

.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 DETECTION 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 VARITY 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 (0) 0?

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 POSTION, 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	137776	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:

@\$7/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	17776	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:

@\$7/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:

@\$7/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	ERRCR	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

8

000000

.ENABLE ABS
 .DSABLE GBL

.LIST SEQ,BIN,CND

```

889
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 BK001
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 $KT11: .WORD 200 ;;SET TO USE MEMORY MANAGEMENT
931 BPL SCORE ;;BR IF NO
932 MOV #SKTNEC,@#ERRVEC ;;SET FOR TIMEOUT
933 TST @#SR0 ;;KT11 ARE YOU THERE?
934 BIS #100000,$KT11 ;;YES--SET KT11 KEY
935 MOV #100$,@#ERRVEC ;;SET FOR TIMEOUT BK001
936 TST @#172516 ;;Q-BUS MAP ARE YOU THERE? BK001
937 MOV #200,$MAP ;;TURN ON MAP INDICATOR BK001
938 MOV #176200,$STOP ;;END OF 2M OF MEMORY
939 BR $MAPRG ;;GO SET UP MAP REGISTERS BK001
940 100$: MOV #6200,@#$STOP ;;COMPARISON VALUE FOR 18 BIT MAPPING BK001
941 CMP (SP)+,(SP)+ ;;CLEAN OFF STACK BK001
942 CLR @#$MAP ;;MAKE SURE MAP INDICATOR TURNED OFF BK001
943 BR $NOMAP
  
```

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	@#\$STOP,(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	\$SIZE	::NO-SKIP	BK001
980		CLR	@#SR3	::TURN OFF MAP	BK001
981		BR	\$SIZE		
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	\$CORE:	MOV	#\$CROUT,@#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	\$CROUT:	SUB	#4000,R1		
996	\$SIZE:	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          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      .SSIZE
  
```

BK001

```

1026 .MACRO .$TYPE
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 #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1052 BEQ 62$ ;;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
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:   ;;
1121          .IIF EQ .-$TKS1,$TKS1: .WORD 177560      ;;TTY KDB STATUS      :MJD001
1122          .IIF NDF $TKB1,$TKB1:   ;;
1123          .IIF EQ .-$TKB1,$TKB1: .WORD 177562      ;;TTY KBD BUFFER        :MJD001
1124          .IIF NDF $XON,$XON = 21
1125          .IIF NDF $XOFF,$XOFF = 23
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:   ;;
1130          .IIF EQ .-$TPS1,$TPS1: .WORD 177564      ;;TTY PRINTER STATUS REG. ADDRESS
1131          .IIF NDF $TPB1,$TPB1:   ;;
1132          .IIF EQ .-$TPB1,$TPB1: .WORD 177566      ;;TTY PRINTER BUFFER REG. ADDRESS
1133          .IIF NDF $NULL,$NULL:   ;;
1134          .IIF EQ .-$NULL,$NULL: .BYTE 0          ;;CONTAINS NULL CHARACTER FOR FILLS
1135          .IIF NDF $FILLS,$FILLS: ;;
1136          .IIF EQ .-$FILLS,$FILLS: .BYTE 2        ;;CONTAINS # OF FILLER CHARACTERS RE
1137          .IIF NDF $FILLC,$FILLC: ;;
1138          .IIF EQ .-$FILLC,$FILLC: .BYTE 12       ;;INSERT FILL CHARS. AFTER A 'LINE F
1139          .IIF NDF $TPFLG1,$TPFLG1:

```

```
1140 .IIF EQ .-STPFLG1,STPFLG1: .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 .STYPOCT
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 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1156 ;*CALL:
1157 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1158 ;*      TYPOS      ;;CALL FOR TYPEOUT
1159 ;*      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1160 ;*      .BYTE      M      ;;M=1 OR 0
1161 ;*                               ;;1=TYPE LEADING ZEROS
1162 ;*                               ;;0=SUPPRESS LEADING ZEROS
1163 ;*
1164 ;*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1165 ;*$TYPOS OR $TYPOC
1166 ;*CALL:
1167 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1168 ;*      STYPON      ;;CALL FOR TYPEOUT
1169 ;*
1170 ;*$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       $STYPON
1179 $TYPOC: MOV      #1, $OFILL        ;;SET THE ZERO FILL SWITCH
1180         MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
1181 $STYPON: 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 ;*
1244 ;*
1245 $TYPDS: MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK
1246 TYPDS ;:GO TO THE ROUTINE
1247 $TYPDS: PUSH <R0,R1,R2,R3,R4,R5>
1248 MOV #20200,-(SP) ;:SET BLANK SWITCH AND SIGN
1249 MOV 22(SP),R5 ;:GET THE INPUT NUMBER
1250 BPL 1$ ;:BR IF INPUT IS POS.
1251 NEG R5 ;:MAKE THE BINARY NUMBER POS.
1252 MOV #'-,1(SP) ;:MAKE THE ASCII NUMBER NEG.
1253 CLR R0 ;:ZERO THE CONSTANTS INDEX
1254 MOV PC,R3 ;:FETCH PC FOR ADDRESSING
1255 ADD #SDBLK-.,R3 ;:SETUP THE OUTPUT POINTER - R3 POINTS TO SDBLK
1256 ;:TABLE
1257 MOV #',(R3)+ ;:SET THE FIRST CHARACTER TO A BLANK
1258 CLR R2 ;:CLEAR THE BCD NUMBER
1259 MOV PC,R4 ;:USE PC FOR LOCATING TABLE
1260 ADD #SDTBL-.,R4 ;:R4 NOW POINTS TO TABLE
1261 ADD R0,R4 ;:INDEX INTO TABLE
1262 MOV (R4),R1 ;:GET THE CONSTANT
1263 SUB R1,R5 ;:FORM THIS BCD DIGIT
1264 BLT 4$ ;:BR IF DONE
1265 INC R2 ;:INCREASE THE BCD DIGIT BY 1
1266 BR 3$
1267 4$: ADD R1,R5 ;:ADD BACK THE CONSTANT
1268 TST R2 ;:CHECK IF BCD DIGIT=0
1269 BNE 5$ ;:FALL THROUGH IF 0
1270 TSTB (SP) ;:STILL DOING LEADING 0'S?
1271 BMI 7$ ;:BR IF YES
1272 ASLB (SP) ;:MSD?
1273 BCC 6$ ;:BR IF NO
1274 MOV 1(SP),-1(R3) ;:YES--SET THE SIGN
1275 BIS #'0,R2 ;:MAKE THE BCD DIGIT ASCII
1276 BIS #' ,R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
1277 MOV R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
1278 TST (R0)+ ;:JUST INCREMENTING
1279 CMP R0,#10 ;:CHECK THE TABLE INDEX
1280 BLT 2$ ;:GO DO THE NEXT DIGIT
1281 BGT 8$ ;:GO TO EXIT
1282 MOV R5,R2 ;:GET THE LSD
1283 BR 6$ ;:GO CHANGE TO ASCII
1284 TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
1285 BPL 9$ ;:BR IF NO
1286 MOV -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
1287 CLRB (R3) ;:SET THE TERMINATOR
1288 POP <R5,R4,R3,R2,R1,R0>
1289 MOV PC,-(SP) ;:FETCH ADDRESS OF NUMBER TO BE TYPED USING
1290 ADD #SDBLK-.,(SP) ;:POSITION INDEPENDENT CODE
;:STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
;:PRINTED

```

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



```

1304 .MACRO $TRAP 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
  
```


Address	Hex	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	
---------	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

```

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$:      MOV      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
000764 001420      BEQ      3$      ;;IF ZERO GET OUT
000766 122716 000060      CMP      #'0,(SP)      ;;MAKE SURE THIS CHARACTER
000772 003026      BGT      4$      ;;IS AN OCTAL DIGIT
000774 122716 000067      CMP      #'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$: RDLIN      ;;READ AN ASCII LINE
001110 012600      MOV      (SP)+,R0      ;;ADDRESS OF 1ST CHAR.
001112 010067 000120      MOV      R0,6$      ;;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 NUMBER 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$: MOV      (SP)+,12(SP)      ;;SAVE THE RESULT
001220 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
001222 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
001224 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
001226 000002      RTI      ;;RETURN
001230 005726      5$: TST      (SP)+      ;;CLEAN PARTIAL NUMBER FROM STACK
001232 105010      CLRB     (R0)      ;;SET A TERMINATOR
001234 104401      TYPE      ;;TYPE THE INPUT UP TO BAD CHAR.
001236 000000      6$: .WORD     0      ;;POINTER GOES HERE
001240 104401 000670      TYPE      ' ','CR' & 'LF'
001244 000720      BR      1$      ;;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


```

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      .WORD      200      ::SET TO USE MEMORY MANAGEMENT
001334 100135      BPL       $CORE      ::BR IF NO
001336 012737 001622 000004 MOV      #SKTNEX,@#ERRVEC ::SET FOR TIMEOUT
001344 005737 177572      TST      @#SR0      ::KT11 ARE YOU THERE?
001350 052767 100000 177754 BIS      #100000,$KT11      ::YES--SET KT11 KEY
001356 012737 001406 000004 MOV      #100$,@#ERRVEC ::SET FOR TIMEOUT
001364 005737 172516      TST      @#172516      ::Q-BUS MAP ARE YOU THERE?
001370 012767 000200 000026 MOV      #200,$MAP      ::TURN ON MAP INDICATOR
001376 012767 176200 000022 MOV      #176200,$STOP      ::END OF 2M OF MEMORY
001404 000411      BR       $MAPRG      ::GO SET UP MAP REGISTERS
001406 012737 006200 001426 100$: MOV      #6200,@#$STOP      ::COMPARISON VALUE FOR 18 BIT MAPPING
001414 022626      CMP      (SP)+,(SP)+      ::CLEAN OFF STACK
001416 005037 001424      CLR      @#$MAP      ::MAKE SURE MAP INDICATOR TURNED OFF
001422 000402      BR       $NOMAP      ::BK001
001424 000000      $MAP: .WORD      0      ::=200 IF MAP PRESENT
001426 000000      $STOP: .WORD      0      ::FILLED WITH APPROPRIATE COMPARISON VALUE
001430 005046      CLR      -(SP)      ::INITIALIZE FOR 'PAR' LOADING
001432 012702 172340      MOV      #KIPAR0,R2      ::ADDRESS OF FIRST 'PAR'
001436 012703 000010      MOV      #^D8,R3      ::LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
001442 012762 077406 177740 1$: 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$,@#ERRVEC      ::CATCH TIMEOUT IF NO SR3
001474 012737 000020 172516 MOV      #20,@#SR3      ::ENABLE 22 BIT MODE AND UNIBUS MAP
001502 000401      BR       3$      ::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      @#$MAP      ::IS MAP THERE?
001524 100006      BPL       4$      ::NO-SKIP
001526 012737 001602 000114 MOV      #SMMOUT,@#114      ::SET UP MEMORY ERROR VECTOR
001534 013737 000006 000116 MOV      @#ERRVEC+2,@#116 ::LOCK OUT INTERRUPTS
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      SKTOUT: MOV      (R2),R2      ::GET LAST BANK+1
001562 005037 177572      CLR      @#SR0      ::TURN OFF MEMORY MANAGEMENT
001566 105737 001424      TSTB      @#$MAP      ::IS MAP THERE?
001572 100034      BPL       $SIZEX      ::NO-SKIP
001574 005037 172516      CLR      @#SR3      ::TURN OFF MAP
001600 000431      BR       $SIZEX
001602 013704 177744      SMMOUT: MOV      @#177744,R4      ::SAVE MEMORY ERROR REGISTER
001606 010437 177744      MOV      R4,@#177744      ::CLEAR BITS IN REGISTER
001612 032704 000001      BIT      #1,R4      ::MEMORY TIMEOUT?
001616 001360      BNE      SKTOUT      ::YES-EXIT
001620 000002      RTI      ::MUST BE PARITY ERROR-IGNORE IT
001622 042767 100000 177502 SKTNEX: BIC      #100000,$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	\$SIZE: SUB	#40,R2	::DROP BACK
001670	010006		MOV	R0,SP	::RESTORE THE STACK
001672	012637	000006	MOV	(SP)+,@#ERRVEC+2	::RESTORE ERROR VECTOR
001676	012637	000004	MOV	(SP)+,@#ERRVEC	
001702	012637	000116	MOV	(SP)+,@#116	::RESTORE MEMORY ERROR VECTOR
001706	012637	000114	MOV	(SP)+,@#114	
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 ROUTINE FETCH
1441 001742
1442 001742
1443 001742 104406 RDCHR ;FETCH ANSWER
1444
1445 001744 012637 010342 POP @#ECHO ;FETCH CHARACTER
1446 001744 012637 010342 MOV (SP)+,@#ECHO ;:POP STACK INTO @#ECHO
1447 001750 104401 010342 TYPE ,ECHO ;ECHO CHARACTER
1448
1449 001754 104401 016155 TYPE ,CRLF1 ;PRINT <CR><LF>
1450
1451 001760 ENDRTN
1452 001760
1453 001760 000207
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 012637 010342 POP @#ECHO ;FETCH BITS 0 - 15
1465 001764 (SP)+,@#ECHO ;;POP STACK INTO @#ECHO
1466 001770 IF #17777 SETIN @#ECHO THEN ;CHECK FOR 4K
1467 001770 032737 017777 010342 BIT #17777,@#ECHO
1468 001776 001403 BEQ 50002$ ;BOUNDARY
1469 ;NOT LEGAL ADDRESS
1470 002004 ELSE
1471 002004 000432 BR 50003$
1472 002006 013737 010342 010344 LET @#SAVLOW := @#ECHO ;SAVE BITS 0 - 15 OF ADDRESS
1473 002006 MOV @#ECHO,@#SAVLOW
1474 002014 013737 001066 010346 LET @#SAVHI := @#$HIOCT ;SAVE BITS 16 - 22 OF ADDRESS
1475 002014 MOV @#$HIOCT,@#SAVHI
1476 ;BUMP TO NEXT 4K IN ORDER TO DETERMINE IF THERE IS ENOUGH
1477 ;MEMORY FOR THE PROGRAM AND TESTING
1478
1479 002022 LET @#ECHO := @#ECHO + #20000 ;ADD 4K
1480 002022 062737 020000 010342 ADD #20000,@#ECHO
1481 002030 IF #1 SETIN CONTROL THEN
1482 002030 032767 000001 014122 BIT #1,CONTROL
1483 002036 001402 BEQ 50004$
1484 LET @#$HIOCT := @#$HIOCT + CARRY ;ADD OVERFLOW
1485 002040 005537 001066 ADC @#$HIOCT
1486
1487 002044 ENDIF
1488 002044 004767 000024 CALL ADDCHK
1489 002050 IF #1 SETIN CONTROL THEN ;MAPPED?
1490 002050 032767 000001 014102 BIT #1,CONTROL
1491 002056 001403 BEQ 50005$
1492 LET R3 := R3 - #200 ;REMOVE 4K TEST INCREMENT
1493 002064 162703 000200 SUB #200,R3
1494 002064 000402 ELSE BR 50006$

```

ENDIF

ENDIF

ENDRTN

```

50005$:
LET R3 := R3 - #20000 ;REMOVE 4K TEST INCREMENT
SUB #20000,R3

```

50005\$:

50006\$:

50003\$:

```
50000$:  
50001$:  
RTS      PC
```

50001\$:

RTS	PC
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100


```

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 ROUTINE ADDCHK
1508
1509                                     IF #1 SETIN CONTROL THEN
1510                                     ;DO WE HAVE MEM. MAN
1511                                     BIT #1,CONTROL
1512                                     BEQ 50002$
1513                                     ;YES
1514                                     ;MAKE ADDRESS IN ECHO AND $HIOCT
1515                                     MOV #1,R3
1516                                     BR 50003$
1517                                     50004$:
1518                                     INC R3
1519                                     50003$:
1520                                     CMP R3,#3
1521                                     BGT 50005$
1522                                     ;COMPATIBLE WITH PAR
1523
1524                                     LET @#ECHO := @#ECHO ROTATE 1
1525                                     ROL @#ECHO
1526
1527                                     LET @#$HIOCT := @#$HIOCT ROTATE 1
1528                                     ROL @#$HIOCT
1529
1530                                     ENDINC
1531                                     BR 50004$
1532                                     50005$:
1533
1534                                     INCR R3 FROM #1 TO #7 BY #1
1535                                     ;NOW HIOCT WILL TAKE LOWER BITS
1536                                     MOV #1,R3
1537                                     BR 50006$
1538                                     50007$:
1539                                     INC R3
1540                                     50006$:
1541                                     CMP R3,#7
1542                                     BGT 50010$
1543                                     ;AND PUT THEM INTO UPPER BITS
1544
1545                                     LET @#$HIOCT := @#$HIOCT SHIFT 1
1546                                     ASL @#$HIOCT
1547
1548                                     ENDINC
1549                                     BR 50007$
1550                                     50010$:
1551
1552                                     LET R3 := @#$HIOCT
1553                                     MOV @#$HIOCT,R3
1554
1555                                     ELSE
1556                                     ;NO MEMORY MANAGEMENT
1557                                     BR 50011$
1558                                     50002$:
1559
1560                                     LET R3 := @#ECHO
1561                                     ;FETCH STARTING ADDRESS
1562                                     MOV @#ECHO,R3

```

```

1532
1533 002172          IF @#$HIOCT NE #0 THEN          ;ADDRESS TO HIGH
      002172 005737 001066                          TST @#$HIOCT
      002176 001403                          BEQ 50012$
1534
1535 002200 104401 010224          TYPE ,OUTMEM      ;TELL OPERATOR THAT ADDRESS
1536                                     ;OUT OF MEMORY
1537
1538 002204          INLINE <BR 1$>                  ;EXIT FROM ROUTINE
      002204 000407                          BR 1$
1539
1540 002206          ENDIF                          50012$:
      002206
1541
1542 002206          ENDIF                          50011$:
      002206
1543
1544 002206          IF R3 HI ENDADD THEN              ;OUTSIDE OF MEMORY
      002206 020367 013762                          CMP R3,ENDADD
      002212 101403                          BLOS 50013$
1545                                     ;IF R3 IS GREATER THAN R2
1546 002214 104401 010224          TYPE ,OUTMEM      ;PRINT STARTING ADDRESS OUTSIDE
1547                                     ;OF ADDRESS RANGE
1548
1549 002220          ELSE
      002220 000401
      002222
1550
1551 002222          LET R4 := R4 + #1                  ;INDICATE OK
      002222 005204                          INC R4
1552
1553 002224          ENDIF                          50014$:
      002224
1554
1555 002224          1$: ENDRTN
      002224
      002224 000207
      002224

```

50000\$:
50001\$:
RTS PC

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 BETWEEN'<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	HELPH:	.ASCII	<12><15>'DO YOU WANT A HELP FILE (L) N? '	
1593	006236	012	015	101	ILLADD:	.ASCII	<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	.ASCII	:	'NON-VOLATILE MEMORIES'<12><15>	
1596	006404	012	015	111	BANK0:	.ASCII	<12><15>'IS THERE NON-VOLATILE MEMORY AT ADDRESS 0 (L) Y? '	
1597	006470	113	040	117	K:	.ASCII	'K OF MEMORY'<12><15>	
1598	006506	012	015	124	MAPPED:	.ASCII	<12><15>'THIS IS A MAPPED SYSTEM (MEMORY MANAGEMENT) WITH '	
1599	006572	012	015	124	UNMAP:	.ASCII	<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	.ASCII	:	'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	.ASCII	:	'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	.ASCII	:	'IN THIS SYSTEM (L) Y? '	
1606	007237	012	015	110	NOBLK:	.ASCII	<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	.ASCII	:	'SET ENABLE/FRONT PANEL SWITCH TO HALT POSITION, APPLY POWER'<12><15>	
1610	007477	074	103	122	CR1:	.ASCII	'<CR>'<12><15>	
1611	007506	124	131	120	PAR0:	.ASCII	'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>	


```

1614 007624      100      067      067      .ASCII 'a772316/XXXXXX 77406<CR>'<12><15>
1615 007657      100      067      067      .ASCII 'a777572/XXXXXX 1<CR>'<12><15>
1616 007713      100      067      067      MMR3: .ASCII 'a772516/XXXXXX 20<CR>'<12><15>
1617 007747      012      123      105      ENABLE: .ASCII <12>'SET ENABLE/HALT FRONT PANEL SWITCH TO ENABLE '
1618 010025      120      117      123      .ASCII 'POSITION AND TYPE:'<12><12><15>
1619 010052      100      044      067      .ASCII 'a$7/XXXXXX '
1620 010066      074      103      122      RHELP: .ASCII '<CR>'<12><15>
1621 010074      100      120      012      .ASCII 'aP'<12><15>
1622 010100      040      040      040      .ASCII '
1623 010152      012      012      015      NOT4K: .ASCII <12><12><15>'ILLEGAL ADDRESS - NOT AT 4K BOUNDARY'<12><15>
1624 010224      012      012      015      OUTMEM: .ASCII <12><12><15>'4K ADDRESS OR NUMBER OF 4K BANKS ARE OUT OF '
1625 010302      040      101      104      .ASCII ' ADDRESS RANGE OF THIS SYSTEM'<12><15>
1626                                     .EVEN

```

```

1627 010342      000000      ECHO: .WORD 0
1628 010344      000000      SAVLOW: .WORD 0
1629 010346      000000      SAVHI: .WORD 0

```

```

1630
1631      006014      DIF1 = ENDPR-Q4      ;THE NUMBER TO BUMP THE POINTER TO BY PASS THE
1632                                     ;PROGRAM
1633
1634
1635

```

```

1636      ;THIS ROUTINE BEGINS THE DIAGNOSTIC, IT WILL DETERMINE IF THIS MACHINE
1637      ;HAS MEMORY MANAGEMENT OR NOT, CALL THE SIZING ROUTINE, ASK THE MENU
1638      ;QUESTIONS, MOVE THE PROGRAM IF NEED BE, WRITE A BACKGROUND OF 125252 THROUGH
1639      ;DUT MEMORY, PRINT RESTART HELP FILE, IF APPLICABLE AND INSTRUCT
1640      ;OPERATOR TO POWER DOWN THE SYSTEM.
1641

```

```

1642 010350      ROUTINE START1
1643 010350
1644 010350      000005      INLINE <RESET>      ;CLEAR THE WORLD
1645                                     RESET
1646 010352      012706      017644      LET      SP := #SPINIT      ;FETCH STACK ADDRESS
1647                                     MOV      #SPINIT,SP
1648 010356      004767      005646      CALL      TRAPCT      ;TRAP CATCHER
1649                                     JSR      PC,TRAPCT
1650 010362      005067      005572      LET      CONTROL := #0      ;CLEAR OUT CONTROL WORD
1651                                     CLR      CONTROL
1652 010366      005067      005576      LET      PPOINT := #0      ;PROGRAM POINTER.
1653                                     CLR      PPOINT
1654 010372      104401      006302      TYPE      ,HELLO      ;PRINT WHO WE ARE
1655
1656 010376      004767      170644      CALL      $SIZE      ;THIS ROUTINE DETERMINES, MEMORY MANAGEMENT,
1657                                     JSR      PC,$SIZE
1658                                     ; 16, 18, AND 22 BIT ADDRESS AND MEMORY SIZE
1659 010402      005767      170724      IF      $KT11 LT #0 THEN      ;SET MEMORY MANAGEMENT BIT
1660                                     TST      $KT11
1661 010410      LET      CONTROL := $MAP      ;THIS SYSTEM HAS
                                     BGE      50002$

```


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	010460				LET (R1)+ := #0		;4K
1681	010460	005021				CLR	(R1)+
1682	010462				ENDINC		
1683	010462	000772				BR	50004\$
1684	010464						50005\$:
1685	010464				LET (R1) := #177600		;SET UP I/O PAGE IN PAR7
1686	010464	012711	177600			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	010510				LET (R1)+ := #77406		;4K
1698	010510	012721	077406			MOV	#77406,(R1)+
1699	010514				ENDINC		
1700	010514	000771				BR	50007\$
1701	010516						50010\$:
1702	010516				LET @#SRO := #1		;SET MEMORY MANAGEMENT BIT
1703	010516	012737	000001	177572		MOV	#1,@#SRO
1704	010524				IFB	\$MAP LT #0 THEN	;DO WE HAVE A 22 ADDRESS BIT
1705	010524	105767	170674			TSTB	\$MAP

```

1690 010530 002003
1691
1692 010532 012737 000020 172516
1693
1694 010540
1695 010540
1696 010540 000406
1697 010542
1698 010542 016767 171170 005424
1699 010550 062767 000002 005416
1700 010556
1701 010556
1702 010556 032767 000001 005374
1703 010564 001403
1704 010566 104401 006506
1705
1706
1707 010572
1708 010572 000402
1709 010574 104401 006572
1710
1711
1712 010600
1713 010600
1714 010600 062767 000040 171132
1715
1716 010606 006267 171126
1717
1718 010612 042767 100000 171120
1719
1720 010620 006267 171114
1721 010624 006267 171110
1722 010630 006267 171104
1723 010634 006267 171100
1724 010640 016746 171074
1725 010644 104405
1726 010646 104401 006470

;MACHINE?
BGE 50011$
;SETUP FOR 22 BIT ADDRESSING
LET @#SR3 := #20
MOV #20,@#SR3
ENDIF
50011$:
ELSE
BR 50012$
50002$:
LET ENDADD := $LSTAD + #2 ;FETCH LAST VIRTUAL ADDRESS
MOV $LSTAD,ENDADD
ADD #2,ENDADD
ENDIF
50012$:
IF #1 SETIN CONTROL THEN ;TELL OPERATOR WHAT KIND OF SYSTEM
BIT #1,CONTROL
BEQ 50013$
TYPE ,MAPPED ;TYPE THIS IS A MAPPED SYSTEM
;(MEMORY MANAGEMENT) WITH
ELSE
BR 50014$
50013$:
TYPE ,UNMAP ;TYPE THIS IS AN UN-MAPPED SYSTEM
;(NO MEMORY MANAGEMENT) WITH
ENDIF
50014$:
LET $LSTBK := $LSTBK + #40 ;FETCH LAST BANK ADDRESS WITH OUT MEMORY
ADD #40,$LSTBK
LET $LSTBK := $LSTBK SHIFT -1 ;CORRECT LSTBK FOR PRINTING
ASR $LSTBK
LET $LSTBK := $LSTBK CLR.BY #100000 ;CLEAR SIGN BIT
BIC #100000,$LSTBK
LET $LSTBK := $LSTBK SHIFT -4 ;CONTINUE CORRECTING
ASR $LSTBK
ASR $LSTBK
ASR $LSTBK
ASR $LSTBK
TYPDEC $LSTBK ;TELL OPERATOR HOW MUCH MEMORY WE FOUND
MOV $LSTBK,-(SP) ;;SAVE $LSTBK FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
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      @#FETCH      ;FETCH ANSWER
                                JSR      PC,@#FETCH
1729
1730 010662 023727 010342 000131  IF @#ECHO EQ #'Y THEN      ;PRINT HELP FILE IF YES
                                CMP      @#ECHO, #'Y
                                BNE      50015$
1731
1732 010672 104401 002226      TYPE      ,HELPF      ;PRINT HELP FILE
1733
1734 010676      ENDIF
                                50015$:
1735
1736 010676 104401 006404      TYPE      ,BANK0      ;PRINT IS THERE NON-VOLATILE MEMORY AT ADDRESS 0
1737
1738 010702 004737 001742      CALL      @#FETCH      ;FETCH ANSWER
                                JSR      PC,@#FETCH
1739
1740 010706 023727 010342 000015  IF @#ECHO EQ #15 THEN      ;LOOK FOR DEFAULT CHARACTER
                                CMP      @#ECHO, #15
                                BNE      50016$
1741
1742 010716 012737 000131 010342      LET @#ECHO := #'Y      ;CHANGE DEFAULT CHARACTER
                                MOV      #'Y,@#ECHO
1743
1744 010724      ENDIF
                                50016$:
1745
1746 010724 023727 010342 000131  IF @#ECHO NE #'Y THEN      ;IF NOT AT BANK0 THEN WHERE
                                CMP      @#ECHO, #'Y
                                BEQ      50017$
1747
1748 010734 005004      LET R4 := #0      ;CORRECT ADDRESS FLAG
                                CLR      R4
1749
1750 010736      LOOP
                                50020$:
1751
1752 010736 104401 006664      TYPE      ,WHERE      ;PRINT WHAT IS THE STARTING ADDRESS
                                ;OF ONE OF THE NON-VOLATILE MEMORY
                                ;MODULES.
1753
1754
1755
1756 010742 004737 001762      CALL      @#FETADD      ;GO FETCH ADDRESS TO
                                JSR      PC,@#FETADD
1757
1758                                ;RELOCATE PROGRAM
1759 010746 005704 001001      EXIF      R4 NE #0
                                TST      R4
                                BNE      50021$
1760
1761 010752 000771      ENDLOOP
                                BR      50020$
1762                                50021$:
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 - #\$TYPOS	;CALCULATE OFFSET BETWEEN TABLE AND
	011134	010003			MOV R0,R3
	011136	162703	017166		SUB #\$TYPOS,R3
1840					; \$TYPOS 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 - #\$TYPON	;CALCULATE OFFSET BETWEEN TABLE AND
	011152	010003			MOV R0,R3
	011154	162703	017226		SUB #\$TYPON,R3
1849					; \$TYPON 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 - #\$TYPDS	;CALCULATE OFFSET BETWEEN TABLE AND
	011170	010003			MOV R0,R3
	011172	162703	016714		SUB #\$TYPDS,R3
1858					; \$TYPDS 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 @#FETCH	;FETCH CHARACTER.
	011212	004737	001742		JSR PC,@#FETCH


```

1870
1871 011216 023727 010342 000015 IF @#ECHO EQ #15 THEN ;LOOK FOR DEFAULT CHARACTER
      011216 001003                                CMP @#ECHO,#15
      011224                                BNE 50027$
1872
1873 011226 012737 000131 010342 LET @#ECHO := #'Y ;CHANGE DEFAULT CHARACTER
      011226                                MOV #'Y,@#ECHO
1874
1875 011234 011234 ENDIF
      011234                                50027$:
1876
1877 011234 023727 010342 000131 IF @#ECHO NE #'Y THEN ;IF NOT ENTIRE SYSTEM FIND OUT WHERE
      011234 001461                                CMP @#ECHO,#'Y
      011242                                BEQ 50030$
1878
1879 011244 005004 LET R4 := #0 ;ILLEGAL ADDRESS FLAG
      011244                                CLR R4
1880
1881 011246 011246 LOOP
      011246                                50031$:
1882
1883 011246 104401 007006 TYPE ,WHERE1 ;PRINT: WHAT IS THE ADDRESS OF THE
1884                                ;NON-VOLATILE MEMORY MEMORY UNDER TEST
1885
1886 011252 004737 001762 CALL @#FETADD ;GO FETCH ADDRESS TO BE TESTED
      011252                                JSR PC,@#FETADD
1887
1888 011256 005704 001001 EXIF R4 NE #0
      011256                                TST R4
      011260                                BNE 50032$
1889
1890 011262 000771 ENDLOOP
      011262                                BR 50031$
      011264                                50032$:
1891
1892 011264 005004 LET R4 := #0 ;ILLEGAL ADDRESS FLAG
      011264                                CLR R4
1893
1894 011266 010367 004676 LET PPOINT := R3 ;NEW PROGRAM POINTER
      011266                                MOV R3,PPOINT
1895
1896 011272 011272 LOOP
      011272                                50033$:
1897
1898 011272 104401 007237 TYPE ,NOBLK ;PRINT WHAT IS THE NUMBER OF 4K
1899                                ;BANKS TO BE TESTED
1900
1901 011276 104411 RDDEC ;FETCH DECIMAL NUMBER
1902
1903 011300 012600 MOV (SP)+,R0 POP R0 ;:POP STACK INTO R0
      011300
1904
1905 011302 005003 LET R3 := #0 ;CLEAR ADD REG.
      011302                                CLR R3
1906
1907 011304 005002 LET R2 := #0 ;CLEAR ADD REG.
      011304                                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				;NUMBER OF BANKS TO BE TESTED BY 4K	
1913					
1914	011326			LET R2 := R2 + #20000	;ADD 4K
	011326	062702	020000	ADD	#20000,R2
1915					
1916	011332			LET R3 := R3 + CARRY	;ADD OVERFLOW TO
	011332	005503		ADC	R3
1917					;UPPER WORD
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
1928					;ADDRESS
1929					
1930	011354			LET @#\$HIOCT := R3	;STORE AWAY HIGHER
	011354	010337	001066	MOV	R3,@#\$HIOCT
1931					;ADDRESS
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	


```

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

```

```

1979 011472
1980 011472
1981 011472
1982 011472 104401 007747
1983
1984
1985
1986
1987 011476 010701 000372
1988 011476 062701 000372
1989 011500
1990 011504 032767 000001 004446
1991 011504 001404
1992 011512
1993 011514 042701 160000
1994 011514
1995 011520 052701 040000
1996 011520
1997 011524
1998 011524
1999 011524 010146
2000 011524 104403
2001 011526 006
2002 011530 000
2003 011531
2004 011532 104401 010066
2005
2006
2007 011536 032767 000001 004414
2008 011536 001414
2009 011546 016737 004416 172342
2010 011546
2011 011554 005767 004410
2012 011554 001003
2013 011562 012700 020400
2014 011562
2015
2016 011566

ENDIF
TYPE ,ENABLE ;PRINT SET SWITCH TO ENABLE AND TYPE
;a777707/

.ENABL LSB
LET R1 := PC + #PWRUP-. ;FETCH RESTART ADDRESS
MOV PC,R1
ADD #PWRUP-.,R1

IF #1 SETIN CONTROL THEN ;ARE WE A MAPPED SYSTEM
BIT #1,CONTROL
BEQ 50046$
;YES

LET R1 := R1 CLR.BY #160000 ;CLEAR OUT PROGRAM ADDRESS
BIC #160000,R1
;REGISTER PRINTERS.

LET R1 := R1 SET.BY #40000 ;LOOK AT PAR2
BIS #40000,R1

ENDIF

50046$:
TYPOCS R1 ;PRINT RESTART ADDRESS
MOV R1,-(SP) ;;SAVE R1 FOR TYPEOUT
TYPOS ;;GO TYPE--OCTAL ASCII
.BYTE 6 ;;TYPE 6 DIGITS
.BYTE 0 ;;SUPPRESS LEADING ZEROS

TYPE ,RHELP ;PRINT REST OF HELP FILE

.DSABL LSB
;WRITE BACKGROUND THROUGH MEMORY

IF #1 SETIN CONTROL THEN ;ARE WE A MAPPED SYSTEM
BIT #1,CONTROL
BEQ 50047$

LET @#KIPAR1 := PPOINT ;FETCH TEST STARTING POINT
MOV PPOINT,@#KIPAR1

IF PPOINT EQ #0 THEN ;ARE WE IN BANK 0
TST PPOINT
BNE 50050$

LET R0 := #20400 ;POINT TO ADDRESS 400, PROGRAM
MOV #20400,R0
;ADDRESS REGISTER 1

ELSE

```



```

011566 000402
011570
2017
2018 011570 012700 020000
011570
2019
2020
2021 011574
011574
2022
2023 011574
011574 000410
011576
2024
2025 011576
011576 005767 004366
011602 001003
2026
2027 011604
011604 012700 000400
2028
2029 011610
011610 000402
011612
2030
2031 011612
011612 016700 004352
2032
2033 011616
011616
2034
2035 011616
011616
2036
2037 011616
011616 005004
2038
2039
2040
2041 011620
011620 010701
011622 062701 004360
2042
2043 011626
011626 010137 000004
2044
2045
2046 011632
011632
2047
2048 011632
011632 012720 125252
2049
2050
2051 011636
011636 032767 000001 004314
011644 001436

                                BR      50051$
                                50050$:
                                ;POINT TO ADDRESS 0, PROGRAM
                                MOV      #20000,R0
                                ;ADDRESS REGISTER 1
                                50051$:
                                BR      50052$
                                50047$:
                                ;ARE WE IN BANK 0
                                TST      PPOINT
                                BNE      50053$
                                LET      R0 := #400
                                ;START AFTER TRAP LOCATIONS
                                MOV      #400,R0
                                ELSE
                                BR      50054$
                                50053$:
                                LET      R0 := PPOINT
                                ;START AT TEST BANK
                                MOV      PPOINT,R0
                                ENDIF
                                50054$:
                                50052$:
                                ;CLEAR EXIT FROM LOOP FLAG
                                CLR      R4
                                .ENABL  LSB
                                LET      R1 := PC + #ROMMD-.
                                ;FETCH ROMMD ROUTINE
                                MOV      PC,R1
                                ADD      #ROMMD-.,R1
                                LET      @#4 := R1
                                ;LOAD ADDRESS INTO 4 INCASE WE
                                MOV      R1,@#4
                                ;HIT A ROM MODULE
                                50055$:
                                LET      (R0)+ := #125252
                                ;WRITE DATA PATTERN
                                MOV      #125252,(R0)+
                                ;THROUGH OUT MEMORY
                                IF #1 SETIN CONTROL THEN
                                ;ARE WE A MAPPED SYSTEM
                                BIT      #1,CONTROL
                                BEQ      50057$

```

2052
 2053 011646 023737 172342 172344
 011646 001014
 011654
 2054
 2055
 2056 011656 010701 000026
 011656 162701
 011660
 2057
 2058
 2059 011664 042701 160000
 011664
 2060
 2061
 2062 011670 010003 160000
 011670 042703
 011672
 2063
 2064
 2065 011676 020301
 011676 001002
 011700
 2066
 2067
 2068 011702 062700 006014
 011702
 2069
 2070
 2071 011706
 011706
 2072 011706
 011706
 2073
 2074 011706 020027 040000
 011706 001005
 011712
 2075
 2076
 2077 011714 012700 020000
 011714
 2078
 2079
 2080 011720 062737 000200 172342
 011720
 011726
 2081
 2082 011726
 011726
 2083
 2084 011726 023767 172342 004240
 011726 001001
 011734
 2085
 2086
 2087 011736 005204
 011736
 2088

```

IF @#KIPAR1 EQ @#KIPAR2 THEN      ;WE IN SAME BLOCK OF
CMP                                @#KIPAR1,@#KIPAR2
BNE                                50060$
;MEMORY

LET R1 := PC - #-Q4                ;DISTANCE BETWEEN
MOV                                PC,R1
SUB                                #-Q4,R1
;BEGINNING OF PROGRAM AND HERE

LET R1 := R1 CLR.BY #160000         ;CLEAR OUT PAR
BIC                                #160000,R1
;POINTER

LET R3 := R0 CLR.BY #160000         ;CLEAR OUT PAR
MOV                                R0,R3
BIC                                #160000,R3
;POINTER

IF R3 EQ R1 THEN                    ;HAVE WE REACHED
CMP                                R3,R1
BNE                                50061$
;BEGINNING OF PROGRAM

LET R0 := R0 + #DIF1                ;HAS BUMP
ADD                                #DIF1,R0
;POINTER TO END OF PROGRAM

ENDIF

50061$:
ENDIF
50060$:

IF R0 EQ #40000 THEN                ;HAVE WE COMPLETED 4K CHUNK
CMP                                R0,#40000
BNE                                50062$
;OF MEMORY

LET R0 := #20000                    ;YES RETURN PAR1 POINTER
MOV                                #20000,R0

;BUMP TO NEXT 4K
LET @#KIPAR1 := @#KIPAR1 + #200
ADD                                #200,@#KIPAR1

ENDIF

50062$:

IF @#KIPAR1 EQ ENDADD THEN          ;HAVE WE REACHED END OF
CMP                                @#KIPAR1,ENDADD
BNE                                50063$
;MEMORY

LET R4 := R4 + #1                    ;YES SET FINISHED FLAG
INC                                R4

```



```

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$:
2109
2110 011770      ENDIF
      011770      50064$:
2111
2112 011770      EXIF R4 NE #0 ;LEAVE IF FINISHED FLAG IS SET
      011770 005704      TST R4
      011772 001001      BNE 50056$
2113
2114 011774      ENDLOOP
      011774 000716
      011776      BR 50055$
      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			LOOP		
2128	012014				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		
	012024	000773			BR 50067\$
	012026				50070\$:
2135					
2136	012026		LET CSTRPG := R3		;SAVE CHECKSUM
	012026	010367			MOV R3,CSTRPG
2137					
2138			;CALCULATE CHECKSUM OF THIS PROGRAM		
2139	012032		LET R1 := PC - #-Q4		;R1 CONTAINS BEGINNING ADDRESS
	012032	010701			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		
	012056	000774			BR 50071\$
	012060				50072\$:
2153					
2154	012060		LET (R2) := R3		;LOAD CHECK SUM INTO CHECKSUM AREA
	012060	010312			MOV R3,(R2)
2155					
2156			;TELL OPERATOR TO POWER DOWN THE SYSTEM		
2157	012062	104401	TYPE		
2158	012064	015776	.WORD PWRDWN		
2159					
2160	012066		INLINE < BR . >		;

1\$:

012066 000777

BR .

2161
2162
2163
2164

.DSABL LSB

ENDRTN

012070
012070
012070
012070 000207

50000\$:
50001\$:
RTS PC

2166				.SBTTL POWER UP ROUTINE	
2167					
2168				:THIS ROUTINE CHECKS THE ENTIRE SYSTEM FOR DATA RETENTION AND OUTPUTS A MEMORY	
2169				:MAP OF THE SYSTEM.	
2170					
2171	012072			ROUTINE PWRUP	PWRUP:
	012072				
2172					
2173	012072			LET SP := PC + #SPINIT-.	:FETCH STACK BUFFER
	012072	010706			MOV PC,SP
	012074	062706	005550		ADD #SPINIT-.,SP
2174					
2175	012100			IF #1 SETIN CONTROL THEN	:IS THIS A MAPPED SYSTEM
	012100	032767	000001 004052		BIT #1,CONTROL
	012106	001462			BEQ 50002\$
2176					:YES
2177					
2178	012110			LET @#KIPAR0 := #0	:CLEAR OUT PAR0
	012110	005037	172340		CLR @#KIPAR0
2179					
2180	012114			LET @#KIPAR1 := #0	:CLEAR OUT PAR1
	012114	005037	172342		CLR @#KIPAR1
2181					
2182	012120			LET R1 := #KIPAR3	:FETCH MEMORY MANAGEMENT PAR
	012120	012701	172346		MOV #KIPAR3,R1
2183					:LOCATION (DO NOT TOUCH PAR2 FOR
2184					:IT CONTROLS OUR PC
2185					
2186	012124			INCR R0 FROM #1 TO #4 BY #1	:CLEAR THOSE REGISTERS
	012124	012700	000001		MOV #1,R0
	012130	000401			BR 50003\$
	012132				50004\$:
	012132	005200			INC R0
	012134				50003\$:
	012134	020027	000004		CMP R0,#4
	012140	003002			BGT 50005\$
2187					
2188	012142			LET (R1)+ := #0	: CLR (R1)+
	012142	005021			
2189					
2190	012144			ENDINC	BR 50004\$
	012144	000772			50005\$:
	012146				
2191					
2192	012146			LET (R1) := #177600	:SET UP I/O PAGE
	012146	012711	177600		MOV #177600,(R1)
2193					
2194	012152			LET R1 := #KIPDR0	:FETCH PDR LOCATION
	012152	012701	172300		MOV #KIPDR0,R1
2195					
2196	012156			INCR R0 FROM #0 TO #7 BY #1	:SET THEM UP TO DO 4K INTERVALS
	012156	005000			CLR R0
	012160	000401			BR 50006\$
	012162				50007\$:
	012162	005200			INC R0
	012164				50006\$:
	012164	020027	000007		CMP R0,#7


```

2197 012170 003003
2198 012172 012721 077406
2199 012176 000771
2200 012176 000771
2201 012200 012737 000001 177572
2202 012200 012737 000001 177572
2203 012206 032767 000200 003744
2204 012214 001403
2205 012216 012737 000020 172516
2206 012216 012737 000020 172516
2207 012224 005767 003740
2208 012224 001003
2209 012232 012700 020400
2210 012232 012700 020400
2211 012236 000405
2212 012236 000405
2213 012240 016737 003724 172342
2214 012240 016737 003724 172342
2215 012246 012700 020000
2216 012246 012700 020000
2217 012252 000410
2218 012252 000410
2219 012254 005767 003710
2220 012254 001003
2221 012262 012700 000400
2222 012262 012700 000400
2223 012262 012700 000400
2224 012262 012700 000400
2225 012262 012700 000400
2226 012262 012700 000400
2227 012262 012700 000400
2228 012262 012700 000400
2229 012262 012700 000400
2230 012262 012700 000400
2231 012262 012700 000400

```

```

LET (R1)+ := #77406
ENDINC
LET @#SR0 := #1
IF #200 SETIN CONTROL THEN
    LET @#SR3 := #20
ENDIF
IF PPOINT EQ #0 THEN
    LET R0 := #20400
ELSE ;NO
    LET @#KIPAR1 := PPOINT
    LET R0 := #20000
ENDIF
ELSE
    IF PPOINT EQ #0 THEN
        LET R0 := #400
    
```

```

BGT 50010$
MOV #77406,(R1)+
BR 50007$
50010$:
;SET MEMORY MANAGEMENT IF NOT
MOV #1,@#SR0
;ALREADY DONE SO
;IS THIS A 22 ADDRESS BIT SYSTEM
BIT #200,CONTROL
BEQ 50011$
;YES, SET UP 22 BIT ADDRESSING
MOV #20,@#SR3
50011$:
;ARE WE CHECKING ENTIRE SYSTEM?
TST PPOINT
BNE 50012$
;YES
;DO NOT CHECK TRAP LOCATIONS
MOV #20400,R0
BR 50013$
50012$:
;SET UP POINTER TO CHECK THAT
MOV PPOINT,@#KIPAR1
;MODULE
;POINT TO PAR1 ADDRESS ZERO
MOV #20000,R0
50013$:
BR 50014$
50002$:
;ARE WE CHECKING ENTIRE SYSTEM?
TST PPOINT
BNE 50015$
;YES
;SKIP TRAP LOCATIONS
MOV #400,R0

```

```

2232 012266      ELSE
      012266      000402
      012270
      BR          50016$
      50015$:
2233
2234 012270      LET R0 := PPOINT      ;FETCH POINTER ADDRESS
      012270      016700  003674      MOV          PPOINT,R0
2235
2236 012274      ENDIF
      012274
      50016$:
2237
2238 012274      ENDIF
      012274
      50014$:
2239
2240      ;CALCULATE CHECKSUM OF TRAP PAGE
2241
2242 012274      LET R1 := #0
      012274      005001      ;ADDRESS OF TRAP LOCATION
      CLR          R1
2243
2244 012276      LET R3 := #0
      012276      005003      ;CHECKSUM WORK REGISTER
      CLR          R3
2245
2246 012300      LOOP
      012300
      50017$:
2247
2248 012300      LET R3 := R3 + (R1)+
      012300      062103      ;CALCULATE CHECKSUM
      ADD          (R1)+,R3
2249
2250 012302      EXIF R1 EQ #400
      012302      020127  000400      ;EXIF WHEN REACHED END OF TRAP PAGE
      012306      001401      CMP          R1,#400
      BEQ          50020$
2251
2252 012310      ENDLOOP
      012310      000773
      012312
      BR          50017$
      50020$:
2253
2254 012312      IF R3 NE CSTRPG THEN
      012312      020367  005202      ;DO CHECKSUMS AGREE?
      012316      001403      CMP          R3,CSTRPG
      BEQ          50021$
2255
2256
      ;NO
2257 012320      MOV          PUSH #4000
      012320      012746  004000      ;:PUSH #4000 ON STACK
      ;INDICATE CHECKSUM ERROR
      ELSE
      BR          50022$
      50021$:
2260
2261 012326      MOV          PUSH #0
      012326      012746  000000      ;:PUSH #0 ON STACK
      ;INDICATE NO ERROR
      ;CHECKSUM ERROR
2262
2263
      ENDIF
      50022$:
2264 012332
      012332
2265
2266 012332      CALL      TRAPCT
      012332      004767  003672      ;LOAD UP TRAP VECTORS WITH
      JSR          PC,TRAPCT
      ;TRAP CATCHERS
2267

```



```

2268
2269 012336          LET      R1 := PC - #-Q4          ;R1 CONTAINS BEGINNING ADDRESS
      012336 010701          MOV      PC,R1
      012340 162701 000506          SUB      #-Q4,R1
2270
2271          ;OF PROGRAM
2272 012344          LET      R2 := PC + #CHKSUM-.      ;R2 CONTAINS END ADDRESS OF
      012344 010702          MOV      PC,R2
      012346 062702 005154          ADD      #CHKSUM-.,R2
2273          ;PROGRAM
2274
2275 012352          LET      R3 := #0          ;R3 CONTAINS CHECK SUM
      012352 005003          CLR      R3
2276
2277 012354          LOOP
      012354
2278          50023$:
2279 012354          LET      R3 := R3 + (R1)+          ;ADD CONTENTS OF PROGRAM
      012354 062103          ADD      (R1)+,R3
2280
2281 012356          EXIF      R1 EQ R2          ;LEAVE WHEN REACHED END OF PROGRAM
      012356 020102          CMP      R1,R2
      012360 001401          BEQ      50024$
2282
2283 012362          ENDLOOP
      012362 000774          BR      50023$
      012364          50024$:
2284
2285 012364          IF (R2) NE R3 THEN          ;DO WE HAVE A CHECKSUM ERROR
      012364 021203          CMP      (R2),R3
      012366 001407          BEQ      50025$
2286          ;YES
2287
2288          ;FETCH LOCATION OF ERROR MESSAGE
2289 012370          LET R1 := PC + #ERR-.
      012370 010701          MOV      PC,R1
      012372 062701 002641          ADD      #ERR-.,R1
2290
2291          ;LOAD ADDRESS INTO TRAP INSTRUCTION
2292 012376          LET 2$ := R1
      012376 010167 000002          MOV      R1,2$
2293
2294          ;TELL OPERATOR CHECK SUM ERROR
2295 012402 104401          TYPE
2296 012404 015233          .WORD      ERR
2297          2$:
2298          ENDIF
      012406          50025$:
2299
2300 012406          LET CONTROL := CONTROL SET.BY (SP)+ ;INDICATE RESULT OF CHECK SUM
      012406 052667 003546          BIS      (SP)+,CONTROL
2301          ;OF TRAP PAGE
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$
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$:
2326
2327 012500                                LET TRCATCH := #0                ;RESUME HALT IF TRAP
012500 005067 003660                                CLR    TRCATCH
2328
2329 012504                                LET R4 := #114                ;LOAD PARITY LOCATION INTO R4
012504 012704 000114                                MOV    #114,R4
2330
2331                                ;SO THAT NEXT INSTRUCTION WILL
2332                                ;WORK ON ALL MACHINES
2333 012510                                LET (R4) := PC + #PARTRP-.                ;LOAD PARITY TRAP LOCATION WITH
012510 010714                                MOV    PC,(R4)
012512 062714 002140                                ADD    #PARTRP-.,(R4)
2334
2335                                ;ADDRESS OF PARITY TRAP ROUTINE
2336 012516                                IF #125252 EQ (R0)+ THEN                ;IS THIS LOCATION VOLATILE
012516 022720 125252                                CMP    #125252,(R0)+
012522 001016                                BNE    50031$
2337
2338                                ;IF PARITY ERROR THEN ERROR COUNT ALREADY DONE
2339 012524                                IF #2 SETIN CONTROL THEN
012524 032767 000002 003426                                BIT    #2,CONTROL

```



```

012532 001404                                BEQ      50032$
2340
2341 012534                                LET CONTROL := CONTROL CLR.BY #2
012534 042767 000002 003416                                BIC      ;CLEAR PARITY
                                                ;#2,CONTROL
                                                ;ERROR FLAG
2342
2343
2344 012542                                ELSE
012542 000405                                BR      50033$
012544                                50032$:
2345
2346                                ;INDICATE NON-VOLATILE
2347 012544                                LET CONTROL := CONTROL SET.BY #100000
012544 052767 100000 003406                                BIS      #100000,CONTROL
2348
2349 012552                                LET NONVOL := NONVOL + #1
012552 005267 003404                                ;UPDATE NONVOL COUNT
                                                INC      NONVOL
2350
2351                                ENDIF
012556                                50033$:
012556
2352
2353                                ELSE
012556 000412                                BR      50034$
012556                                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
012570 042767 000002 003362                                BIC      ;CLEAR PARITY
                                                ;#2,CONTROL
                                                ;ERROR FLAG
2359
2360
2361 012576                                ELSE
012576 000402                                BR      50036$
012600                                50035$:
2362
2363 012600                                LET VOL := VOL + #1
012600 005267 003360                                ;UPDATE VOL COUNT
                                                INC      VOL
2364
2365                                ENDIF
012604                                50036$:
012604
2366
2367                                ENDIF
012604                                50034$:
012604
2368
2369 012604                                LET R4 := #0
012604 005004                                ;FLAG FOR NON VOLATILE MEMORY HITS
                                                CLR      R4
2370
2371                                ; FAKED OUT LOOP BECAUSE LOOP IS OUT SIDE OF ADDRESS RANGE
2372
2373 012606                                STLOOP: IF #1 SETIN CONTROL THEN ;ARE WE MAPPED
012606 032767 000001 003344                                BIT      #1,CONTROL
012614 001407                                BEQ      50037$
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$:
2381
2382 012632      ELSE
012632                    BR      50041$
012634 000405                    50037$:
2383
2384 012634      IF R0 EQ ENDADD THEN ;IS IT TIME TO LEAVE?
012634 020067 003334              CMP      R0,ENDADD
012640 001002                    BNE      50042$
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 (R0)+ THEN ;DOES THIS LOCATION CONTAIN
012646 022720 125252              CMP      #125252,(R0)+
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                BIT      #2,CONTROL
      012710 032767 000002 003250                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$:
2418
2419 012722                LET VOL := VOL + #1
      012722 005267 003236                ;NO, UPDATE
                                          INC      VOL
                                          ;VOLATILE COUNT
2420
2421                                50050$:
2422 012726                ENDIF
      012726
2423
2424 012726                ENDIF
      012726                50046$:
2425
2426 012726                IF #1 SETIN CONTROL THEN
      012726 032767 000001 003224                ;ARE WE MAPPED
      012734 001421                BIT      #1,CONTROL
                                          BEQ      50051$
2427
2428 012736                IF @#KIPAR1 EQ @#KIPAR2 THEN
      012736 023737 172342 172344                ;ARE PARS IN SAME BANK?
      012744 001014                CMP      @#KIPAR1,@#KIPAR2
                                          BNE      50052$
                                          ;YES
2429
2430                                ;DISTANCE BETWEEN BEGINNING OF PROGRAM AND HERE
2431                                LET R1 := PC - #-Q4
2432 012746                MOV      PC,R1
      012746 010701                SUB      #-Q4,R1
      012750 162701 001116
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                                50053$:
      012776                ENDIF
      012776
2447

```

```

2448 012776          ENDIF
2449 012776          50052$:
2450 012776 000407    ELSE
2451 013000          BR      50054$
2452 013000          50051$:
2453 013000 010701    ;ADDRESS OF BEGINING OF PROGRAM
2454 013002 162701 001150 LET R1 := PC - #-Q4
2455 013006 020001    MOV     PC,R1
2456 013010 001002    SUB     #-Q4,R1
2457          IF R0 EQ R1 THEN ;POINTER REACHED PROGRAM?
2458          CMP     R0,R1
2459          BNE     50055$
2460          ;BUMP POINTER TO END OF PROGRAM
2461          LET R0 := R0 + #DIF1
2462          ADD     #DIF1,R0
2463          ENDIF
2464          ENDIF
2465          50055$:
2466          50054$:
2467 013016 032700 017777 IF #17777 SETIN R0 THEN ;ARE WE AT 4K BOUNDARY?
2468 013022 001402    BIT     #17777,R0
2469          BEQ     50056$
2470          ;NO
2471          INLINE <JMP STLOOP> ;CONTINUE LOOP & BYPASS PRINT
2472          JMP STLOOP ;ROUTINES
2473          ENDIF
2474          50056$:
2475          ;ENABLE PARITY
2476          LET R3 := #172100 ;FETCH CSR LOCATIONS
2477          MOV     #172100,R3
2478          LET TRCATCH := #240 ;NOP TRAP CATCHER
2479          MOV     #240,TRCATCH
2480          INCR R2 FROM #0 TO #16. BY #1 ;ENABLE PARITY FOR ALL CSRS
2481          CLR     R2
2482          BR      50057$
2483          50060$:
2484          INC     R2
2485          50057$:
2486          CMP     R2,#16.
2487          BGT     50061$
2488          ;
2489          MOV     #1,(R3)+
2490          LET (R3)+ := #1

```



```

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

```

```

2516
2517 013142
      013142 026767 003032 003026
      013150 001066
2518
2519
2520
2521 013152
      013152 052767 100000 003000
2522
2523 013160
      013160 032767 000001 002772
      013166 001404
2524
2525
2526 013170
      013170 013767 172342 002770
2527
2528 013176
      013176 000405
      013200
2529
2530
2531 013200
      013200 010067 002762
      013204 162767 000002 002754
2532
2533 013212
      013212
2534
2535 013212
      013212 004767 001172
2536
2537 013216
      013216 004767 001142
2538
2539 013222
      013222 010146
2540
2541 013224
      013224 016701 002734
2542
2543 013230
      013230 010146
      013232 104405
2544
2545 013234
      013234 004767 001124
2546
2547 013240
      013240 016701 002726
2548
2549 013244
      013244 010146
      013246 104405
2550
2551 013250

```

MOV R1,-(SP)

MOV R1,-(SP)
TYPDS

MOV R1,-(SP)
TYPDS

```

IF L4KST EQ START THEN ;ONLY NEED TO PRINT ONE LINE
      CMP L4KST,START
      BNE 50067$

;TAKE CARE OF ANY PROBLEMS WITH START ADDRESS
;BEING VOLATILE WITHIN NON-VOLATILE MEMORY
LET CONTROL := CONTROL SET.BY #100000
      BIS #100000,CONTROL

IF #1 SETIN CONTROL THEN ;MAPPED
      BIT #1,CONTROL
      BEQ 50070$

;FETCH LAST BANK ADDRESS
LET END := @#KIPAR1
      MOV @#KIPAR1,END

ELSE
      BR 50071$
      50070$:

;FETCH LAST BANK ADDRESS
LET END := R0 - #2
      MOV R0,END
      SUB #2,END

ENDIF
      50071$:

CALL PRINT ;PRINT ADDRESSES
      JSR PC,PRINT

CALL TAB ;PRINT TAB
      JSR PC,TAB

PUSH R1
;;PUSH R1 ON STACK

LET R1 := VOL ;FETCH VOL COUNT
      MOV VOL,R1

TYPDEC R1 ;PRINT ERROR COUNT
;;SAVE R1 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN

CALL TAB ;PRINT TAB
      JSR PC,TAB

LET R1 := PARCNT ;FETCH PARITY COUNT
      MOV PARCNT,R1

TYPDEC R1 ;PRINT PARITY ERROR CNT.
;;SAVE R1 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN

IF PARCNT EQ #10. THEN ;DO WE HAVE 10 OR MORE

```


	013250	026727	002716	000012		CMP	PARCNT,#10.
2552	013256	001002				BNE	50072\$
2553						;PARITY	ERRORS
2554	013260	104401	015100		TYPE ,MORE ;PRINT OR MORE		
2555							
2556	013264				ENDIF		
	013264						50072\$:
2557							
2558	013264				POP R1 ;RESTORE R1		
	013264	012601			::POP STACK INTO R1		
2559							
2560	013266				CALL CRLF2	;PRINT <CR> <LF>	
	013266	004767	001046			JSR PC,CRLF2	
2561							
2562							
2563	013272				;UP DATE START BANK TO NEXT LEVEL TO BE TESTED		
	013272	032767	000001	002660	IF #1 SETIN CONTROL THEN	;MAPPED	
	013300	001407				BIT #1,CONTROL	
2564						BEQ	50073\$
2565	013302				LET START := @#KIPAR1 + #200		
	013302	013767	172342	002666		MOV @#KIPAR1,START	
	013310	062767	000200	002660		ADD #200,START	
2566							
2567	013316				ELSE		
	013316	000402				BR 50073\$:	50074\$
	013320						
2568							
2569	013320				LET START := RO		
	013320	010067	002652			MOV RO,START	
2570							
2571	013324				ENDIF		
	013324						50074\$:
2572							
2573	013324				ELSE		
	013324	000532				BR 50067\$:	50075\$
	013326						
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 50076\$:	50077\$
	013354						
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							

2585	013370				CALL PRINT ;PRINT ADDRESSES
	013370	004767	001014		JSR PC,PRINT
2586					
2587	013374				IF CONTROL LT #0 THEN ;ARE WE CHECKING NON-VOL
	013374	005767	002560		TST
	013400	002012			BGE 50100\$
2588					
2589	013402				CALL TAB ;PRINT TAB
	013402	004767	000756		JSR PC,TAB
2590					
2591					;NO ERRORS IN THIS PART
2592	013406				TYPDEC #0
	013406	012746	000000	MOV #0,-(SP)	::SAVE #0 FOR TYPEOUT
	013412	104405		TYPDS	::GO TYPE--DECIMAL ASCII WITH SIGN
2593					
2594	013414				CALL TAB ;PRINT TAB
	013414	004767	000744		JSR PC,TAB
2595					
2596	013420				TYPDEC #0 ;NO PARITY EITHER
	013420	012746	000000	MOV #0,-(SP)	::SAVE #0 FOR TYPEOUT
	013424	104405		TYPDS	::GO TYPE--DECIMAL ASCII WITH SIGN
2597					ENDIF
2598	013426				50100\$:
	013426				
2599					;INDICATE THAT NOW THIS IS NON-VOLATILE
2600					LET CONTROL := CONTROL SET BY #100000
2601	013426	052767	100000	002524	BIS #100000,CONTROL
2602					
2603	013434				CALL CRLF2 ;PRINT <CR><LF>
	013434	004767	000700		JSR PC,CRLF2
2604					
2605	013440				IF #1 SET IN CONTROL THEN ;MAPPED
	013440	032767	000001	002512	BIT #1,CONTROL
	013446	001404			BEQ 50101\$
2606					
2607					;LAST 4K BANK
2608	013450				LET END := @#KIPAR1
	013450	013767	172342	002510	MOV @#KIPAR1,END
2609					
2610	013456				ELSE
	013456	000405			BR 50102\$
	013460				50101\$:
2611					
2612					;LAST ADDRESS
2613	013460				LET END := R0 - #2
	013460	010067	002502		MOV R0,END
	013464	162767	000002	002474	SUB #2,END
2614					
2615	013472				ENDIF
	013472				50102\$:
2616					
2617	013472				LET START := L4KST
	013472	016767	002502	002476	MOV L4KST,START
2618					
2619	013500				CALL PRINT ;PRINT ADDRESSES
	013500	004767	000704		JSR PC,PRINT

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

```

2656 013612
2657 013612
2658 013612
2659 013612 005704 20$: IF R4 EQ #0 THEN ;HAVE WE PRINTED ANYTHING UP TO NOW
    013612 001132 TST R4
    013614 BNE 50106$
2660
2661 013616 026767 002354 002354 IF START EQ L4KST THEN ;MAKE SURE THAT CONTROL IS
    013616 001013 CMP START,L4KST
    013624 BNE 50107$
    ;SET PROPERLY
2662
2663
2664 013626 IF VOL GE NONVOL THEN
    013626 026767 002332 002326 CMP VOL, NONVOL
    013634 002404 BLT 50110$
2665
2666 013636 LET CONTROL := CONTROL CLR.BY #100000
    013636 042767 100000 002314 BIC #100000, CONTROL
2667
2668 013644 ELSE
    013644 000403 BR 50111$
    013646 50110$:
2669
2670 013646 LET CONTROL := CONTROL SET.BY #100000
    013646 052767 100000 002304 BIS #100000, CONTROL
2671
2672 013654
    013654
2673
2674 013654
    013654
2675
2676 013654 IF VOL GE NONVOL THEN ;NO, ARE WE IN NON VOLATILE
    013654 026767 002304 002300 CMP VOL, NONVOL
    013662 002451 BLT 50112$
    ;MEMORY?
2677
2678
2679
2680 013664 ; WERE WE LOOKING FOR NON-VOLATILE
    013664 005767 002270 IF CONTROL LT #0 THEN
    013670 002045 TST CONTROL
    BGE 50113$
2681
2682
2683 013672 ;ARE WE A MAPPED SYSTEM?
    013672 032767 000001 002260 IF #1 SETIN CONTROL THEN
    013700 001407 BIT #1, CONTROL
    BEQ 50114$
2684
2685
2686 013702 ;FETCH LAST 4K BANK
    013702 016767 002272 002256 LET END := L4KST - #200
    013710 162767 000200 002250 MOV L4KST, END
    SUB #200, END
2687
2688 013716 ELSE
    013716 000406 BR 50115$
    013720 50114$:

```



```

;FETCH LAST ADDRESS OF
;OF NON-VOLATILE MEMORY
LET END := L4KST - #2
MOV L4KST,END
SUB #2,END

ENDIF

50115$:

CALL PRINT JSR ;PRINT ADDRESSES
PC,PRINT

CALL TAB JSR ;PRINT TAB
PC,TAB

TYPDEC #0 ;NO ERRORS
;;SAVE #0 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN

CALL TAB JSR ;PRINT TAB
PC,TAB

TYPDEC #0 ;NO PARITY ERRORS
;;SAVE #0 FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN

CALL CRLF2 JSR ;PRINT <CR><LF>
PC,CRLF2

LET START := L4KST
MOV L4KST,START

;INDICATE NOW LOOKING FOR VOLATILE
LET CONTROL := CONTROL CLR.BY #100000
BIC #100000,CONTROL

ENDIF

50113$:

BR 50116$
50112$:

;YES, WERE WE LOOKING FOR VOLATILE
IF CONTROL GE #0 THEN

TST CONTROL
BLT 50117$

;YES, ARE WE MAPPED
IF #1 SETIN CONTROL THEN

BIT #1,CONTROL
BEQ 50120$

;FETCH LAST 4K BANK BEFORE CHANGE
LET END := L4KST - #200
MOV L4KST,END

```

2725 014032 162767 000200 002126
2726 014040 000406
2727 014042
2728 014042 016767 002132 002116
2729 014050 162767 000002 002110
2730 014056
2731 014056
2732 014056 004767 000326
2733 014056
2734 014062 004767 000252
2735 014062
2736 014062 052767 100000 002064
2737 014066
2738 014066
2739 014074 016767 002100 002074
2740 014074
2741 014102
2742 014102
2743 014102
2744 014102
2745 014102
2746 014102
2747 014102
2748 014102
2749 014102
2750 014102 032767 000040 002050
2751 014102 001410
2752 014112
2753 014112 010701 000775
2754 014114 062701
2755 014120 104401
2756 014122 015111 100\$:
2757 014124
2758 014124 042767 000040 002026
2759 014132
2760 014132
2761 014132
2762

SUB #200,END
ELSE
BR 50121\$
50120\$:
;FETCH LAST ADDRESS BEFORE CHANGE
LET END := L4KST - #2
MOV L4KST,END
SUB #2,END
ENDIF
50121\$:
;PRINT ADDRESSES
CALL PRINT
JSR PC,PRINT
CALL CRLF2 ;PRINT <CR><LF>
JSR PC,CRLF2
;INDICATE NOW LOOKING FOR
;NON-VOLATILE MEMORY
LET CONTROL := CONTROL SET.BY #100000
BIS #100000,CONTROL
;BUMP POINTER
LET START := L4KST
MOV L4KST,START
ENDIF
50117\$:
50116\$:
50106\$:
IF #40 SETIN CONTROL THEN ;DO WE PRINT TRAP PAGE CHECKSUM
BIT #40,CONTROL
BEQ 50122\$
;ERROR. YES
LET R1 := PC + #TRAPER-. ;FETCH MESSAGE
MOV PC,R1
ADD #TRAPER-.,R1
TYPE .WORD TRAPER ;TELL OF ERROR
LET CONTROL := CONTROL CLR.BY #40 ;INDICATE
BIC #40,CONTROL
;MESSAGE ALREADY PRINTED
50122\$:
ENDIF


```

2763 014132 005067 002026      LET VOL := #0          ;CLEAR VOLATILE FLAG
                                CLR      VOL
2764 014132 005067 002026
2765 014136 005067 002020      LET NONVOL := #0        ;CLEAR NON-VOLATILE FLAG
                                CLR      NONVOL
2766 014142 005067 002024      LET PARCNT := #0         ;CLEAR PARITY COUNT
                                CLR      PARCNT
2767 014142 005067 002024
2768 014146 032767 000001 002004  IF #1 SETIN CONTROL THEN ;ARE WE MAPPED?
2769 014146 001411              BIT      #1,CONTROL
                                BEQ      50123$
                                ;YES
2770
2771
2772 014156 062737 000200 172342  LET @#KIPAR1 := @#KIPAR1 + #200 ;GOTO NEXT BANK
                                ADD      #200,@#KIPAR1
2773
2774 014164 012700 020000      LET R0 := #20000        ;RETURN TO ADDRESS 0 PAR1
                                MOV      #20000,R0
2775
2776 014170 013767 172342 002002  LET L4KST := @#KIPAR1 ;FETCH NEXT BANK
                                MOV      @#KIPAR1,L4KST
2777
2778 014176 000402      ELSE
                                BR      50124$
                                50123$:
2779
2780 014200 010067 001774      LET L4KST := R0        ;FETCH NEXT BANK
                                MOV      R0,L4KST
2781
2782 014204      ENDIF
                                50124$:
2783
2784 014204 000167 176376      INLINE <JMP STLOOP> ;CONTINUE LOOP
                                JMP      STLOOP
2785
2786 014210 005704      EDLOOP: IF R4 EQ #0 THEN ;DO WE NEED TO PRINT LAST MEMORY UNDER TEST
                                TST      R4
                                BNE      50125$
                                ;YES
2787
2788
2789 014214 032767 000001 001736  IF #1 SETIN CONTROL THEN ;ARE WE MAPPED?
                                BIT      #1,CONTROL
                                BEQ      50126$
2790
2791 014224 013767 172342 001734  LET END := @#KIPAR1 - #200 ;FETCH LAST 4K BANK
                                MOV      @#KIPAR1,END
                                SUB      #200,END
2792
2793 014240 000405      ELSE
                                BR      50127$
                                50126$:
2794
2795 014242 010067 001720 001712  LET END := R0 - #2        ;FETCH LAST ADDRESS
                                MOV      R0,END
                                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                ;: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                ;: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
2819 014316          ;FETCH LOCATION OF FINISHED MESSAGE
      014316 010701 001037  LET R1 := PC + #TAF-.
      014320 062701
2820
2821          ;LOAD ADDRESS INTO TRAP INSTRUCTION
2822 014324          LET 18$ := R1
      014324 010167 000002
2823
2824          ;TELL OPERATOR WE ARE FINISHED
2825 014330 104401  TYPE
2826 014332 015357  .WORD  TAF
2827
2828 014334          INLINE  <HALT>
      014334 000000
2829
2830          ENDRTN
      014336
      014336
      014336
      014336 000207

```

50127\$:

;PRINT ADDRESSES
JSR PC,PRINT

;WAS LAST CHUNK OF MEMORY
TST CONTROL
BGE 50130\$
;NON-VOL - YES, PRINT NO ERRORS

;PRINT TAB
JSR PC,TAB

;PRINT ANY ERRORS
;:SAVE #0 FOR TYPEOUT
;:GO TYPE--DECIMAL ASCII WITH SIGN

;PRINT TAB
JSR PC,TAB

;PARITY ERRORS
;:SAVE #0 FOR TYPEOUT
;:GO TYPE--DECIMAL ASCII WITH SIGN

50130\$:

;PRINT <CR><LF>
JSR PC,CRLF2

50125\$:

MOV PC,R1
ADD #TAF-.,R1

MOV R1,18\$

HALT

50000\$:
50001\$:
RTS PC

2832	014340			ROUTINE CRLF2		
	014340					CRLF2:
2833						
2834				:SAVE R1		
2835	014340			PUSH R1		
	014340	010146		MOV R1,-(SP)	::PUSH R1 ON STACK	
2836						
2837				:FETCH LOCATION OF <CR><LF> MESSAGE		
2838	014342			LET R1 := PC + #CRLF1-		
	014342	010701				MOV PC,R1
	014344	062701	001611			ADD #CRLF1-.,R1
2839						
2840				:LOAD ADDRESS INTO TRAP INSTRUCTION		
2841	014350			LET 5\$:= R1		
	014350	010167	000002			MOV R1,5\$
2842						
2843				:TELL OPERATOR <CR><LF>		
2844	014354	104401		TYPE		
2845	014356	016155		5\$: .WORD CRLF1		
2846						
2847				:RESTORE R1		
2848	014360			POP R1		
	014360	012601		MOV (SP)+,R1	::POP STACK INTO R1	
2849						
2850	014362			ENDRTN		
	014362					50000\$:
	014362					50001\$:
	014362	000207				RTS PC

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


```

2872          .SBTTL PRINT ADDRESSES ROUTINE
2873
2874          ;THIS ROUTINE PRINTS THE ADDRESSES IN THE MEMORY MAP
2875          ROUTINE PRINT
2876 014410
2877 014410
2878 014410 032767 000001 001542          IF #1 SETIN CONTROL THEN          ;ARE WE A MAPPED SYSTEM
2879 014416 001416          ;YES, FETCH 4K BANK
2880 014420 016702 001552          LET R2 := START          ;YES, FETCH 4K BANK
2881 014424 005001          LET R1 := #0          ;ALWAYS AT 4K BOUNDARY
2882 014426 004767 000366          CALL P22BAD          ;PRINT START ADDRESS
2883 014432 004767 177726          CALL TAB          ;PRINT TAB
2884 014436 016702 001524          LET R2 := END          ;FETCH LAST 4K BANK
2885 014442 012701 017776          LET R1 := #17776          ;LAST ADDRESS IN 4K BANK
2886 014446 004767 000346          CALL P22BAD          ;PRINT TAB
2887 014452 000442          ELSE          ;LAST ADDRESS IN 4K BANK
2888 014454 000442          ;PRINT TAB
2889 014460 005701 002405          LET R1 := START          ;FETCH BEGINNING ADDRESS
2890 014462 000404          IF R1 GE #0 THEN          ;TYPE UPPER BIT AS ZERO
2891 014474 000404          ;TYPE UPPER BIT AS ONE
2892 014476 000404          ;TYPE UPPER BIT AS ONE
2893 014476 012746 000000          MOV #0,-(SP)          ;TYPE UPPER BIT AS ONE
2894 014476 104403 000001          TYPOS #0,4          ;TYPE UPPER BIT AS ONE
2895 014476 004          ;TYPE UPPER BIT AS ONE
2896 014476 000          ;TYPE UPPER BIT AS ONE
2897 014476 000          ;TYPE UPPER BIT AS ONE
2898 014476 000          ;TYPE UPPER BIT AS ONE
2899 014476 000          ;TYPE UPPER BIT AS ONE
2900 014476 000          ;TYPE UPPER BIT AS ONE
2901 014476 000          ;TYPE UPPER BIT AS ONE
2902 014476 000          ;TYPE UPPER BIT AS ONE
2903 014476 000          ;TYPE UPPER BIT AS ONE
2904 014476 000          ;TYPE UPPER BIT AS ONE

```

```

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 005 TYPOS ;;GO TYPE--OCTAL ASCII
014512 001 .BYTE 5 ;;TYPE 5 DIGIT(S)
014513 .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
2934      LET 5$ := R1
2935      MOV      R1,5$
2936
2937      ;TELL OPERATOR NON VOLATILE
2938      TYPE
2939      .WORD    NVMES
2940      IF START EQ #0 THEN      ;ARE WE IN PAGE 0
                                TST    START
                                BNE     50011$
                                ;YES
2941
2942      IF #4000 SETIN CONTROL THEN ;WAS THERE TRAP PAGE
                                BIT     #4000,CONTROL
                                BEQ      50012$
                                ;CHECKSUM ERROR
                                ;YES
2943      014612  032767  004000  001340
2944      014612  001403
2945
2946      ;INDICATE PRINT ERROR MESSAGE
2947      LET CONTROL := CONTROL SET.BY #40 ;
2948      BIS      #40,CONTROL
2949
2950      ENDIF
2951
2952      ENDIF
2953
2954      ELSE
2955      014630  000407
2956      014630
2957      014632  010701  000647
2958      014632  062701
2959
2960      ;LOAD ADDRESS INTO TRAP INSTRUCTION
2961      LET 6$ := R1
2962      MOV      R1,6$
2963
2964      ;TELL OPERATOR VOLATILE
2965      TYPE
2966      .WORD    VMES
2967      014650
2968      014650
2969      014650
2970      014650  000207
2971
2972      ENDRTN
2973
2974      50000$:
2975      50001$:
2976      RTS      PC
2977
2978      50012$:
2979
2980      50011$:
2981
2982      BR      50013$
2983      50010$:
2984
2985      50013$:
2986
2987      50000$:
2988      50001$:
2989      RTS      PC
2990
2991      50013$:
2992
2993      50000$:
2994      50001$:
2995      RTS      PC
2996
2997      50013$:
2998
2999      50000$:
3000      50001$:
3001      RTS      PC
3002
3003      50013$:
3004
3005      50000$:
3006      50001$:
3007      RTS      PC
3008
3009      50013$:
3010
3011      50000$:
3012      50001$:
3013      RTS      PC
3014
3015      50013$:
3016
3017      50000$:
3018      50001$:
3019      RTS      PC
3020
3021      50013$:
3022
3023      50000$:
3024      50001$:
3025      RTS      PC
3026
3027      50013$:
3028
3029      50000$:
3030      50001$:
3031      RTS      PC
3032
3033      50013$:
3034
3035      50000$:
3036      50001$:
3037      RTS      PC
3038
3039      50013$:
3040
3041      50000$:
3042      50001$:
3043      RTS      PC
3044
3045      50013$:
3046
3047      50000$:
3048      50001$:
3049      RTS      PC
3050
3051      50013$:
3052
3053      50000$:
3054      50001$:
3055      RTS      PC
3056
3057      50013$:
3058
3059      50000$:
3060      50001$:
3061      RTS      PC
3062
3063      50013$:
3064
3065      50000$:
3066      50001$:
3067      RTS      PC
3068
3069      50013$:
3070
3071      50000$:
3072      50001$:
3073      RTS      PC
3074
3075      50013$:
3076
3077      50000$:
3078      50001$:
3079      RTS      PC
3080
3081      50013$:
3082
3083      50000$:
3084      50001$:
3085      RTS      PC
3086
3087      50013$:
3088
3089      50000$:
3090      50001$:
3091      RTS      PC
3092
3093      50013$:
3094
3095      50000$:
3096      50001$:
3097      RTS      PC
3098
3099      50013$:
3100
3101      50000$:
3102      50001$:
3103      RTS      PC
3104
3105      50013$:
3106
3107      50000$:
3108      50001$:
3109      RTS      PC
3110
3111      50013$:
3112
3113      50000$:
3114      50001$:
3115      RTS      PC
3116
3117      50013$:
3118
3119      50000$:
3120      50001$:
3121      RTS      PC
3122
3123      50013$:
3124
3125      50000$:
3126      50001$:
3127      RTS      PC
3128
3129      50013$:
3130
3131      50000$:
3132      50001$:
3133      RTS      PC
3134
3135      50013$:
3136
3137      50000$:
3138      50001$:
3139      RTS      PC
3140
3141      50013$:
3142
3143      50000$:
3144      50001$:
3145      RTS      PC
3146
3147      50013$:
3148
3149      50000$:
3150      50001$:
3151      RTS      PC
3152
3153      50013$:
3154
3155      50000$:
3156      50001$:
3157      RTS      PC
3158
3159      50013$:
3160
3161      50000$:
3162      50001$:
3163      RTS      PC
3164
3165      50013$:
3166
3167      50000$:
3168      50001$:
3169      RTS      PC
3170
3171      50013$:
3172
3173      50000$:
3174      50001$:
3175      RTS      PC
3176
3177      50013$:
3178
3179      50000$:
3180      50001$:
3181      RTS      PC
3182
3183      50013$:
3184
3185      50000$:
3186      50001$:
3187      RTS      PC
3188
3189      50013$:
3190
3191      50000$:
3192      50001$:
3193      RTS      PC
3194
3195      50013$:
3196
3197      50000$:
3198      50001$:
3199      RTS      PC
3200
3201      50013$:
3202
3203      50000$:
3204      50001$:
3205      RTS      PC
3206
3207      50013$:
3208
3209      50000$:
3210      50001$:
3211      RTS      PC
3212
3213      50013$:
3214
3215      50000$:
3216      50001$:
3217      RTS      PC
3218
3219      50013$:
3220
3221      50000$:
3222      50001$:
3223      RTS      PC
3224
3225      50013$:
3226
3227      50000$:
3228      50001$:
3229      RTS      PC
3230
3231      50013$:
3232
3233      50000$:
3234      50001$:
3235      RTS      PC
3236
3237      50013$:
3238
3239      50000$:
3240      50001$:
3241      RTS      PC
3242
3243      50013$:
3244
3245      50000$:
3246      50001$:
3247      RTS      PC
3248
3249      50013$:
3250
3251      50000$:
3252      50001$:
3253      RTS      PC
3254
3255      50013$:
3256
3257      50000$:
3258      50001$:
3259      RTS      PC
3260
3261      50013$:
3262
3263      50000$:
3264      50001$:
3265      RTS      PC
3266
3267      50013$:
3268
3269      50000$:
3270      50001$:
3271      RTS      PC
3272
3273      50013$:
3274
3275      50000$:
3276      50001$:
3277      RTS      PC
3278
3279      50013$:
3280
3281      50000$:
3282      50001$:
3283      RTS      PC
3284
3285      50013$:
3286
3287      50000$:
3288      50001$:
3289      RTS      PC
3290
3291      50013$:
3292
3293      50000$:
3294      50001$:
3295      RTS      PC
3296
3297      50013$:
3298
3299      50000$:
3300      50001$:
3301      RTS      PC
3302
3303      50013$:
3304
3305      50000$:
3306      50001$:
3307      RTS      PC
3308
3309      50013$:
3310
3311      50000$:
3312      50001$:
3313      RTS      PC
3314
3315      50013$:
3316
3317      50000$:
3318      50001$:
3319      RTS      PC
3320
3321      50013$:
3322
3323      50000$:
3324      50001$:
3325      RTS      PC
3326
3327      50013$:
3328
3329      50000$:
3330      50001$:
3331      RTS      PC
3332
3333      50013$:
3334
3335      50000$:
3336      50001$:
3337      RTS      PC
3338
3339      50013$:
3340
3341      50000$:
3342      50001$:
3343      RTS      PC
3344
3345      50013$:
3346
3347      50000$:
3348      50001$:
3349      RTS      PC
3350
3351      50013$:
3352
3353      50000$:
3354      50001$:
3355      RTS      PC
3356
3357      50013$:
3358
3359      50000$:
3360      50001$:
3361      RTS      PC
3362
3363      50013$:
3364
3365      50000$:
3366      50001$:
3367      RTS      PC
3368
3369      50013$:
3370
3371      50000$:
3372      50001$:
3373      RTS      PC
3374
3375      50013$:
3376
3377      50000$:
3378      50001$:
3379      RTS      PC
3380
3381      50013$:
3382
3383      50000$:
3384      50001$:
3385      RTS      PC
3386
3387      50013$:
3388
3389      50000$:
3390      50001$:
3391      RTS      PC
3392
3393      50013$:
3394
3395      50000$:
3396      50001$:
3397      RTS      PC
3398
3399      50013$:
3400
3401      50000$:
3402      50001$:
3403      RTS      PC
3404
3405      50013$:
3406
3407      50000$:
3408      50001$:
3409      RTS      PC
3410
3411      50013$:
3412
3413      50000$:
3414      50001$:
3415      RTS      PC
3416
3417      50013$:
3418
3419      50000$:
3420      50001$:
3421      RTS      PC
3422
3423      50013$:
3424
3425      50000$:
3426      50001$:
3427      RTS      PC
3428
3429      50013$:
3430
3431      50000$:
3432      50001$:
3433      RTS      PC
3434
3435      50013$:
3436
3437      50000$:
3438      50001$:
3439      RTS      PC
3440
3441      50013$:
3442
3443      50000$:
3444      50001$:
3445      RTS      PC
3446
3447      50013$:
3448
3449      50000$:
3450      50001$:
3451      RTS      PC
3452
3453      50013$:
3454
3455      50000$:
3456      50001$:
3457      RTS      PC
3458
3459      50013$:
3460
3461      50000$:
3462      50001$:
3463      RTS      PC
3464
3465      50013$:
3466
3467      50000$:
3468      50001$:
3469      RTS      PC
3470
3471      50013$:
3472
3473      50000$:
3474      50001$:
3475      RTS      PC
3476
3477      50013$:
3478
3479      50000$:
3480      50001$:
3481      RTS      PC
3482
3483      50013$:
3484
3485      50000$:
3486      50001$:
3487      RTS      PC
3488
3489      50013$:
3490
3491      50000$:
3492      50001$:
3493      RTS      PC
3494
3495      50013$:
3496
3497      50000$:
3498      50001$:
3499      RTS      PC
3500
3501      50013$:
3502
3503      50000$:
3504      50001$:
3505      RTS      PC
3506
3507      50013$:
3508
3509      50000$:
3510      50001$:
3511      RTS      PC
3512
3513      50013$:
3514
3515      50000$:
3516      50001$:
3517      RTS      PC
3518
3519      50013$:
3520
3521      50000$:
3522      50001$:
3523      RTS      PC
3524
3525      50013$:
3526
3527      50000$:
3528      50001$:
3529      RTS      PC
3530
3531      50013$:
3532
3533      50000$:
3534      50001$:
3535      RTS      PC
3536
3537      50013$:
3538
3539      50000$:
3540      50001$:
3541      RTS      PC
3542
3543      50013$:
3544
3545      50000$:
3546      50001$:
3547      RTS      PC
3548
3549      50013$:
3550
3551      50000$:
3552      50001$:
3553      RTS      PC
3554
3555      50013$:
3556
3557      50000$:
3558      50001$:
3559      RTS      PC
3560
3561      50013$:
3562
3563      50000$:
3564      50001$:
3565      RTS      PC
3566
3567      50013$:
3568
3569      50000$:
3570      50001$:
3571      RTS      PC
3572
3573      50013$:
3574
3575      50000$:
3576      50001$:
3577      RTS      PC
3578
3579      50013$:
3580
3581      50000$:
3582      50001$:
3583      RTS      PC
3584
3585      50013$:
3586
3587      50000$:
3588      50001$:
3589      RTS      PC
3590
3591      50013$:
3592
3593      50000$:
3594      50001$:
3595      RTS      PC
3596
3597      50013$:
3598
3599      50000$:
3600      50001$:
3601      RTS      PC
3602
3603      50013$:
3604
3605      50000$:
3606      50001$:
3607      RTS      PC
3608
3609      50013$:
3610
3611      50000$:
3612      50001$:
3613      RTS      PC
3614
3615      50013$:
3616
3617      50000$:
3618      50001$:
3619      RTS      PC
3620
3621      50013$:
3622
3623      50000$:
3624      50001$:
3625      RTS      PC
3626
3627      50013$:
3628
3629      50000$:
3630      50001$:
3631      RTS      PC
3632
3633      50013$:
3634
3635      50000$:
3636      50001$:
3637      RTS      PC
3638
3639      50013$:
3640
3641      50000$:
3642      50001$:
3643      RTS      PC
3644
3645      50013$:
3646
3647      50000$:
3648      50001$:
3649      RTS      PC
3650
3651      50013$:
3652
3653      50000$:
3654      50001$:
3655      RTS      PC
3656
3657      50013$:
3658
3659      50000$:
3660      50001$:
3661      RTS      PC
3662
3663      50013$:
3664
3665      50000$:
3666      50001$:
3667      RTS      PC
3668
3669      50013$:
3670
3671      50000$:
3672      50001$:
3673      RTS      PC
3674
3675      50013$:
3676
3677      50000$:
3678      50001$:
3679      RTS      PC
3680
3681      50013$:
3682
3683      50000$:
3684      50001$:
3685      RTS      PC
3686
3687      50013$:
3688
3689      50000$:
3690      50001$:
3691      RTS      PC
3692
3693      50013$:
3694
3695      50000$:
3696      50001$:
3697      RTS      PC
3698
3699      50013$:
3700
3701      50000$:
3702      50001$:
3703      RTS      PC
3704
3705      50013$:
3706
3707      50000$:
3708      50001$:
3709      RTS      PC
3710
3711      50013$:
3712
3713      50000$:
3714      50001$:
3715      RTS      PC
3716
3717      50013$:
3718
3719      50000$:
3720      50001$:
3721      RTS      PC
3722
3723      50013$:
3724
3725      50000$:
3726      50001$:
3727      RTS      PC
3728
3729      50013$:
3730
3731      50000$:
3732      50001$:
3733      RTS      PC
3734
3735      50013$:
3736
3737      50000$:
3738      50001$:
3739      RTS      PC
3740
3741      50013$:
3742
3743      50000$:
3744      50001$:
3745      RTS      PC
3746
3747      50013$:
3748
3749      50000$:
3750      50001$:
3751      RTS      PC
3752
3753      50013$:
3754
3755      50000$:
3756      50001$:
3757      RTS      PC
3758
3759      50013$:
3760
3761      50000$:
3762      50001$:
3763      RTS      PC
3764
3765      50013$:
3766
3767      50000$:
3768      50001$:
3769      RTS      PC
3770
3771      50013$:
3772
3773      50000$:
3774      50001$:
3775      RTS      PC
3776
3777      50013$:
3778
3779      50000$:
3780      50001$:
3781      RTS      PC
3782
3783      50013$:
3784
3785      50000$:
3786      50001$:
3787      RTS      PC
3788
3789      50013$:
3790
3791      50000$:
3792      50001$:
3793      RTS      PC
3794
3795      50013$:
3796
3797      50000$:
3798      50001$:
3799      RTS      PC
3800
3801      50013$:
3802
3803      50000$:
3804      50001$:
3805      RTS      PC
3806
3807      50013$:
3808
3809      50000$:
3810      50001$:
3811      RTS      PC
3812
3813      50013$:
3814
3815      50000$:
3816      50001$:
3817      RTS      PC
3818
3819      50013$:
3820
3821      50000$:
3822      50001$:
3823      RTS      PC
3824
3825      50013$:
3826
3827      50000$:
3828      50001$:
3829      RTS      PC
3830
3831      50013$:
3832
3833      50000$:
3834      50001$:
3835      RTS      PC
3836
3837      50013$:
3838
3839      50000$:
3840      50001$:
3841      RTS      PC
3842
3843      50013$:
3844
3845      50000$:
3846      50001$:
3847      RTS      PC
3848
3849      50013$:
3850
3851      50000$:
3852      50001$:
3853      RTS      PC
3854
3855      50013$:
3856
3857      50000$:
3858      50001$:
3859      RTS      PC
3860
3861      50013$:
3862
3863      50000$:
3864      50001$:
3865      RTS      PC
3866
3867      50013$:
3868
3869      50000$:
3870      50001$:
3871      RTS      PC
3872
3873      50013$:
3874
3875      50000$:
3876      50001$:
3877      RTS      PC
3878
3879      50013$:
3880
3881      50000$:
3882      50001$:
3883      RTS      PC
3884
3885      50013$:
3886
3887      50000$:
3888      50001$:
3889      RTS      PC
3890
3891      50013$:
3892
3893      50000$:
3894      50001$:
3895      RTS      PC
3896
3897      50013$:
3898
3899      50000$:
3900      50001$:
3901      RTS      PC
3902
3903      50013$:
3904
3905      50000$:
3906      50001$:
3907      RTS      PC
3908
3909      50013$:
3910
3911      50000$:
3912      50001$:
3913      RTS      PC
3914
3915      50013$:
3916
3917      50000$:
3918      50001$:
3919      RTS      PC
3920
3921      50013$:
3922
3923      50000$:
3924      50001$:
3925      RTS      PC
3926
3927      50013$:
3928
3929      50000$:
3930      50001$:
3931      RTS      PC
3932
3933      50013$:
3934
3935      50000$:
3936      50001$:
3937      RTS      PC
3938
3939      50013$:
3940
3941      50000$:
3942      50001$:
3943      RTS      PC
3944
3945      50013$:
3946
3947      50000$:
3948      50001$:
3949      RTS      PC
3950
3951      50013$:
3952
3953      50000$:
3954      50001$:
3955      RTS      PC
3956
3957      50013$:
3958
3959      50000$:
3960      50001$:
3961      RTS      PC
3962
3963      50013$:
3964
3965      50000$:
3966      50001$:
3967      RTS      PC
3968
3969      50013$:
3970
3971      50000$:
3972      50001$:
3973      RTS      PC
3974
3975      50013$:
3976
3977      50000$:
3978      50001$:
3979      RTS      PC
3980
3981      50013$:
3982
3983      50000$:
3984      50001$:
3985      RTS      PC
3986
3987      50013$:
3988
3989      50000$:
3990      50001$:
3991      RTS      PC
3992
3993      50013$:
3994
3995      50000$:
3996      50001$:
3997      RTS      PC
3998
3999      50013$:
4000
3999      50000$:
4000      50001$:
4001      RTS      PC
4002
4003      50013$:
4004
4003      50000$:
4004      50001$:
4005      RTS      PC
4006
4007      50013$:
4008
4007      50000$:
4008      50001$:
4009      RTS      PC
4009
4009      50013$:
4010
4009      50000$:
4010      50001$:
4011      RTS      PC
4010
4010      50013$:
4011
4010      50000$:
4011      50001$:
4012      RTS      PC
4011
4011      50013$:
4012
4011      50000$:
4012      50001$:
4013      RTS      PC
4012
4012      50013$:
4013
4012      50000$:
4013      50001$:
4014      RTS      PC
4013
4013      50013$:
4014
4013      50000$:
4014      50001$:
4015      RTS      PC
4014
4014      50013$:
4015
4014      50000$:
4015      50001$:
4016      RTS      PC
4015
4015      50013$:
4016
4015      50000$:
4016      50001$:
4017      RTS      PC
4016
4016      50013$:
4017
4016      50000$:
4017      50001$:
4018      RTS      PC
4017
4017      50013$:
4018
4017      50000$:
4018      50001$:
4019      RTS      PC
4018
4018      50013$:
4019
4018      50000$:
4019      50001$:
4020      RTS      PC
4019
4019      50013$:
4020
4019      50000$:
4020      50001$:
4021      RTS      PC
4020
4020      50013$:
4021
4020      50000$:
4021      50001$:
4022      RTS      PC
4021
4021      50013$:
4022
4021      50000$:
4022      50001$:
4023      RTS      PC
4022
4022      50013$:
4023
4022      50000$:
4023      50001$:
4024      RTS      PC
4023
4023      50013$:
4024
4023      50000$:
4024      50001$:
4025      RTS      PC
4024
4024      50013$:
4025
4024      50000$:
4025      50001$:
4026      RTS      PC
4025
4025      50013$:
4026
4025      50000$:
4026      50001$:
4027      RTS      PC
4026
4026      50013$:
4027
4026      50000$:
4027      50001$:
4028      RTS      PC
4027
4027      50013$:
4028
4027      50000$:
4028      50001$:
4029      RTS      PC
4028
4028      50013$:
4029
4028      50000$:
4029      50001$:
4030      RTS      PC
4029
4029      50013$:
4030
4029      50000$:
4030      50001$:
4031      RTS      PC
4030
4030      50013$:
4031
4030      50000$:
4031      50001$:
4032      RTS      PC
4031
4031      50013$:
4032
4031      50000$:
4032      50001$:
4033      RTS      PC
4032
4032      50013$:
4033
4032      50000$:
4033      50001$:
4034      RTS      PC
4033
4033      50013$:
4034
4033      50000$:
4034      50001$:
4035      RTS      PC
4034
4034      50013$:
4035
4034      50000$:
4035      50001$:
4036      RTS      PC
4035
4035      50013$:
4036
4035      50000$:
4036      50001$:
4037      RTS      PC
4036
```

```

2970          .SBTTL  PARITY CATCHER
2971
2972          ;THIS ROUTINE CATCHES PARITY ERRORS AND STORES THEN IN PARCNT
2973
2974          IROUTINE PARTRP
2975
2976          014652          LET PARCNT := PARCNT + #1          ;INDICATE PARITY ERROR
2977          014652          005267  001314          INC          PARCNT
2978
2979          014656          ;DISABLE PARITY
2980          014656          LET TRCATCH := #240          ;IF CSR DOES NOT EXIST THEN JUST RTI
2981          014664          012767  000240  001500          MOV          #240,TRCATCH
2982
2983          014664          LET R2 := #172100          ;FIRST CSR ADDRESS
2984          014664          012702  172100          MOV          #172100,R2
2985
2986          014670          INCR R1 FROM #0 TO #16. BY #1          ;CLEAR ALL CSRS
2987          014670          005001          CLR          R1
2988          014672          000401          BR          50002$
2989          014674          005201          50003$:          INC          R1
2990          014676          020127  000020          50002$:          CMP          R1,#16.
2991          014702          003002          BGT          50004$
2992
2993          014704          LET (R2)+ := #0          CLR          (R2)+
2994          014704          005022
2995
2996          014706          ENDINC          BR          50003$
2997          014706          000772          50004$:
2998          014710          LET TRCATCH := #0          ;RESUME HALT IF TRAP
2999          014710          005067  001450          CLR          TRCATCH
3000
3001          014714          IF #125252 EQ -(R0) THEN          ;RECHECK ADDRESS WITH PARITY OFF
3002          014714          022740  125252          CMP          #125252,-(R0)
3003          014720          001003          BNE          50005$
3004
3005          014722          LET NONVOL := NONVOL + #1          ;WE ARE NON-VOLATILE
3006          014722          005267  001234          INC          NONVOL
3007
3008          014726          ELSE          BR          50006$
3009          014726          000402          50005$:
3010          014730          LET VOL := VOL + #1          ;WE ARE VOLATILE
3011          014730          005267  001230          INC          VOL
3012
3013          014734          ENDIF          50006$:
3014          014734
3015          052767  000002  001216          LET CONTROL := CONTROL SET.BY #2          ;INDICATE PARITY ERROR WAS
3016
3017          ;SERVICED
  
```



```

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
3009                                     ;ENABLE PARITY ERRORS THROUGH THE CSRS
3010 014756      LET TRCATCH := #240      ;IF CSR DOES NOT EXIST THEN JUST RTI
      014756 012767 000240 001400      MOV      #240,TRCATCH
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 003Q03      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
3055                                     ;14 & 13 IN R1
3056
3057     LET R2 := R2 SHIFT -1 ;FETCH ADDRESS BITS & PUT BIT 8 INTO
3058                                     ASR R2
3059                                     ;LOWER BYTE
3060
3061     LET R2 := SWAP R2 ;NOW ADDRESS BITS 21-15 ARE IN LSB
3062                                     SWAB R2
3063
3064     LET R2 := R2 CLR.BY #177600 ;CLEAR OUT ALL OTHER BITS
3065                                     BIC #177600,R2
3066
3067     TYPOCS R2,,4 ;PRINT ADDRESS BITS 21 - 15
3068     MOV R2,-(SP) ;:SAVE R2 FOR TYPEOUT
3069     TYPOS ;:GO TYPE--OCTAL ASCII
3070     .BYTE 4 ;:TYPE 4 DIGIT(S)
3071     .BYTE 0 ;:SUPPRESS LEADING ZEROS
3072
3073     TYPOCS R1,,5,X ;PRINT ADDRESS BITS 14 - 0
3074     MOV R1,-(SP) ;:SAVE R1 FOR TYPEOUT
3075     TYPOS ;:GO TYPE--OCTAL ASCII
3076     .BYTE 5 ;:TYPE 5 DIGIT(S)
3077     .BYTE 1 ;:TYPE LEADING ZEROS
3078
3079     POP <R2,R1>
3080     MOV (SP)+,R2 ;:POP STACK INTO R2
3081     MOV (SP)+,R1 ;:POP STACK INTO R1
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
53
```


3060 015076
015076
015076
015076 000207
ENDRTN

50000\$:
50001\$:
RTS PC

```

3062      .SBTTL  DEFINITIONS
3063
3064 015100      040      117      122  MORE:  .ASCIIZ  ' OR MORE '
3065 015111      124      122      101  TRAPER: .ASCII  'TRAP LOCATIONS (ADDRESS 0 - 376) HAVE CHECK SUM ERROR'
3066 015176      040      111      116      .ASCIIZ  ' 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      .ASCIIZ  ' RESIDENT'<12><15>'IN HAS VOLATILE LOCATIONS'<12><15>
3069 015357      012      012      015  TAF:    .ASCIIZ  <12><12><15>'THIS CONCLUDES THE NON-VOLATILE DATA RETENTION TEST'<12><15>
3070 015450      040      040      040  TAB1:   .ASCIIZ  '
3071 015460      040      040      040  NVMES:  .ASCIIZ  '      NON VOLATILE'
3072 015503      040      040      040  VMES:   .ASCIIZ  '      VOLATILE'
3073 015526      012      012      015  HEAD:   .ASCII  <12><12><15>
3074 015531      040      040      040      .ASCII  '
3075 015604      040      040      040      .ASCII  '
3076 015660      123      124      101      .ASCII  'START ADR.      END ADR.      MEMORY MAP'<12><15>
3077 015756      120      101      122      .ASCIIZ  'PARITY ERROR'<12><12><15>      -----'<15><12><12>
3078 015776      120      114      105  PWRDWN: .ASCII  'PLEASE POWER DOWN THIS SYSTEM. AFTER 2 MINUTES BUT'
3079 016060      040      116      117      .ASCII  ' NO LONGER'<12><15>'THAN 100 HOURS,'
3080 016113      040      105      130      .ASCIIZ  ' EXECUTE THE RESTART HELP FILE.'<12><15>
3081 016155      012      015      000  CRLF1:  .ASCIIZ  <12><15>
3082
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      IROUTINE ROMMD
      016202
3097
3098 016202      IF #1 SETIN CONTROL THEN
      016202 032767 000001 177750
      016210 001403
3099
3100
3101 016212      LET      R0 := #40000
      016212 012700 040000
3102
3103
3104
3105 016216      ELSE
      016216 000403
      016220
3106
3107 016220      INLINE <TST -(R0)>
      016220 005740
3108
3109
3110 016222      LET      R0 := R0 + #40000
      016222 062700 040000
3111
3112 016226      ENDIF
      016226
3113
3114 016226      ENDRTI
      016226
      016226 000002
  
```

ROMMD:

```

;ARE WE IN A MAPPED SYSTEM
BIT      #1,CONTROL
BEQ      50002$
;YES
;BUMP POINTER TO INDICATE
MOV      #40000,R0
;WE HAVE REACHED THE END
;OF THIS 4K CHUNK OF MEMORY

BR      50003$
50002$:
;BUMP POINTER BACK TO WHAT IT
TST      -(R0)
;WAS BEFORE TRAP
;BUMP TO NEXT 4K BOUNDARY
ADD      #40000,R0

50003$:

50000$:
50001$:
RTI
  
```

```

3116                                     ;THIS ROUTINE SETS UP TRAP CATCHERS IN ALL TRAP LOCATIONS
3117                                     ;FROM 0 TO 400
3118
3119 016230 ROUTINE TRAPCT
3120 016230
3121 016230 010046
3122                                     TRAPCT:
3123 016232                                     ;ARE WE IN MEMORY MANAGEMENT
016232 032767 000001 177720                                     BIT #1,CONTROL
016240 001407                                     BEQ 50002$
                                     ;SYSTEM
3124
3125
3126 016242 PUSH @#KIPAR1
016242 013746 172342 MOV @#KIPAR1,-(SP) ;;PUSH @#KIPAR1 ON STACK
3127
3128 016246 LET @#KIPAR1 := #0 ;YES, PAR1 WILL POINT TO LOWEST
016246 005037 172342 CLR @#KIPAR1
                                     ;4K
3129
3130 LET R0 := #20000 ;R0 POINTS TO PAR1 AND
016252 012700 020000 MOV #20000,R0
016252                                     ;ADDRESS 0
3132
3133 ELSE
3134 016256 BR 50003$
016256 000401 50002$:
016260
3135
3136 016260 LET R0 := #0 ;START WITH ADDRESS 0
016260 005000 CLR R0
3137
3138 016262 ENDIF
016262
3139 50003$:
3140 016262 LET R1 := PC + #TRCATCH-. ;FIND TRAP CATCHER ROUTINE
016262 010701 MOV PC,R1
016264 062701 000100 ADD #TRCATCH-.,R1
3141
3142 016270 LOOP
016270
3143 50004$:
3144 016270 EXIF R0 EQ #400 OR R0 EQ #20400
016270 020027 000400 CMP R0,#400
016274 001423 BEQ 50005$
016276 020027 020400 CMP R0,#20400
016302 001420 BEQ 50005$
3145
3146 016304 IF R0 EQ #34 OR R0 EQ #20034 THEN ;LOAD TRAP ROUTINE
016304 020027 000034 CMP R0,#34
016310 001403 BEQ 50006$
016312 020027 020034 CMP R0,#20034
016316 001006 BNE 50007$
016320 50006$:
3147
3148 016320 LET (R0) := PC ;LOAD TRAP INSTRUCTION LOCATION
016320 010710 MOV PC,(R0)
  
```



```

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

[illegible]

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: .ASCIIZ <12>	::LINEFEED
3185						
3186	016714				.\$TYPDEC	
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	#\$DBLK-.,R3	::SETUP THE OUTPUT POINTER - R3 POINTS TO \$DBLK
016762	112723	000040		MOVB	#' ,(R3)+	::SET THE FIRST CHARACTER TO A BLANK
016766	005002			2\$: CLR	R2	::CLEAR THE BCD NUMBER
016770	010704			MOV	PC,R4	::USE PC FOR LOCATING TABLE
016772	062704	000154		ADD	#\$DTBL-.,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(R3)	::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	MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
017100	105013		9\$:	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	#\$DBLK-,(SP)	::STACK NOW CONTAINS ADDRESS OF NUMBER TO BE
017124	012667	000002		MOV	(SP)+,10\$	::SET UP ADDRESS FOR TYPE COMMAND
017130	104401			TYPE		::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		::RETURN TO USER
017146	023420			\$DTBL:	10000.	
017150	001750				1000.	
017152	000144				100.	
017154	000012				10.	
3187						
3188	017166			.STYPOCT		
017166	017646	000000		\$TYPOS:	MOV @ (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		BIS	#',R3	::MAKE ASCII IF NOT ALREADY
017342	110367	000052		MOVB	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\$:	DECB	\$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			.\$TRAP		
017426	010046		\$TRAP:	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			MOVB	(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	#\$TRAPD-.,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		\$TRAP2:	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	\$TRAP2	
017476	016370			\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
017500	017212			\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
017502	017166			\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
017504	017226			\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
017506	016714			\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
017510	000404			\$RDCHR	::CALL=RDCHR	TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
017512	000534			\$RDLIN	::CALL=RDLIN	TRAP+7(104407) TTY TYPEIN STRING ROUTINE
017514	000730			\$RDOCT	::CALL=RDOCT	TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
017516	001070			\$RDDEC	::CALL=RDDEC	TRAP+11(104411) READ A DECIMAL NUMBER FROM TTY

3192 017520 000000
3193 017522 000000
3194 017524
3195 017644 000000
3196
3197 017646 000000
3198
3199 000001

CSTRPG: .WORD 0
CHKSUM: .WORD 0
 .BLKW 50
SPINIT: .WORD 0

ENDPR: .WORD 0

 .END

ADDCHK 002074	HELLO 006302	Q4 011632	SW14 = 040000	\$F\$INC= 000210
ASSEMB= 000010	HELP 006174	RDCHR = 104406	SW15 = 100000	\$F\$LOO= 000200
BANK0 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	\$CORE 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	PR0 = 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

\$SSK0 = 050005	\$TPB1 016702	\$TSK5 = 050037	\$SARGC= 000000	\$SRETU= 000000
\$STOP 001426	\$TPFLG 016707	\$TTYIN 000642	\$SBYTE= 000403	\$SRTN1= 050000
\$STUP = 177777	\$TPS1 016700	\$TYPDS 016714	\$SCASE= 000000	\$SRTN2= 050001
\$SWR = 160000	\$TRAP 017426	\$TYPE 016370	\$SDST = 000000	\$SSRC = 000000
\$TAGLE= 177777	\$TRAP2 017462	\$TYPEC 016552	\$SELOC= 000402	\$STGSV= 000000
\$TAGNU= 050002	\$TRP = 000012	\$TYPEX 016672	\$SERFL= 000000	\$STGS1= 000000
\$TEMP = 000350	\$TRPAD 017474	\$TYPOC 017212	\$SFLAG= 000001	\$STGS2= 000000
\$TKB 000402	\$TSK0 = 050011	\$TYPON 017226	\$SFROM= 000000	\$STO = 000000
\$TKB1 016676	\$TSK1 = 050005	\$TYPOS 017166	\$SINH = 000402	\$STOTL= 000000
\$TKS 000400	\$TSK2 = 050010	\$U = 000403	\$SLOC = 016352	\$STAG= 050000
\$TKS1 016674	\$TSK3 = 050121	\$XOFF = 000023	\$SLOCN= 000000	\$OFILL 017423
\$TN = 000001	\$TSK4 = 050061	\$XON = 000021	\$SREG = 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