS DIGIT ASCII
017336 052703 000040      5$: BIS      #' ,R3        ;;MAKE ASCII IF NOT ALREADY
017342 110367 000052      MOV      R3,8$         ;;SAVE FOR TYPING
017346 010746      MOV      PC,-(SP)      ;;FETCH ADDRESS OF NUMBER TO BE TYPED USING
017350 062716 000050      ADD      #8$-.,(SP)   ;;STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
017354 012667 000002      MOV      (SP)+,10$     ;;SET UP ADDRESS FOR TYPE COMMAND
017360 104401      TYPE                                ;;GO TYPE THIS DIGIT
017362 017420      10$: .WORD 8$          ;;ADDRESS OF CHARACTER TO BE PRINTED
017364 105367 000032      7$: DEC      $OCNT      ;;COUNT BY 1
017370 003342      BGT      2$            ;;BR IF MORE TO DO
017372 002402      BLT      6$            ;;BR IF DONE
017374 005204      INC      R4            ;;INSURE LAST DIGIT ISN'T A BLANK
017376 000737      BR       2$            ;;GO DO THE LAST DIGIT
017400 012605      6$: MOV      (SP)+,R5     ;;RESTORE R5
017402 012604      MOV      (SP)+,R4     ;;RESTORE R4
017404 012603      MOV      (SP)+,R3     ;;RESTORE R3
017406 016666 000002 000004      MOV      2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
017414 012616      MOV      (SP)+,(SP)
017416 000002      RTI                                ;;RETURN
017420      000      8$: .BYTE 0        ;;STORAGE FOR ASCII DIGIT
017421      000      .BYTE 0        ;;TERMINATOR FOR TYPE ROUTINE
017422      000      $OCNT: .BYTE 0    ;;OCTAL DIGIT COUNTER
017423      000      $OFILL: .BYTE 0  ;;ZERO FILL SWITCH
017424 000000      $OMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE

3189
3190 017426      .STRAP
017426 010046      $STRAP: MOV      R0,-(SP)  ;;SAVE R0
017430 010146      MOV      R1,-(SP)  ;;SAVE R1
017432 016600 000004      MOV      4(SP),R0   ;;GET TRAP ADDRESS
017436 005740      TST      -(R0)      ;;BACKUP BY 2
017440 111000      MOV      (R0),R0    ;;GET RIGHT BYTE OF TRAP
017442 006300      ASL      R0         ;;POSITION FOR INDEXING
017444 010701      MOV      PC,R1      ;;FETCH THIS PROGRAM POINTER FOR PIC
017446 062701 000026      ADD      #STRPAD-.,R1 ;;POINT TO TABLE
017452 060001      ADD      R0,R1      ;;R1 NOW POINTS TO ROUTINE TRAP CALL WANTED
017454 011100      MOV      (R1),R0    ;;R0 CONTAINS ADDRESS FOR RTS
017456 012601      MOV      (SP)+,R1   ;;RESTORE R1
017460 000200      RTS      R0         ;;GO TO ROUTINE
017462 011646      $STRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
017464 016666 000004 000002      MOV      4(SP),2(SP) ;;MOVE THE PSW DOWN
017472 000002      RTI                                ;;RESTORE THE PSW
017474 017462      $TRPAD: .WORD $STRAP2
017476 016370      $TYPE  ;;CALL=TYPE
017500 017212      $TYPOC ;;CALL=TYPOC
017502 017166      $TYPOS ;;CALL=TYPOS
017504 017226      $TYPON ;;CALL=TYPON
017506 016714      $TYPDS ;;CALL=TYPDS
017510 000404      $RDCHR ;;CALL=RDCHR
017512 000534      $RDLIN ;;CALL=RDLIN
017514 000730      $RDOCT ;;CALL=PDOCT
017516 001070      $RDDEC ;;CALL=RDDEC

TRAP+1(104401) TTY TYPEOUT ROUTINE
TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
TRAP+7(104407) TTY TYPEIN STRING ROUTINE
TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
TRAP+11(104411) READ A DECIMAL NUMBER FROM TTY
    
```

3192	017520	000000	CSTRPG:	.WORD	0
3193	017522	000000	CHKSUM:	.WORD	0
3194	017524			.BLKW	50
3195	017644	000000	SPINIT:	.WORD	0
3196					
3197	017646	000000	ENDPR:	.WORD	0
3198					
3199		000001		.END	

ADDCHK	002074	HELLO	006302	Q4	011632	SW14	= 040000	\$F\$INC=	000210
ASSEMB=	000010	HELP	006174	RDCHR	= 104406	SW15	= 100000	\$F\$L00=	000200
BANKO	006404	HELPF	002226	RDDEC	= 104411	SW2	= 000004	\$F\$NAM=	000160
BIT0	= 000001	HT	= 000011	RDLIN	= 104407	SW3	= 000010	\$F\$NO	= 000403
BIT00	= 000001	ILLADD	006236	RDOCT	= 104410	SW4	= 000020	\$F\$OR	= 000320
BIT01	= 000002	IOTVEC=	000020	RESHLP	007326	SW5	= 000040	\$F\$RTI=	000350
BIT02	= 000004	K	006470	RESVEC=	000010	SW6	= 000100	\$F\$RTN=	000300
BIT03	= 000010	KIPAR0=	172340	RHELP	010066	SW7	= 000200	\$F\$SEL=	000140
BIT04	= 000020	KIPAR1=	172342	ROMMD	016202	SW8	= 000400	\$F\$THE=	000330
BIT05	= 000040	KIPAR2=	172344	R6	= %000006	SW9	= 001000	\$F\$TRU=	000404
BIT06	= 000100	KIPAR3=	172346	R7	= %000007	SYMD	= 000000	\$F\$UNT=	000130
BIT07	= 000200	KIPAR4=	172350	SAVHI	010346	SYMS	= 000007	\$F\$WHI=	000120
BIT08	= 000400	KIPAR5=	172352	SAVLOW	010344	TAB	014364	\$F\$YES=	000402
BIT09	= 001000	KIPAR6=	172354	SCOPE	= 000004	TAB1	015450	\$HD	= 000003
BIT1	= 000002	KIPAR7=	172356	SIPAR0=	172240	TAF	015357	\$HIOCT	001066
BIT10	= 002000	KIPDR0=	172300	SIPAR1=	172242	TBITVE=	000014	\$IFLEV=	177777
BIT11	= 004000	KIPDR1=	172302	SIPAR2=	172244	TKVEC	= 000060	\$ISK0	= 000001
BIT12	= 010000	KIPDR2=	172304	SIPAR3=	172246	TPVEC	= 000064	\$ISK1	= 000001
BIT13	= 020000	KIPDR3=	172306	SIPAR4=	172250	TRAPCT	016230	\$ISK2	= 000001
BIT14	= 040000	KIPDR4=	172310	SIPAR5=	172252	TRAPER	015111	\$ISK3	= 000001
BIT15	= 100000	KIPDR5=	172312	SIPAR6=	172254	TRAPVE=	000034	\$KTNEX	001622
BIT2	= 000004	KIPDR6=	172314	SIPAR7=	172256	TRCATC	016364	\$KTOUT	001560
BIT3	= 000010	KIPDR7=	172316	SIPDR0=	172200	TRTVEC=	000014	\$KT11	001332
BIT4	= 000020	LF	= 000012	SIPDR1=	172202	TYPDS	= 104405	\$LF	000672
BIT5	= 000040	L4KST	016200	SIPDR2=	172204	TYPE	= 104401	\$LF1	016711
BIT6	= 000100	MAPPED	006506	SIPDR3=	172206	TYPOC	= 104402	\$LOCTA=	177777
BIT7	= 000200	MMR3	007713	SIPDR4=	172210	TYPON	= 104404	\$LSTAD	001736
BIT8	= 000400	MMVEC	= 000250	SIPDR5=	172212	TYPOS	= 104403	\$LSTBK	001740
BIT9	= 001000	MORE	015100	SIPDR6=	172214	UNMAP	006572	\$LSTIN=	000001
BPTVEC=	000014	NOBLK	007237	SIPDR7=	172216	VMES	015503	\$LSTTA=	000001
BYTECO=	000403	NONVOL	016162	SPINIT	017644	VOL	016164	\$MAP	001424
CHKSUM	017522	NOT4K	010152	SR0	= 177572	WHERE	006664	\$MAPRG	001430
CONTR0	016160	NVMES	015460	SR1	= 177574	WHERE1	007006	\$MMOUT	001602
CR	= 000015	OUTMEM	010224	SR2	= 177576	WHOLE	007124	\$MNEW	000717
CRLF	= 000200	PARCNT	016172	SR3	= 172516	\$BGNLE=	177777	\$MSWR	000706
CRLF1	016155	PARTRP	014652	STACK	= 001100	\$BRJMP=	177777	\$NESTL=	177777
CRLF2	014340	PAR0	007506	START	016176	\$CHARC	016670	\$NOMAP	001430
CR1	007477	PAR7	007536	START1	010350	\$CNTLG	000701	\$NSK0	= 000350
CSTRPG	017520	PIRQ	= 177772	STKLMT=	177774	\$CNTLU	000674	\$NSK1	= 000110
DDISP	= 177570	PIRQVE=	000240	STLOOP	012606	\$SCORE	001630	\$NSK2	= 000110
DIAGMC=	000000	PPOINT	016170	SWREG	000176	\$CRLF	000671	\$NSK3	= 000110
DIF1	= 006014	PRINT	014410	SW0	= 000001	\$CRLF1	016710	\$NSK4	= 000110
DISPRE	000174	PRO	= 000000	SW00	= 000001	\$CROUT	001660	\$NULL	016704
DSWR	= 177570	PR1	= 000040	SW01	= 000002	\$DBLK	017156	\$OCNT	017422
ECHO	010342	PR2	= 000100	SW02	= 000004	\$DTBL	017146	\$OMODE	017424
EDLOOP	014210	PR3	= 000140	SW03	= 000010	\$ERFLG=	000400	\$QUES	000670
EMTVEC=	000030	PR4	= 000200	SW04	= 000020	\$FILLC	016706	\$RDCHR	000404
ENABLE	007747	PR5	= 000240	SW05	= 000040	\$FILLS	016705	\$RDDEC	001070
END	016166	PR6	= 000300	SW06	= 000100	\$F\$AND=	000310	\$RDLIN	000534
ENDADD	016174	PR7	= 000340	SW07	= 000200	\$F\$BAD=	000401	\$RDOCT	000730
ENDPR	017646	PS	= 177776	SW08	= 000400	\$F\$BLA=	000170	\$RDSZ	= 000026
ERR	015233	PSW	= 177776	SW09	= 001000	\$F\$CAS=	000150	\$SAVE	= 000001
ERROR	= 104000	PWRDWN	015776	SW1	= 000002	\$F\$DEC=	000220	\$SAVLE=	177777
ERRVEC=	000004	PWRUP	012072	SW10	= 002000	\$F\$DO	= 000340	\$SELLE=	177777
FETADD	001762	PWRVEC=	000024	SW11	= 004000	\$F\$FAL=	000405	\$SETUP=	000004
FETCH	001742	P22BAD	015020	SW12	= 010000	\$F\$G00=	000400	\$SIZE	001246
HEAD	015526	Q1	011034	SW13	= 020000	\$F\$IF	= 000110	\$SIZEX	001664

SYMBOL TABLE

\$SSKO = 050005	\$TPB1 016702	\$TSK5 = 050037	\$SARGC= 000000	\$\$RETU= 000000
\$STOP 001426	\$TPFLG 016707	\$TTYIN 000642	\$\$BYTE= 000403	\$\$RTN1= 050000
\$STUP = 177777	\$TPS1 016700	\$TYPDS 016714	\$\$CASE= 000000	\$\$RTN2= 050001
\$SWR = 160000	\$TRAP 017426	\$TYPE 016370	\$\$DST = 000000	\$\$SRC = 000000
\$TAGLE= 177777	\$TRAP2 017462	\$TYPEC 016552	\$\$ELOC= 000402	\$\$TGSV= 000000
\$TAGNU= 050002	\$TRP = 000012	\$TYPEX 016672	\$\$ERFL= 000000	\$\$TGS1= 000000
\$TEMP = 000350	\$TRPAD 017474	\$TYPOC 017212	\$\$FLAG= 000001	\$\$TGS2= 000000
\$TKB 000402	\$TSK0 = 050011	\$TYPON 017226	\$\$FROM= 000000	\$\$TO = 000000
\$TKB1 016676	\$TSK1 = 050005	\$TYPOS 017166	\$\$INH = 000402	\$\$TOTL= 000000
\$TKS 000400	\$TSK2 = 050010	\$U = 000403	\$\$LOC = 016352	\$\$TAG= 050000
\$TKS1 016674	\$TSK3 = 050121	\$XOFF = 000023	\$\$LOCN= 000000	\$OFILL 017423
\$TN = 000001	\$TSK4 = 050061	\$XON = 000021	\$\$REG = 177777	

. ABS. 017650 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 60528 WORDS (237 PAGES)
DYNAMIC MEMORY: 21870 WORDS (84 PAGES)
ELAPSED TIME: 00:22:02