

.REM 6

IDENTIFICATION

PRODUCT CODE: AC-T227B MC
PRODUCT NAME: CZUACBO NI EXERCISER DIAGNOSTIC
PRODUCT DATE: 22-MAR-84
MAINTAINER: MERRIMACK DIAGNOSTIC ENGINEERING
AUTHOR: GARY MCCOY

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) 1983,1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

TABLE OF CONTENTS

1.0	GENERAL INFORMATION
1.1	PROGRAM ABSTRACT
1.2	SYSTEM REQUIREMENTS
1.3	RELATED DOCUMENTS AND STANDARDS
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES
1.5	ASSUMPTIONS
1.6	HISTORY
2.0	OPERATING INSTRUCTIONS
2.1	COMMANDS
2.2	SWITCHES
2.3	FLAGS
2.4	HARDWARE QUESTIONS
2.5	SOFTWARE QUESTIONS
2.6	QUICK STARTUP PROCEDURE
3.0	ERROR INFORMATION
4.0	PERFORMANCE AND PROGRESS REPORTS
5.0	DEVICE INFORMATION TABLES
6.0	TEST SUMMARIES
6.1	DIRECT
6.2	LOOPPAIR
6.3	PATTERN
6.4	ALL

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

THE NETWORK INTERCONNECT EXERCISER (NIE) PROGRAM IS MEANT TO PROVIDE FIELD SERVICE WITH A TOOL FOR DETERMINING THE CONNECTIVITY OF NODES ON THE NETWORK INTERCONNECT (NI).

THE NIE PROGRAM WILL DETERMINE THE ABILITY OF NODES ON THE NI TO COMMUNICATE WITH EACH OTHER AND PROVIDE NODE INSTALLATION VERIFICATION AND PROBLEM ISOLATION. THE NIE USES THE LOW LEVEL MAINTENANCE FEATURES OF THE DEUNA TO PROVIDE TESTING WITHOUT INTERRUPTING NORMAL OPERATION OF THE NI. THE VAX VERSION OF THE NIE CAN ALSO BE RUN CONCURRENTLY ON ANOTHER NODE, WITH EACH VERSION RUNNING INDEPENDENTLY OF EACH OTHER.

THIS DIAGNOSTIC HAS BEEN WRITTEN FOR USE WITH THE DIAGNOSTIC RUNTIME SERVICES SOFTWARE (SUPERVISOR). THESE SERVICES PROVIDE THE INTERFACE TO THE OPERATOR AND TO THE SOFTWARE ENVIRONMENT. THIS PROGRAM CAN BE USED WITH XXDP+, ACT, APT, SLIDE AND PAPER TAPE. FOR A COMPLETE DESCRIPTION OF THE RUNTIME SERVICES, REFER TO THE XXDP+ USER'S MANUAL. THERE IS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES IN SECTION 2 OF THIS DOCUMENT.

1.2 SYSTEM REQUIREMENTS

IN ORDER TO RUN THE CZUAC NIE PROGRAM, THE FOLLOWING MINIMUM HARDWARE IS REQUIRED:

- A PDP-11 CPU
- MINIMUM OF 24K WORDS OF MEMORY
- A WORKING, LINE OR REAL-TIME CLOCK
- A CONSOLE TERMINAL
- ANY XXDP+ SUPPORTED LOAD MEDIA
- DEUNA-11 UNIBUS TO ETHERNET ADAPTER

1.3 RELATED DOCUMENTS AND STANDARDS

- DEUNA USER'S GUIDE EK-DEUNA-UG-001
- DEUNA TECHNICAL DESCRIPTION EK-DEUNA-TD-001
- XXDP+ USER'S MANUAL (CHQUS?.SEQ WHERE ? IS THE REV. LEVEL OF THE MANUAL - "C" IS THE CURRENT REV.)

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

THE GOAL OF THE NIE IS TO TEST THE COMMUNICATIONS LINK AND THEREFORE ASSUMES THAT THE CPU'S, CLOCKS, AND DEUNA'S AT EACH END OF THE LINK HAVE ALREADY BEEN TESTED.

IF NO LINE OR REAL-TIME CLOCK IS FOUND, THE PROGRAM WILL CONTINUE BUT ANY OF THE PROGRAM THAT TIMES THE DEVICE WILL HANG IF THE DEVICE TIMES OUT.

IT IS NOT THE INTENTION OF THE NIE TO TEST THE DEVICE (DEUNA), BUT TO TEST THE COMMUNICATIONS LINK TO WHICH IT IS CONNECTED.

THE PREREQUISITE DIAGNOSTICS ARE:

- CZUAAA0 DEUNA REPAIR LEVEL DIAGNOSTIC
- CZUABAO DEUNA FUNCTIONAL LEVEL DIAGNOSTIC

ALSO AVAILABLE FOR TEST IS:

- CXUACAO DEUNA DEC/X-11 MODULE

1.5 ASSUMPTIONS

IT IS ASSUMED THAT THE COMMUNICATIONS DEVICE (DEUNA) HAS BEEN TESTED USING THE PREREQUISITE DIAGNOSTICS. THE OPERATOR SHOULD HAVE READ THE USER DOCUMENTATION PORTION OF THE LISTING TO FAMILIARIZE HIMSELF WITH THE COMMANDS AND CAPABILITIES AVAILABLE UNDER THE DIAGNOSTIC SUPERVISOR AND NIE.

1.6 HISTORY

22-MARCH-84 NICK MCCAMY

THE SECTION "ALL" AND "LOOPPAIR" TESTS HAVE BEEN MODIFIED. THESE SECTIONS WOULD NOT RUN CORRECTLY WHEN NODES WERE DELETED FROM THE NODE TABLE PRIOR TO RUNNING. THE PROGRAM WAS TRYING TO SET-UP THE BLANK SPACE (LEFT BY THE DELETED NODE) AS A TARGET OR ASSIST NODE. THIS PROBLEM HAS BEEN CORRECTED.

2.0 OPERATING INSTRUCTIONS

THIS SECTION CONTAINS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES. FOR DETAILED INFORMATION, REFER TO THE XXDP+ USER'S MANUAL (CHQUS).

2.1 COMMANDS

THERE ARE ELEVEN LEGAL COMMANDS FOR THE DIAGNOSTIC RUNTIME SERVICES (SUPERVISOR). THIS SECTION LISTS THE COMMANDS AND GIVES A VERY BRIEF DESCRIPTION OF THEM. THE XXDP+ USER'S MANUAL HAS MORE DETAILS.

COMMAND	EFFECT
START	START THE DIAGNOSTIC FROM AN INITIAL STATE
RESTART	START THE DIAGNOSTIC WITHOUT INITIALIZING
CONTINUE	CONTINUE AT TEST THAT WAS INTERRUPTED (AFTER ^C)
PROCEED	CONTINUE FROM AN ERROR HALT
EXIT	RETURN TO XXDP+ MONITOR (XXDP+ OPERATION ONLY!)
ADD	ACTIVATE A UNIT FOR TESTING (ALL UNITS ARE CONSIDERED TO BE ACTIVE AT START TIME)
DROP	DEACTIVATE A UNIT
PRINT	PRINT STATISTICAL INFORMATION (IF IMPLEMENTED BY THE DIAGNOSTIC - SECTION 4.0)
DISPLAY	TYPE A LIST OF ALL DEVICE INFORMATION
FLAGS	TYPE THE STATE OF ALL FLAGS (SEE SECTION 2.3)
ZFLAGS	CLEAR ALL FLAGS (SEE SECTION 2.3)

A COMMAND CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS, SO YOU MAY, FOR EXAMPLE, TYPE "STA" INSTEAD OF "START".

2.2 SWITCHES

THERE ARE SEVERAL SWITCHES WHICH ARE USED TO MODIFY SUPERVISOR OPERATION. THESE SWITCHES ARE APPENDED TO THE LEGAL COMMANDS. ALL OF THE LEGAL SWITCHES ARE TABULATED BELOW WITH A BRIEF DESCRIPTION OF EACH. IN THE DESCRIPTIONS BELOW, A DECIMAL NUMBER IS DESIGNATED BY "DDDDD".

SWITCH	EFFECT
/TESTS:LIST	EXECUTE ONLY THOSE TESTS SPECIFIED IN THE LIST. LIST IS A STRING OF TEST NUMBERS, FOR EXAMPLE - /TESTS:1:5:7-10. THIS LIST WILL CAUSE TESTS 1,5,7,8,9,10 TO BE RUN. ALL OTHER TESTS WILL NOT BE RUN.
/PASS:DDDDD	EXECUTE DDDDD PASSES (DDDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP.DDDDD	REPORT END OF PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDDD = 1 TO 64000)
/UNITS:LIST	TEST/ADD/DROP ONLY THOSE UNITS SPECIFIED IN THE LIST. LIST EXAMPLE - /UNITS:0:5:10-12 USE UNITS 0,5,10,11,12 (UNIT NUMBERS = 0-63)

EXAMPLE OF SWITCH USAGE:

START/TESTS:1-5/PASS:1000/EOP:100

THE EFFECT OF THIS COMMAND WILL BE: 1) TESTS 1 THROUGH 5 WILL BE EXECUTED, 2) ALL UNITS WILL TESTED 1000 TIMES AND 3) THE END OF PASS MESSAGES WILL BE PRINTED AFTER EACH 100 PASSES ONLY. A SWITCH CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. YOU MAY, FOR EXAMPLE, TYPE "/TES:1-5" INSTEAD OF "/TESTS:1-5".

BELOW IS A TABLE THAT SPECIFIES WHICH SWITCHES CAN BE USED BY EACH COMMAND.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X
FLAGS					
ZFLAGS					
EXIT					

2.3 FLAGS

FLAGS ARE USED TO SET UP CERTAIN OPERATIONAL PARAMETERS SUCH AS LOOPING ON ERROR. ALL FLAGS ARE CLEARED AT STARTUP AND REMAIN CLEARED UNTIL EXPLICITLY SET USING THE FLAGS SWITCH. FLAGS ARE ALSO CLEARED AFTER A START COMMAND UNLESS SET USING THE FLAG SWITCH. THE ZFLAGS COMMAND MAY ALSO BE USED TO CLEAR

ALL FLAGS. WITH THE EXCEPTION OF THE START AND ZFLAGS COMMANDS, NO COMMANDS AFFECT THE STATE OF THE FLAGS; THEY REMAIN SET OR CLEARED AS SPECIFIED BY THE LAST FLAG SWITCH.

FLAG	EFFECT
HOE	HALT ON ERROR - CONTROL IS RETURNED TO RUNTIME SERVICES COMMAND MODE
LOE	LOOP ON ERROR
IER*	INHIBIT ALL ERROR REPORTS
IBE*	INHIBIT ALL ERROR REPORTS EXCEPT FIRST LEVEL (FIRST LEVEL CONTAINS ERROR TYPE, NUMBER, PC, TEST AND UNIT)
IXE*	INHIBIT EXTENDED ERROR REPORTS (THOSE CALLED BY PRINTX MACRO'S)
PRI	DIRECT MESSAGES TO LINE PRINTER
PNT	PRINT TEST NUMBER AS TEST EXECUTES
BOE	"BELL" ON ERROR
UAM	UNATTENDED MODE (NO MANUAL INTERVENTION)
ISR	INHIBIT STATISTICAL REPORTS (DOES NOT APPLY TO DIAGNOSTICS WHICH DO NOT SUPPORT STATISTICAL REPORTING)
IDR	INHIBIT PROGRAM DROPPING OF UNITS
ADR	EXECUTE AUTODROP CODE
LOT	LOOP ON TEST
EVL	EXECUTE EVALUATION (ON DIAGNOSTICS WHICH HAVE EVALUATION SUPPORT)

*ERROR MESSAGES ARE DESCRIBED IN SECTION 3.1

SEE THE XXDP* USER'S MANUAL FOR MORE DETAILS ON FLAGS. YOU MAY SPECIFY MORE THAN ONE FLAG WITH THE FLAG SWITCH. FOR EXAMPLE, TO CAUSE THE PROGRAM TO LOOP ON ERROR, INHIBIT ERROR REPORTS AND TYPE A "BELL" ON ERROR, YOU MAY USE THE FOLLOWING STRING:

```
/FLAGS:LOE:IER:BOE
```

2.4 HARDWARE QUESTIONS

WHEN A DIAGNOSTIC IS STARTED, THE RUNTIME SERVICES WILL PROMPT THE USER FOR HARDWARE INFORMATION BY TYPING "CHANGE HW (L) ?" YOU MUST ANSWER "Y" AFTER A START COMMAND UNLESS THE HARDWARE INFORMATION HAS BEEN "PRELOADED" USING THE SETUP UTILITY (SEE CHAPTER 6 OF THE XXDP* USER'S MANUAL). WHEN YOU ANSWER THIS QUESTION WITH A "Y", THE RUNTIME SERVICES WILL ASK FOR THE NUMBER OF UNITS (IN DECIMAL). YOU WILL THEN BE ASKED THE FOLLOWING QUESTIONS FOR EACH UNIT.

```
* UNITS (0) ? 1<CR>

UNIT 0
DEVICE CSR ADDRESS : (0) 164524 ?<CR>
INTERRUPT VECTOR ADDRESS : (0) 120 ?<CR>
INTERRUPT PRIORITY : (0) 5 ?<CR>
```

WHEN YOU COMPLETE THE ABOVE SEQUENCE YOU WILL BE AT THE NIE>
COMMAND LEVEL.

NIE> (A) ?

2.5 NETWORK INTERCONNECT EXERCISER COMMANDS

THE "NIE>" COMMAND LEVEL FOLLOWS THE ATTACHING OF THE DEVICE AND ISSUING THE START TO THE SUPERVISOR. THESE COMMANDS CAN BE TYPED WHEN THE "NIE>" PROMPT IS PRINTED.

YOU ONLY HAVE TO TYPE ENOUGH CHARACTERS TO UNIQUELY SPECIFY A COMMAND. THE COMMAND LINE IS INTERPRETED FROM LEFT TO RIGHT.

HELP OR ? PRINTS OUT A BRIEF DESCRIPTION OF NIE COMMANDS.

SHOW NODES PRINTS OUT THE CONTENTS OF THE NODE TABLE.

SHOW MESSAGE PRINTS OUT THE CURRENT MESSAGE PARAMETERS FOR SIZE, TYPE AND COPIES.

SHOW COUNTERS PRINTS OUT THE CONTENTS OF THE HOST NODE DEUNA INTERNAL COUNTERS.

NODE ADR/TYPE THE NODE COMMAND ALLOWS THE OPERATOR TO ENTER NODES INTO THE NODE TABLE. NODES ARE SPECIFIED USING THEIR 12 HEX DIGIT ETHERNET PHYSICAL ADDRESS AND CAN BE SPECIFIED AS EITHER TARGET OR ASSIST (A DEFAULT OF TARGET IS ASSUMED).

MESSAGE/TYPE=/
SIZE=N/COPIES=M THE MESSAGE COMMAND ALLOWS THE OPERATOR TO SELECT THE CURRENT MESSAGE PARAMETERS AS FOLLOWS. ANY OR ALL OF THE PARAMETER CAN BE CHANGED WITH THE COMMAND. THE DEFAULT PARAMETERS ARE TYPE=ALPHA, SIZE=512, COPIES=1.

TYPE ONE OF THE FOLLOWING MESSAGE TYPES:

ALPHA . !"#\$%&'()*+,-./0123456789;:~?ABCDEFGH ETC.
ONES -- MESSAGE OF ALL ONES (11111111....)
ZEROS -- MESSAGE OF ALL ZEROS (00000000....)
1ALT -- ALTERNATING 1'S AND 0'S (10101010...)
0ALT -- ALTERNATING 0'S AND 1'S (01010101...)
CCITT -- "CCITT" PSEUDO-RANDOM TEST PATTERN
OPERATOR SELECTED -- OPERATOR CHOSEN PATTERN OF LESS THAN 72 CHARACTERS USING 0-9, A-Z AND SPACES. (NOT USED IN PATTERN TEST)

SIZE THE SIZE OF THE MESSAGE BUFFER (DATA ONLY) MAY BE BETWEEN 32 AND 1466 BYTES.

COPIES THE NUMBER OF COPIES OF EACH MESSAGE SENT TO EACH NODE DURING A TEST MAY BE BETWEEN 1 AND 255.

RUN TEST/PASS=NN CAUSES EXECUTION OF THE SPECIFIED TEST FOR NN NUMBER OF PASSES. A DEFAULT VALUE OF 1 IS ASSUMED IF /PASS=NN IS

NOT INCLUDED IN THE COMMAND LINE. A VALUE OF NN=-1 WILL CAUSE THE TEST TO BE RUN INDEFINATELY. NODE ADDRESSES FOR THE TESTS ARE TAKEN FROM THE NODE TABLE AND SHOULD BE ENTERED PRIOR TO RUNNING THE TEST USING THE NODE COMMAND. IN THE CASE OF THE LOOPPAIR TEST, NODE PAIRS ARE REQUIRED AND MUST BE SPECIFIED AS TARGET AND ASSIST NODES. THE CURRENTLY SELECTED VALUES FOR MESSAGE TYPE, SIZE AND COPIES ARE USED BY EACH TEST.

THERE ARE FOUR TEST TO CHOSE FROM:

DIRECT

THE DIRECT TEST SENDS A LOOP DIRECT MESSAGE TO ALL OF THE NODES CONTAINED IN THE NODE TABLE AND WAITS FOR A RESPONSE. THE INTEGRITY OF THE RETURNED DATA IS CHECKED AND ANY ERRORS ARE REPORTED TO THE OPERATOR.

LOOPPAIR

THE LOOPPAIR TEST SENDS ASSISTED LOOPBACK MESSAGES TO THE NODE PAIRS CONTAINED IN THE NODE TABLE. THREE TYPE OF ASSISTED MESSAGES ARE SENT:

- 1) RECEIVE ASSIST -- HOST -> TARGET -> ASSIST -> HOST
- 2) TRANSMIT ASSIST -- HOST -> ASSIST -> TARGET -> HOST
- 3) FULL ASSIST -- HOST -> ASSIST -> TARGET -> ASSIST -> HOST

IN EACH CASE A RESPONSE IS WAITED FOR AND THE DATA IS CHECKED.

WHEN RUNNING THE LOOPPAIR TEST, BE VERY CAREFUL WHEN DELETING NODES. IT IS BETTER TO DELETE NODES IN PAIRS (ONE TARGET NODE AND ONE ASSIST NODE).

*** IMPORTANT!! *** ---> THE LOOPPAIR TEST EXPECTS THAT EACH TARGET NODE HAVE A CORRESPONDING ASSIST NODE, AND THAT THE ORDER OF THE NODES IN THE NODE TABLE BE THE FOLLOWING:

FIRST NODE IN TABLE: TARGET
 2ND NODE IN TABLE: ASSIST
 3RD NODE IN TABLE: TARGET
 4TH NODE IN TABLE: ASSIST
 ETC.

BE VERY CARFFUL WHEN DELETING NODES. IT IS BEST TO DELETE NODES IN PAIRS (ONE TARGET NODE AND ONE ASSIST NODE).

PATTERN

THE PATTERN TEST SENDS SIX DIFFERENT LOOP DIRECT MESSAGES TO EACH NODE CONTAINED IN THE NODE TABLE. EACH OF THE SIX PATTERN TYPES (ALPHA, ONES, ZEROS, 1ALT, 0ALT, CCITT) IS USED FOR EACH NODE. RETURNED DATA IS CHECKED FOR ERRORS.

ALL

THE ALL NODE TEST PERFORMS THE MOST EXTENSIVE CHECK OF THE NETWORK AND IS COMPOSED OF TWO PARTS. FIRST A LOOP DIRECT MESSAGE IS SENT TO EACH NODE IN THE TABLE. IF THIS IS SUCCESSFUL, THE EXERCISER BUILDS AN ARRAY OF NODE PAIRS FROM THE TABLE AND SENDS A FULL ASSISTED LOOPBACK MESSAGE TO EACH PAIR IN THE ARRAY. A SAMPLE ARRAY OF PAIRS FOR A TABLE WITH 7 NODES IS SHOWN BELOW.

1-2	2-3	3-4	4-5	5-6	6-7
1-3	2-4	3-5	4-6	5-7	
1-4	2-5	3-6	4-7		
1-5	2-6	3-7			
1-6	2-7				
1-7					

- IDENTIFY ADR A REQUEST ID MESSAGE IS SENT TO THE NODE SPECIFIED BY ADR AND THE RESPONDED SYSTEM ID PARAMETERS ARE PRINTED.
- BUILD THE BUILD COMMAND CAUSES THE EXERCISERS TO LISTEN FOR SYSTEM ID MESSAGES WHICH ARE BROADCAST BY ALL DEUNA NODES ONCE EVERY 10 MINUTES. ALL NODES IDENTIFYING THEMSELVES ARE ADDED TO THE NODE TABLE. THE BUILD COMMAND STOPS WHEN NO NEW NODES HAVE BEEN ADDED FOR 10 MINUTES OR WHEN 40 MINUTES HAVE ELAPSED. THE AVERAGE TIME FOR THIS COMMAND SHOULD BE 15-25 MINUTES.
- CLEAR NODE/ADR THE CLEAR NODE COMMAND CLEARS THE SPECIFIED NODE FROM THE NODE TABLE. THE NODE CAN BE SPECIFIED BY EITHER ITS 12 DIGIT PHYSICAL ADDRESS OR ITS LOGICAL NAME (AS ASSIGNED BY NODE TABLE).
- CLEAR NODE/ALL THIS COMMAND CLEARS THE NODE TABLE.
- CLEAR MESSAGE THIS COMMAND SETS THE MESSAGE PARAMETERS BACK TO THE DEFAULT VALUES.
- CLEAR SUMMARY THIS COMMAND CLEARS THE SUMMARY TABLE.
- SUMMARY THE SUMMARY COMMAND PRINTS OUT THE SUMMARY TABLE. THE NIE MAINTAINS THE FOLLOWING INFORMATION FOR NODES WHO HAVE BEEN SENT MESSAGES:

RECEIVES NOT COMPLETE	RECEIVES COMPLETE
LENGTH ERRORS	DATA COMPARE ERRORS
BYTES COMPARED	BYTES TRANSFERED
- SAVE THE SAVE COMMAND SAVES THE CONTENTS OF THE NODE TABLE. FOR THE VAX VERSION, THE TABLE IS SAVED IN A FILE CALLED NIE.TBL. THE PDP-11 VERSION CANNOT WRITE TO EXTERNAL MEDIA, SO THE CONTENTS ARE SAVE INTERNALLY.
- UNSAVE THE UNSAVE COMMAND RESTORES THE CONTENTS OF THE NODE TABLE. USED. THE PDP-11 VERSION USES THE CONTENTS OF ITS INTERNALLY SAVED TABLE.
- UNSAVE/FILE.EXT THE UNSAVE COMMAND WHEN USED WITH A FILE NAME WILL READ A NODE TABLE CREATED BY USING AN EDITOR FOR AN XXDP+ MEDIA.
- EXIT RETURNS CONTROL TO THE DIAGNOSTIC SUPERVISOR (EITHER VDS OR DRS).

NOTES: 1) ADR IS THE PHYSICAL ADDRESS OF A NODE ON THE NI.
2) PASS COUNT IS A DECIMAL NUMBER BETWEEN 1 AND 65534.

A DEFAULT VALUE OF 1 IS ASSUMED. SPECIFYING -1 CAUSES THE TEST TO BE RUN INDEFINATELY.

2.5 SOFTWARE QUESTIONS

AFTER YOU HAVE ANSWERED THE HARDWARE QUESTIONS OR AFTER A RESTART OR CONTINUE COMMAND, THE RUNTIME SERVICES WILL ASK FOR SOFTWARE PARAMETERS. THESE PARAMETERS WILL GOVERN SOME DIAGNOSTIC SPECIFIC OPERATION MODES. YOU WILL BE PROMPTED BY "CHANGE SW (L) ?" IF YOU WISH TO CHANGE ANY PARAMETERS, ANSWER BY TYPING "Y". THE SOFTWARE QUESTIONS AND THE DEFAULT VALUES ARE DESCRIBED IN THE NEXT PARAGRAPH(S).

2.6 QUICK START-UP PROCEDURE (XXDP+)

TO START-UP THIS PROGRAM:

1. BOOT XXDP+
2. GIVE THE DATE AND ANSWER THE LSI AND 50HZ (IF THERE IS A CLOCK) QUESTIONS
3. TYPE "R NAME", WHERE NAME IS THE NAME OF THE BIN OR BIC FILE FOR THIS PROGRAM
4. TYPE "START"
5. ANSWER THE "CHANGE HW" QUESTION WITH "Y"
6. ANSWER ALL THE HARDWARE QUESTIONS
7. ANSWER THE "CHANGE SW" QUESTION WITH "N"

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS AND SOFTWARE PARAMETERS. THESE DEFAULTS ARE DESCRIBED IN SECTIONS 2.3 AND 2.5.

3.0 ERROR INFORMATION

3.1 TYPES OF ERROR MESSAGES

THERE ARE THREE LEVELS OF ERROR MESSAGES THAT MAY BE ISSUED BY A DIAGNOSTIC: GENERAL, BASIC AND EXTENDED. GENERAL ERROR MESSAGES ARE ALWAYS PRINTED UNLESS THE "IER" FLAG IS SET (SECTION 2.3). THE GENERAL ERROR MESSAGE IS OF THE FORM:

```
NAME TYPE NUMBER ON UNIT NUMBER TST NUMBER PC:XXXXXX
ERROR MESSAGE
```

WHERE; NAME = DIAGNOSTIC NAME
 TYPE = ERROR TYPE (SYS FATAL, DEV FATAL, HARD OR SOFT)
 NUMBER = ERROR NUMBER
 UNIT NUMBER = 0 - N (N IS LAST UNIT IN PTABLE)
 TST NUMBER = TEST AND SUBTEST WHERE ERROR OCCURRED
 PC:XXXXXX = ADDRESS OF ERROR MESSAGE CALL

BASIC ERROR MESSAGES ARE MESSAGES THAT CONTAIN SOME ADDITIONAL INFORMATION ABOUT THE ERROR. THESE ARE ALWAYS PRINTED UNLESS THE "IER" OR "IBR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL MESSAGE.

EXTENDED ERROR MESSAGES CONTAIN SUPPLEMENTARY ERROR INFORMATION SUCH AS REGISTER CONTENTS OR GOOD/BAD DATA. THESE ARE ALWAYS PRINTED UNLESS THE "IIR", "IBR" OR "IXR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL ERROR MESSAGE AND ANY ASSOCIATED BASIC ERROR MESSAGES.

3.2 SPECIFIC ERROR MESSAGES

ERROR MESSAGE:	MEANING
-----	-----
?ILL CMD-BAD SYNTAX	A COMMAND WITH AN ILLEGAL CHAR WAS TYPED - RETYPE THE COMMAND. THE VAILD COMMANDS AND THEIR SYNTAX ARE SHOWN IN SECTION 2.5
?INCOMPLETE	A REQUIRED PART OF A COMMAND WAS LEFT OUT.
?NUMBER TOO BIG	THE VALUE OF A NUMERIC STRING IN THE COMMAND LINE WAS LARGER THAN 65535 OR 177777 OCTAL. (>16 BITS).
?BAD RADIX	A "8" OR "9" WAS TYPED WHEN AN OCTAL STRING WAS EXPECTED. PROBABLY OCCURED WHEN TYPING A "DUMP" COMMAND WHERE OCTAL ADDRESSES ARE EXPECTED.

EXAMPLE OF A LOST PACKET ERROR DURING LOOPPAIR TESTING

 CZUAC HRD ERR 00028 ON UNIT 00 TST 001 SUB 000 PC:064442

TIMEOUT OCCURED - LOOP 1. PAGE TYPE - RECEIVE ASSIST
 FAILING TARGET NODE ADDRESS: AA-00-03-00-00-00
 FAILING ASSIST NODE ADDRESS: AA-00-03-00-00-00

EXAMPLE OF A LOST PACKET ERROR DURING PATTERN TESTING

 CZUAC HRD ERR 00028 ON UNIT 00 TST 001 SUB 000 PC:63730

TIMEOUT OCCURED BEFORE LOOPBACK REPLY
 FAILING NODE ADDRESS: AA-00-03-00-00-00
 DATA PATTERN: ONES

4.0 PERFORMANCE AND PROGRESS REPORTS

AT THE END OF EACH PASS, THE PASS COUNT IS GIVEN ALONG WITH THE TOTAL NUMBER OF ERRORS REPORTED SINCE THE DIAGNOSTIC WAS STARTED. THE "EOP" SWITCH CAN BE USED TO CONTROL HOW OFTEN THE END OF PASS MESSAGE IS PRINTED. SECTION 2.2 DESCRIBES SWITCHES.

5.0 DEVICE INFORMATION TABLES

THIS IS THE DEFAULT HARDWARE P-TABLE. THE VALUES AND SIZE ARE USED AS A "TEMPLATE" FOR CREATING ACTUAL P-TABLE ENTRIES AND THE DEFAULT VALUES PROVIDED FOR THE OPERATOR. SEE SECTION 2.4 FOR AN EXAMPLE OF THE HARDWARE QUESTIONS.

THE NUMBERS IN BRACKETS (I.E. [6]) INDICATES THE OFFSET OF THE WORD INTO THE HARDWARE P-TABLE. THE OFFSETS MUST MATCH THE P-TABLE OFFSETS USED IN THE HARDWARE PARAMETER CODING SECTION WHERE THE "GET PARAMETER" CALLS ARE USED TO FILL THE P-TABLE.

```
.WORD 174510      ;[0] CSR ADDRESS
.WORD 120        ;[2] INTERRUPT VECTOR
.WORD 240        ;[6] INTERRUPT PRIORITY (5)
```

6.0 TEST SUMMARIES

6.1 DIRECT

THE DIRECT TEST SENDS A LOOP DIRECT MESSAGE TO ALL OF THE NODES CONTAINED IN THE NODE TABLE AND WAITS FOR A RESPONSE. THE INTEGRITY OF THE RETURNED DATA IS CHECKED AND ANY ERRORS ARE REPORTED TO THE OPERATOR.

6.2 LOOPPAIR

THE LOOPPAIR TEST SENDS ASSISTED LOOPBACK MESSAGES TO THE NODE PAIRS CONTAINED IN THE NODE TABLE. THREE TYPE OF ASSISTED MESSAGES ARE SENT:

- 1) RECEIVE ASSIST -- HOST -> TARGET -> ASSIST -> HOST
- 2) TRANSMIT ASSIST -- HOST -> ASSIST -> TARGET -> HOST
- 3) FULL ASSIST -- HOST -> ASSIST -> TARGET -> ASSIST -> HOST

IN EACH CASE A RESPONSE IS WAITED FOR AND THE DATA IS CHECKED.

6.3 PATTERN

THE PATTERN TEST SENDS SIX DIFFERENT LOOP DIRECT MESSAGES TO EACH NODE CONTAINED IN THE NODE TABLE. EACH OF THE SIX PATTERN TYPES (ALPHA, ONES, ZEROS, 1ALT, 0ALT, CCITT) IS USED FOR EACH NODE. RETURNED DATA IS CHECKED FOR ERRORS.

6.4 ALL

THE ALL NODE TEST PERFORMS THE MOST EXTENSIVE CHECK OF THE NETWORK AND IS COMPOSED OF TWO PARTS. FIRST A LOOP DIRECT MESSAGE IS SENT TO EACH NODE IN THE TABLE. IF THIS IS SUCCESSFUL, THE EXERCISER BUILDS AN ARRAY OF NODE PAIRS FROM THE TABLE AND SENDS A FULL ASSISTED LOOPBACK MESSAGE TO EACH PAIR IN THE ARRAY. A SAMPLE ARRAY OF PAIRS FOR A TABLE WITH 7 NODES IS SHOWN BELOW.

1-2	2-3	3-4	4-5	5-6	6-7
1-3	2-4	3-5	4-6	5-7	
1-4	2-5	3-6	4-7		
1-5	2-6	3-7			
1-6	2-7				
1-7					

ε

```

683
709
710 000000
711          002000
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754

```

```

.SBTTL PROGRAM HEADER
.ENABL ABS,AMA
      *      2000

.SBTTL PROGRAM MACROS

;I$STACK MACRO
;-----
;+++
;THE I$STACK MACRO FACILITATES INITIALIZING THE R6 (HARDWARE) STACK
;AND THE R5 (PARAMETER) STACK. R5 IS SET TO THE STACK LOW LIMIT
;(STAKLO) AND THE PARAMETER STACK GROWS UPWARD. R6 IS SET TO THE
;STACK HIGH LIMIT (STAKHI) AND THE HARDWARE STACK GROWS DOWNWARD.
;IF THERE IS A STACK OVER-RUN, IT WILL BE DETECTED BY THE PREG14
;ROUTINE.
;----
.MACRO I$STACK STAKLO,STAKHI
MOV     STAKLO,R5      ;INITIALIZE THE PARAMETER STACK POINTER.
MOV     STAKHI,SP     ;INITIALIZE THE HARDWARE STACK POINTER.
.ENDM   I$STACK

;PUSH MACRO
;-----
;+++
;THE "PUSH" MACRO FACILITATES PUSHING ITEMS ON THE HARDWARE STACK.
;UP TO SEVEN ITEMS MAY BE PLACED ON THE STACK WITH ONE MACRO.
;----
.MACRO PUSH  A,B,C,D,E,F,G
      .IF NB G
MOV     G,-(SP)
      .ENDC
      .IF NB F
MOV     F,-(SP)
      .ENDC
      .IF NB E
MOV     E,-(SP)

```

```

755      .ENOC
756
757      .IF NB D
758      MOV     D,-(SP)
759      .ENOC
760
761      .IF NB C
762      MOV     C,-(SP)
763      .ENOC
764
765      .IF NB B
766      MOV     B,-(SP)
767      .ENOC
768
769      .IF NB A
770      MOV     A,-(SP)
771      .ENOC
772
773      .ENDM   PUSH
774
775      ;POP MACRO
776      |-----
777
778      |***
779      |THE "POP" MACRO FACILITATES RETRIEVING ITEMS FROM THE HARDWARE STACK.
780      |UP TO SEVEN ITEMS MAY BE RETRIEVED WITH ONE MACRO.
781      |----
782
783      .MACRO  POP      A,B,C,D,E,F,G
784
785      .IF NB A
786      MOV     (SP)+,A
787      .ENOC
788
789      .IF NB B
790      MOV     (SP)+,B
791      .ENOC
792
793      .IF NB C
794      MOV     (SP)+,C
795      .ENOC
796
797      .IF NB D
798      MOV     (SP)+,D
799      .ENOC
800
801      .IF NB E
802      MOV     (SP)+,E
803      .ENOC
804
805      .IF NB F
806      MOV     (SP)+,F
807      .ENOC
808
809      .IF NB G
810      MOV     (SP)+,G
811      .ENOC

```

```

812
813 .ENDM FOP
814
815 ;CALL MACRO
816 ;-----
817
818 ;***
819 ;THE CALL MACRO FACILITATES CALLING A SUBROUTINE VIA THE REGISTER
820 ;PRESERVE ROUTINE (PREG14). IT PLACES THE PARAMETERS TO BE PASSED ON
821 ;THE PARAMETER STACK. UP TO 7 PARAMETERS MAY BE PASSED USING THIS
822 ;MACRO.
823 ;---
824
825 .MACRO CALL S A,B,C,D,E,F,G
826
827 .IF NB G
828 MOV G,(R5)+
829 .ENOC
830
831 .IF NB F
832 MOV F,(R5)+
833 .ENOC
834
835 .IF NB E
836 MOV E,(R5)+
837 .ENOC
838
839 .IF NB D
840 MOV D,(R5)+
841 .ENOC
842
843 .IF NB C
844 MOV C,(R5)+
845 .ENOC
846
847 .IF NB B
848 MOV B,(R5)+
849 .ENOC
850
851 .IF NB A
852 MOV A,(R5)+
853 .ENOC
854
855 JSR R4,PREG14
856 .WORD S-ANCHOR
857
858 .ENDM CALL
859
860 ;RETURN MACRO
861 ;-----
862
863 ;***
864 ;THE RETURN MACRO FACILITATES PASSING PARAMETERS BACK TO A CALLING
865 ;ROUTINE. UP TO 7 PARAMETERS MAY BE PASSED BACK ON THE PARAMETER
866 ;STACK.
867 ;---
868

```

```

869      .MACRO RETURN A,B,C,D,E,F,G
870
871      .IF NB G
872      MOV     G,(R5)+
873      .ENDC
874
875      .IF NB F
876      MOV     F,(R5)+
877      .ENDC
878
879      .IF NB E
880      MOV     E,(R5)+
881      .ENDC
882
883      .IF NB D
884      MOV     D,(R5)+
885      .ENDC
886
887      .IF NB C
888      MOV     C,(R5)+
889      .ENDC
890
891      .IF NB B
892      MOV     B,(R5)+
893      .ENDC
894
895      .IF NB A
896      MOV     A,(R5)+
897      .ENDC
898
899      RTS     PC
900
901      .ENDM   RETURN
902
903      ;P$PUSH MACRO
904      ;-----
905
906      ;***
907      ;THE P$PUSH MACRO FACILITATES PUSHING PARAMETERS ON THE PARAMETER
908      ;STACK. UP TO SEVEN ITEMS MAY BE PUSHED WITH ONE MACRO.
909      ;----
910
911      .MACRO P$PUSH A,B,C,D,E,F,G
912
913      .IF NB G
914      MOV     G,(R5)+
915      .ENDC
916
917      .IF NB F
918      MOV     F,(R5)+
919      .ENDC
920
921      .IF NB E
922      MOV     E,(R5)+
923      .ENDC
924
925      .IF NB D

```



```

926      MOV     D,(R5)+
927      .ENDC
928
929      .IF NB C
930      MOV     C,(R5)+
931      .ENDC
932
933      .IF NB B
934      MOV     B,(R5)+
935      .ENDC
936
937      .IF NB A
938      MOV     A,(R5)+
939      .ENDC
940
941      .ENDM   P$PUSH
942
943      ;P$POP MACRO
944      ;-----
945
946
947
948      ;***
949      ;THE P$POP MACRO FACILITATES RETRIEVING PARAMETERS FROM THE PARAMETER
950      ;STACK.  UP TO 7 PARAMETERS MAY BE RETRIEVED.
951      ;
952      ;THE ROUTINE THAT RECEIVES THE PARAMETERS HAS THE RESPONSIBILITY OF
953      ;CLEANING UP THE PARAMETER STACK.  THIS MACRO IS AN AID TO MAKING
954      ;A LOCAL COPY OF PASSED PARAMETERS AND CLEANING UP THE PARAMETER STACK.
955      ;----
956
957      .MACRO P$POP  A,B,C,D,E,F,G
958
959      .IF NB A
960      MOV     -(R5),A
961      .ENDC
962
963      .IF NB B
964      MOV     -(R5),B
965      .ENDC
966
967      .IF NB C
968      MOV     -(R5),C
969      .ENDC
970
971      .IF NB D
972      MOV     -(R5),D
973      .ENDC
974
975      .IF NB E
976      MOV     -(R5),E
977      .ENDC
978
979      .IF NB F
980      MOV     -(R5),F
981      .ENDC
982
983      .IF NB G

```

```

983
984          MOV      -(R5),G
985          .ENDC
986          .ENDM   P#POP
987
988
993
994          .MACRO  CLI      CHAR,HITVAL,MISADR,CMPSTR
995          NODCL   CHAR,HITVAL,\X#,MISADR,CMPSTR    ;,#### PARSE TREE ####
996          .ENDM
997
998
999          .MACRO  NODCL   CHAR,HITVAL,XY,MISADR,CMPSTR
1000 NOD'XY: .BYTE   CHAR,HITVAL                      ;SPECIAL CHAR. CODE OR COMPARE CHAR.
1001                                           ; AND ACTION (HIT) VALUE FOR ACTION
1002                                           ; ROUTINES.
1003
1004          .IF NB  MISADR
1005          .WORD   MISADR-NOD'XY                    ;DISPLACEMENT TO "MISS" NODE (BYTES)
1006          .ENDC
1007          .IF NB  CMPSTR
1008          .WORD   1#-NOD'XY                        ;DISPLACEMENT TO GET TO NEXT NODE
1009          .ASCIZ  CMPSTR                          ; (ONLY IF ITS A "CLISTR" NODE)
1010          .EVEN
1011          .NLIST
1012          1#:
1013          .LIST
1014          .ENDC
1015          .NLIST
1016          X#-X#+1
1017          .LIST
1018          .ENDM
1019
1020          .MACRO  RNGFRM A,B,C
1021                                           ; MACRO TO FORM TRANSMIT AND RECIEVE
1022                                           ; DESCRIPTOR RINGS.
1023          .LIST
1024          .WORD   RPKLEN
1025          .NLIST
1026          NEXT   A,\B
1027          B=B+1
1028          .LIST
1029          .WORD   C
1030          .WORD   0
1031          .WORD   0
1032          .NLIST
1033          .ENDM
1034
1035          .MACRO  NEXT A,B
1036          .LIST
1037          .WORD   A'B
1038          .NLIST
1039          .ENDM
1040
1041          ;**
1042          ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
1043          ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.

```

```

1044
1045
1046 002000          POINTER BGNRPT
1047
1064
1065 002000          HEADER CZUAC,A,O,O,1,PRI07
002000
002000          103
002001          132
002002          125
002003          101
002004          103
002005          000
002006          000
002007          000
002010
002010          101
002011
002011          060
002012
002012 000000
002014
002014 000000
002016
002016 112762
002020
002020 000000
002022
002022 002170
002024
002024 000000
002026
002026 113166
002030
002030 000000
002032
002032 000000
002034
002034 000001
002036
002036 000000
002040
002040 002164
002042
002042 000340
002044
002044 000000
002046
002046 000000
002050
002050          003
002051          003
002052
002052 000000
002054 000000
002056
002056 000000

```

```

L$NAME::
          .ASCII /C/
          .ASCII /Z/
          .ASCII /U/
          .ASCII /A/
          .ASCII /C/
          .BYTE 0
          .BYTE 0
          .BYTE 0
L$REV::
          .ASCII /A/
L$DEPO::
          .ASCII /O/
L$UNIT::
          .WORD 0
L$TIM::
          .WORD 0
L$HPCP::
          .WORD L$HARD
L$SPCP::
          .WORD 0
L$HPTP::
          .WORD L$HW
L$SPTP::
          .WORD 0
L$LADP::
          .WORD L$LAST
L$STA::
          .WORD 0
L$CO::
          .WORD 0
L$DTYP::
          .WORD 1
L$APT::
          .WORD 0
L$DTP::
          .WORD L$DISPATCH
L$PRIO::
          .WORD PRI07
L$ENVI::
          .WORD 0
L$EXP1::
          .WORD 0
L$MREV::
          .BYTE C$REVISION
          .BYTE C$EDIT
L$EF::
          .WORD 0
          .WORD 0
L$SPC::
          .WORD 0

```

002060
 002060 002122
 002062
 002062 076560
 002064
 002064 000000
 002066
 002066 000000
 002070
 002070 000000
 002072
 002072 000000
 002074
 002074 000000
 002076
 002076 002130
 002100
 002100 104035
 002102
 002102 000000
 002104
 002104 076600
 002106
 002106 100142
 002110
 002110 100140
 002112
 002112 076572
 002114
 002114 000000
 002116
 002116 000000
 002120
 002120 000000

1066
 1077
 1078
 1079
 1080

```

;
; NAMES OF DEVICES SUPPORTED BY PROGRAM
;
  DEVTYP <DEUNA>

```

1081 002122
 002122
 002122 104 105 125
 002125 116 101 000

1082
 1088
 1089
 1090
 1091

```

;
; TEST DESCRIPTION
;
  DESCRIPT <CZUAC DEUNA NI EXERCISER>

```

002130
 002130
 002130 103 132 125
 002133 101 103 040
 002136 104 105 125
 002141 116 101 040
 002144 116 111 040
 002147 105 130 105
 002152 122 103 111

CISER/

L\$DEVP:: .WORD L\$DVTYP
 L\$REPP:: .WORD L\$RPT
 L\$EXP4:: .WORD 0
 L\$EXP5:: .WORD 0
 L\$AUT:: .WORD 0
 L\$DUT:: .WORD 0
 L\$LUN:: .WORD 0
 L\$DESP:: .WORD L\$DESC
 L\$LOAD:: EMT E\$LOAD
 L\$ETP:: .WORD 0
 L\$ICP:: .WORD L\$INIT
 L\$CCP:: .WORD L\$CLEAN
 L\$ACP:: .WORD L\$AUTO
 L\$PRT:: .WORD L\$PRGT
 L\$TEST:: .WORD 0
 L\$DLY:: .WORD 0
 L\$HIME:: .WORD 0

L\$DVTYP:: .ASCIZ /DEUNA/
 .EVEN

L\$DESC:: .ASCIZ /CZUAC DEUNA NI EXER

002155 123 105 122
002160 000

1092
1093
1100
1101
1102
1103
1104
1115
1116

.EVEN

.EVEN

;
; FORMAT STATEMENTS USED IN PRINT CALLS
;

J2

1125
1126
1127
1128
1129
1130
1131
1132
1133

002162
002162 000001
002164
002164 100274

.SBTTL DISPATCH TABLE

; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.

DISPATCH 1

.WORD 1
L\$DISPATCH::
.WORD T1

```

1141          .SBTTL  DEFAULT HARDWARE P-TABLE
1142
1143          ;**
1144          ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
1145          ; THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
1146          ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
1147          ; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
1148          ;--
1149
1150 002166          BGNHW  DFPTBL.
1151          002166  000003
1152          002170
1153          002170
1154          002170
1155
1156          .WORD  174510          ; CSR
1157          .WORD  120             ; VECTOR
1158          .WORD  PRI05          ; PRIORITY
1159
1160          ENDPHW
1161
1162          L$HW:: .WORD  L10000-L$HW/2
1163          DFPTBL::
1164
1165          L10000:

```

```

1169
1170
1171 .SBTTL SOFTWARE P-TABLE
1172
1173 ;**
1174 ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
1175 ; PROGRAM AS OPERATIONAL PARAMETERS. THESE PARAMETERS ARE
1176 ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
1177 ; AT RUN TIME.
1178 ;--
1179 002176          BGNSW   SFPTBL
1180                .WORD   L10001-L$SW/2
1181                SFPTBL::
1182
1183
1184
1185
1186
1187
1188
1189 002200          ENDSW
1190
1191
1192 .SBTTL GLOBAL EQUATES SECTION
1193
1194
1195
1196
1197
1198
1199
1200
1201 ;**
1202 ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
1203 ; ARE USED IN MORE THAN ONE TEST.
1204 ;--
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224 002200

```

100000	BIT15== 100000
040000	BIT14== 40000
020000	BIT13== 20000
010000	BIT12== 10000
004000	BIT11== 4000
002000	BIT10== 2000
001000	BIT09== 1000
000400	BIT08== 400
000200	BIT07== 200
000100	BIT06== 100
000040	BIT05== 40
000020	BIT04== 20
000010	BIT03== 10
000004	BIT02== 4
000002	BIT01== 2
000001	BIT00== 1
001000	BIT9== BIT09
000400	BIT8== BIT08
000200	BIT7== BIT07
000100	BIT6== BIT06
000040	BIT5== BIT05
000020	BIT4== BIT04
000010	BIT3== BIT03


```

000004      BIT2== BIT02
000002      BIT1== BIT01
000001      BIT0== BIT00
;
; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
;
000040      EF.START==      32.      ; START COMMAND WAS ISSUED
000037      EF.RESTART==    31.      ; RESTART COMMAND WAS ISSUED
000036      EF.CONTINUE==   30.      ; CONTINUE COMMAND WAS ISSUED
000035      EF.NEW==        29.      ; A NEW PASS HAS BEEN STARTED
000034      EF.PWR==        28.      ; A POWER-FAIL/POWER-UP OCCURRED
;
; PRIORITY LEVEL DEFINITIONS
;
000340      PRI07== 340
000300      PRI06== 300
000240      PRI05== 240
000200      PRI04== 200
000140      PRI03== 140
000100      PRI02== 100
000040      PRI01== 40
000000      PRI00== 0
;
; OPERATOR FLAG BITS
;
000004      EVL==      4
000010      LOT==     10
000020      ADR==     20
000040      INU==     40
000100      ISR==    100
000200      UAM==    200
000400      BOE==    400
001000      PNT==   1000
002000      PRI==   2000
004000      IXE==   4000
010000      IBE==  10000
020000      IER==  20000
040000      LOE==  40000
100000      HOE== 100000

```

```

1226
1227
1228
1229      000000      CTARGET==0
1230      000001      CASIST==1
1231      000002      CSHCTR==2
1232      000004      CCLNAD==4      ;ARG TYPE FOR 'SHOW COUNTERS' CMD
1233      000010      CCLNAL==8      ;ARG TYPE FOR 'CLEAR NODE/ADR' CMD
1234      000020      CEXIT==16      ;ARG TYPE FOR 'CLEAR NODE/ALL' CMD
1235
1236      ;;;CLOCK ENABLE VALUES TO BE LOADED IN CLK'S CSR;;;
1237
1238      000100      LCLKEN==100      ; L-CLOCK CSR VALUE TO ENABLE THE CLOCK
1239      000111      PCLKEN==111      ; P-CLOCK CSR VALUE TO ENABLE THE CLOCK
1240      001600      PCLKCT==1600      ; P-CLOCK COUNT SET REGISTER FOR COUNTER
1241
1242      ; SPECIAL CLI CODES FOR "CHAR" ARGUMENT IN CLI CALLS
1243      ; (COMMAND LINE INTERPRETER DEFINITIONS)
1244      000000      CLIERR= 0
1245      000001      CLIEXI= 1
1246      000002      CLIBR = 2
1247      000003      CLIBIF= 3
1248      000004      CLISPA= 4
1249      000005      CLINUM= 5
1250      000006      CLIALP= 6
1251      000007      CLIALN= 7
1252      000010      CLIOCT= 8
1253      000011      CLIDEC= 9
1254      000012      CLISTR= 10
1255
1256      ;DEFS FOR COMMAND LINE INTERPRETATION ACTION VALUES
1257
1258      000000      NULL=0
1259      000001      HELP=1
1260      000002      NODE=2
1261      000003      BUILD=3
1262      000004      CRUN=4
1263      000005      CPATRN=5
1264      000006      CSAVE=6
1265      000007      CUMRY=7
1266      000010      CIDENT=10
1267      000011      CEXIT=11
1268      000012      CNOTNUF=12
1269      000013      CEXADR=13
1270      000014      CSAVR4=14
1271      000015      CNODE=15
1272      000016      CALPHA=16
1273      000017      CONES=17
1274      000020      CZEROS=20
1275      000021      CIALT=21
1276      000022      CCALT=22
1277      000023      CCCITT=23
1278      000024      COPRSL=24
1279      000025      CTYPE=25
1280      000026      CSIZE=26
1281      000027      CCPYS=27
1282      000030      CNDADR=30
    
```

```

1283      000031      CNODAL=31
1284      000032      CRNALL=32
1285      000033      CLUPPR=33
1286      000034      CSHMSG=34
1287      000035      CCLMSG=35
1288      000036      CCNTR=36
1289      000037      CNOLOG=37
1290      000040      CFUNCT=40
1291      000041      CUNSAV=41
1292      000042      CCLSUM=42
1293      000043      CDIR=43
1294      000044      CDEFLT=44
1295      000045      CUNSVF=45
1296
1297      000000
1298      000001      ALPHA==0                ;MESSAGE: TYPE VALUES
1299      000002      ONES==1
1300      000003      ZEROS==2
1301      000004      ONEALT==3
1302      000005      ZROALT==4
1303      000006      CCITT==5
1304
1305
1306      ;
1307      ; GLOBAL EQUATES FOR THE DEUNA DRIVER
1308      ;
1309      ;PORT CONTROL AND STATUS REGISTER 0
1310
1311      100000      SERI    ..    BIT15      ; STATUS ERROR INTERRUPT
1312      040000      PCEI    ..    BIT14      ; PORT COMMAND ERROR INTERRUPT
1313      020000      RXI     ..    BIT13      ; RECEIVE RING INTERRUPT
1314      010000      TXI     ..    BIT12      ; TRANSMIT RING INTERRUPT
1315      004000      DNI     ..    BIT11      ; DONE INTERRUPT
1316      002000      RCBI    ..    BIT10      ; RECEIVE BUFFER UNAVAILABLE
1317
1318      000400      FATI    ..    BIT08      ; FATAL ERROR INTERERUPT
1319      000200      INTR    ..    BIT07      ; INTERRUPT SUMMARY <15:0>
1320      000100      INTE    ..    BIT06      ; INTERRUPT ENABLE
1321      000040      RSET    ..    BIT05      ; UNA RESET
1322
1323      ; PORT COMMANDS IN BIT 3 TO BIT 0
1324      ; -----
1325
1326      000001      GETPCB  .. BIT00      ; GET ADDRESS OF PORT CONTROL BLOCK
1327      000002      GETFNT  .. BIT01      ; GET COMMAND IN PORT CONTROL BLOCK
1328      000003      PNOP    .. BIT00:BIT01 ; NO OPERATION PERFORMED
1329      000004      STRT    .. BIT02      ; ENABLE XMIT AND RCVR
1330      000005      BOOT    .. BIT02:BIT00 ; BOOT , -> PRIM LOAD STATE,
1331                                     ; INITATE DOWNLINE LOAD
1332
1333      000010      PDMD    .. BIT03      ; POLLING DEMAND/WAKE UP BIT
1334      000011      TMRO    .. BIT03:BIT00 ; SANITY TIMER ENABLE ( *1 ITS ON)
1335      000012      TMRF    .. BIT03:BIT01 ; SANITY TIMER OFF
1336      000015      RSTI    .. BIT03:BIT02:BIT00 ; RESET SANITY TIMER
1337      000017      STQP    .. BIT03:BIT02:BIT01:BIT00 ; SUSPEND UNA OPERATION
1338
1339

```

```

1340
1341
1342
1343
1344      100000      XPWR == BIT15      ; TRANSCEIVER POWER OK
1345      040000      ICAB == BIT14      ; PORT TO LINK CABLE OK
1346
1347      ; SELF TEST ERROR CODE IN BIT 13 TO BIT 08
1348      000200      PCTO == BIT07      ; PORT COMMAND TIMEOUT
1349
1350      000010      RMTC == BIT03      ; REMOTE CONSOLE RESERVED (=1)
1351
1352      ; PORT STATE IN BIT 2 TO BIT 0
1353
1354      000000      RESET == 0      ; 000 RESET STATE
1355      000001      PRIMLD == BIT00      ; 001 PRIMARY LOAD STATE
1356      000002      READY == BIT01      ; 010 READY STATE
1357      000003      RUN == BIT01!BIT00      ; 011 RUNNING STATE
1358
1359      000005      UNHMT == BIT02!BIT00      ; 101 UNIBUS HALTED STATE
1360      000006      NIHMT == BIT02!BIT01      ; 110 NI HALTED STATE
1361      000007      NIUNI == BIT02!BIT01!BIT00      ; 111 NI AND UNIBUS HALTED STATE
1362
1363
1364
1365      ;PORT CONTROL AND STATUS REGISTER 2
1366
1367      ; LOWER 16 ADDRESS BITS OF THE PORT CONTROL BLOCK BASE
1368      ; ADDRESS POINTER IN BIT 15 TO BIT 0
1369
1370      ;PORT CONTROL AND STATUS REGISTER 3
1371
1372      ; UPPER 2 ADDRESS BITS OF THE PORT CONTROL BLOCK BASE
1373      ; ADDRESS POINTER IN BIT 1 TO BIT 0
1374
1375      ;PORT FUNCTIONS
1376
1377      ; FUNCTION CODES ARE AS FOLLOWS
1378
1379      000000      PFNOP == 0      ; NO OPERATION PERFORMED
1380      000002      RDDEFA == BIT01      ; READ DEFAULT PHYSICAL ADDRESS
1381
1382      000004      RDPHYA == BIT02      ; READ PHYSICAL ADDRESS
1383      000005      WDPHYA == BIT02!BIT00      ; WRITE PHYSICAL ADDRESS
1384
1385      000006      RDMULA == BIT02!BIT01      ; READ LIST OF MULTICAST ADDRESSES
1386      000007      WDMULA == BIT02!BIT01!BIT00      ; WRITE LIST OF MULTICAST ADDRESSES
1387
1388      000010      RDRNGS == BIT03      ; READ BOTH THE RCVR AND XMIT RINGS
1389      000011      WRNGS == BIT03!BIT00      ; WRITE BOTH THE RCVR AND XMIT RINGS
1390
1391      000012      RDCNTS == BIT03!BIT01      ; READ COUNTERS
1392      000013      CLRCNTS == BIT03!BIT01!BIT00      ; READ AND CLEAR COUNTERS
1393
1394      000014      RDMODE == BIT03!BIT02      ; READ INTERNAL LINK MODE REGISTER
1395      000015      WDMODE == BIT03!BIT02!BIT00      ; WRITE INTERNAL LINK MODE REGISTER
1396

```

```

1397      000016      RDSTA == BIT03!BIT02!BIT01 ; READ PORT STATUS
1398      000017      CLRSTA == BIT03!BIT02!BIT01!BIT00 ; READ AND CLEAR PORT STATUS
1399
1400
1401      000020      DMPMEM == BIT04 ; DUMP INTERNAL MEMORY
1402      000021      LDMEM == BIT04!BIT00 ; LOAD INTERNAL MEMORY
1403
1404      000022      RDSYS == BIT04!BIT01 ; READ SYSTEM ID PARAMETERS
1405      000023      WDSYS == BIT04!BIT01!BIT00 ; WRITE SYSTEM ID PARAMETERS
1406
1407      ;
1408      ; ETHERNET PACKET OFFSETS
1409      ;
1410
1411
1412      000016      HEADER == 14. ; OFFSET (SIZE) TO END OF HEADER IN BYTES
1413
1414      000000      DESTIN == 0 ; DESTINATION ADDRESS
1415      000006      SOURCC == 6 ; SOURCE ADDRESS
1416      000014      PROTOT == 12. ; PROTOCOL TYPE FIELD
1417
1418      ;
1419      ; -----
1420      ; ! DESTINATION ADDRESS !
1421      ; ! (6 BYTES) !
1422      ; ! -----
1423      ;
1424      ;
1425      ; +6 ! SOURCE ADDRESS !
1426      ; ! -----
1427      ; ! (6 BYTES) !
1428      ; ! -----
1429      ;
1430      ;
1431      ; +12. ! PROTOCOL TYPE !
1432      ; ! -----
1433      ; +14. ! DATA !
1434      ; ! -----
1435      ; ! MORE DATA !
1436      ;
1437      ;
1438      ; *
1439      ; XMIT RING DESCRIPTOR DEFINITIONS
1440      ; -
1441      ;
1442      ; TDRB*0
1443      ;
1444      ; NOTHING NEEDED
1445      ;
1446      ; TDRB*2
1447      ;
1448      ; NOTHING NEEDED
1449      ;
1450      ;
1451      ; TDRB*4
1452      ;
1453

```

```

1454      000400      ENP      **      BIT08      ; END OF PACKET FLAG
1455      001000      STP      **      BIT09      ; STOP OF PACKET FLAG
1456      002000      DEF      **      BIT10      ; DEFFERRING PACKET FLAG
1457      004000      ONE      **      BIT11      ; XMIT SUCCESSFUL AFTER ONE RETRY
1458      010000      MORE     **      BIT12      ; XMIT SUCCESSFUL AFTER MORE THAN
1459                                     ; ONE RETRY
1460      040000      ERRS     **      BIT14      ; ERROR SUMMARY BIT
1461      100000      OWN      **      BIT15      ; OWNERSHIP BIT (=1 UNA, =0 HOST)
1462
1463      ; TDRB+6
1464
1465      002000      RTRY     **      BIT10      ; RETRY ERROR BIT
1466      004000      LCAR     **      BIT11      ; LOST CARRIER ERROR BIT
1467      010000      LCOL     **      BIT12      ; LATE COLLISION ERROR BIT
1468
1469      040000      UBTO     **      BIT14      ; UNIBUS TIMEOUT ERROR BIT
1470      100000      BUFL     **      BIT15      ; BUFFER LENGTH ERROR BIT
1471
1472      ; *
1473      ; RCVR RING DESCRIPTOR DEFINITIONS
1474      ; -
1475
1476      ; RDRB+0
1477      ;
1478      ; NOTHING NEEDED
1479
1480      ; RDRB+2
1481      ;
1482      ; NOTHING NEEDED
1483
1484
1485      ; RDRB+4
1486      ;
1487
1488      ; <- INDICATES SAME AS DEFINED FOR XMIT RING
1489
1490      ;ENP      **      BIT08      ; END OF PACKET FLAG
1491      ;STP      **      BIT09      ; STOP OF PACKET FLAG
1492
1493      004000      CRC      **      BIT11      ; CRC ERROR IN RECEIVED PACKET
1494      010000      OFLO     **      BIT12      ; MESSAGE OVERFLOW
1495      020000      FRAM     **      BIT13      ; FRAMING ERROR
1496
1497      ;ERRS     **      BIT14      ; ERROR SUMMARY BIT
1498      ;OWN      **      BIT15      ; OWNERSHIP BIT (=1 UNA, =0 HOST)
1499
1500      ; RDRB+6
1501
1502      020000      NCHN     **      BIT13      ; SET TO INDICATE UNA IN NO
1503                                     ; BUFFER CHAIN ON RCVR MODE
1504
1505      ;UBTO     **      BIT14      ; UNIBUS TIMEOUT ERROR BIT
1506      ;BUFL     **      BIT15      ; BUFFER LENGTH ERROR BIT
1507
1508      002756      XPKLEN   ** 1518. ; TRANSMIT PACKET LENGTH
1509      002756      RPKLEN   ** 1518. ; RECIEVE PACKET LENGTH
1510      000006      NO.NTR   ** 6      ; NUMBER OF ENTRIES IN DESCRIPTOR RINGS

```

```

1511      000050      TBLLEN  == 40.
1512
1513      000032      STBLLEN == 90.
1514      000004      FRDADR  == 4
1515
1516
1517
1518      ;
1519      ; SYSTEM ID REPLY MESSAGE OFFSETS
1520      ;
1521      000022      SIRCPT  == 22
1522      000027      SIVERS  == 27
1523      000030      SIECO   == 30
1524      000031      SIUECO  == 31
1525      000035      SIFNCT  == 35
1526      000042      SIADDR  == 42
1527      000053      SIDEV   == 53
1528
1529      ;
1530      ; LOOP DIRECT OFFSETS
1531      ;
1532      000016      LDSKIP  == 16
1533      000020      LDFCT1  == 20
1534      000022      LDAOR1  == 22
1535      000030      LDFCT2  == 30
1536      000032      LDAOR2  == 32
1537
1538      ;
1539      ; FULL ASSIST OFFSETS
1540      ;
1541      000016      FASKIP  == 16
1542      000020      FAFCT1  == 20
1543      000022      FAADR1  == 22
1544      000030      FAFCT2  == 30
1545      000032      FAADR2  == 32
1546      000040      FAFCT3  == 40
1547      000042      FAADR3  == 42
1548      000050      FAFCT4  == 50
1549      000052      FAADR4  == 52
1550
1551      ;
1552      ; COUNTER OFFSETS
1553      ;
1554      000002      C. SECS  == 2
1555      000004      C. PREC  == 4
1556      000010      C. MREC  == 10
1557      000014      C. RERB  == 14
1558      000016      C. RERR  == 16
1559      000020      C. RDAT  == 20
1560      000024      C. RMOB  == 24
1561      000030      C. RLIN  == 30
1562      000032      C. RLEX  == 32
1563      000034      C. FXMT  == 34
1564      000040      C. MXMT  == 40
1565      000044      C. PXM3  == 44
1566      000050      C. PXM2  == 50
1567      000054      C. PXM0  == 54
1568      000060      C. XDAT  == 60
1569      000064      C. XMOB  == 64
1570      000066      C. XABB  == 66

```

```

; NODE TABLE LENGTH (CHANGE STBLLEN IF
; THIS IS CHANGED, OR ELSE!)
; SUMMARY TABLE LENGTH (= 2.25 X TBLLEN)
; OFFSET FOR MESSAGE HEADERS

```

```

1568          000070          C.XABT == 70
1569          000074          C.COLL == 74
1570
1571          .SBTTL GLOBAL DATA SECTION
1572
1573          ;**
1574          ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1575          ; IN MORE THAN ONE TEST.
1576          ;--
1577          ;COMMAND LINE BUFFER, DATA LOCATIONS AND MESSAGES FOR ACTION ROUTINES
1578
1579 002200      CMDBUF: .BLKB 72.          ;BUFFER FOR OPERATOR COMMANDS
1580 002310      000000      KEYWD1: .WORD 0          ;
1581 002312      000000      KEYWD2: .WORD 0
1582 002314      000000      ADRBUF: .WORD 0          ;BUFFER FOR NODE ADDRESS
1583 002316      000000      .WORD 0
1584 002320      000000      .WORD 0
1585 002322      STRBUF: .BLKB 18.        ;BUFFER FOR ALPHANUM. ADDRESS STRING
1586 002344      STRBU1: .BLKB 18.
1587 002366      000000      CBOADR: .WORD 0          ;POINTER FOR BEGINING OF ADDRESS STRING
1588 002370      000000      P#TYPE: .WORD 0          ;LOC. TO HOLD MESSAGE TYPE
1589 002372      000000      P#SIZE: .WORD 0          ;LOC. TO HOLD MESSAGE SIZE
1590 002374      000000      P#CPYS: .WORD 0          ;LOC. TO HOLD NO. OF MESSAGE COPIES
1591 002376      000000      P#PASS: .WORD 0          ;LOC. TO HOLD NO. OF PASSES
1592 002400      000000      NODTY: .WORD 0          ;LOC. TO HOLD NODE TYPE FOR NODE TABLE SETUP
1593 002402      000000      SLOT: .WORD 0          ;USED BY NODE TABLE SUBROUTINES
1594 002404      NODTBL: .BLKW TBLLEN      ; SPACE FOR NODE TABLE
1595 002524      177777      .WORD 177777          ;FILL LAST FOUR BYTES OF TABLE WITH ONES
1596 002526      177777      .WORD 177777
1597 002530      SAVTBL: .BLKW TBLLEN      ;SPACE FOR SAVE TABLE
1598 002650      177777      ILLADR: .WORD 177777     ;ILLEGAL ADDRESS FOR COMPARISON
1599 002652      177777      .WORD 177777         ; (MUST NOT BE PHYSICALLY SEPARATED FROM
1600 002654      177777      .WORD 177777         ; END OF SAVTBL)
1601 002656      STATBL: .BLKW STBLEN      ;SPACE FOR SUMMARY TABLE
1602 003142      177777      .WORD 177777
1603
1604          ;COMMAND LINE TRAVERSE LOCATIONS (USED BY "P#TRV")
1605
1606 003144      000000      P#BUFA: .WORD 0          ;LOC. TO HOLD ADDR. OF CMD LINE BUFFER
1607 003146      000000      P#TREE: .WORD 0          ;LOC. TO HOLD ADDR. OF PARSING TREE
1608 003150      000000      P#ACT: .WORD 0          ;LOC. TO HOLD ADDR. OF ACTION ROUTINE
1609 003152      000000      P#CNI: .WORD 0          ;LOC. TO BE A COUNTER LOCATION
1610 003154      000000      P#NUM: .WORD 0          ;LOC. TO HOLD NUMERIC VALUE FROM PARSE
1611 003156      000000      P#RADX: .WORD 0          ;LOC. TO HOLD RADIX(LO) & +/- (HI BYTE)
1612 003160      000          P#NUF: .BYTE 0          ;RETURN *0 IF ENOUGH OF COMMAND FOUND
1613 003161      000          P#GDBD: .BYTE 0          ;RETURN CODE 0 IF NO ERROR FOUND
1614 003162      000          P#AERR: .BYTE 0          ;RETURN 0 IF 12 DIGIT ADDRESS ENTERED
1615 003163      000          P#MERR: .BYTE 0          ;RETURN -1 IF ERROR IN OPERATOR SELECTED
1616
1617 003164      056060      HLPTAB: .WORD HELP1
1618 003166      056161      .WORD HELP2
1619 003170      056254      .WORD HELP3
1620 003172      056325      .WORD HELP4
1621 003174      056376      .WORD HELP5
1622 003176      056476      .WORD HELP6
1623 003200      056611      .WORD HELP7
1624 003202      056722      .WORD HELP8

```


1625	003204	057012	.WORD	HELP9
1626	003206	057101	.WORD	HELP10
1627	003210	057172	.WORD	HELP11
1628	003212	057270	.WORD	HELP12
1629	003214	057375	.WORD	HELP13
1630	003216	057474	.WORD	HELP14
1631	003220	057573	.WORD	HELP15
1632	003222	057676	.WORD	HELP16
1633	003224	057765	.WORD	HELP17
1634	003226	060070	.WORD	HELP18
1635	003230	060140	.WORD	HELP19
1636	003232	060243	.WORD	HELP20
1637	003234	060321	.WORD	HELP21
1638	003236	060404	.WORD	HELP22
1639	003240	060505	.WORD	HELP23
1640	003242	060605	.WORD	HELP24
1641	003244	060735	.WORD	HELP25
1642	003246	061021	.WORD	HELP26
1643	003250	061125	.WORD	HELP27
1644	003252	061227	.WORD	HELP28
1645	003254	061346	.WORD	HELP29
1646	003256	061416	.WORD	HELP30
1647	003260	000000	.WORD	0
1648	003262	062372	.WORD	MSGTY0
1649	003264	062400	.WORD	MSGTY1
1650	003266	062405	.WORD	MSGTY2
1651	003270	062413	.WORD	MSGTY3
1652	003272	062420	.WORD	MSGTY4
1653	003274	062425	.WORD	MSGTY5
1654	003276	062433	.WORD	MSGTY6

HLPEND:
MSGTAB:

;MESSAGE TYPE ASCII ADDRESS TABLE

; THIS SECTION DEFINES THE DATA PATTERNS USED BY THE EXERCISER

1655				
1656				
1657				
1658	003300		MSGCNT::	
1659	003300	000130	MSG0C: .WORD	EMSG0-MSG00
1660	003302	000001	MSG1C: .WORD	EMSG1-MSG01
1661	003304	000001	MSG2C: .WORD	EMSG2-MSG02
1662	003306	000001	MSG3C: .WORD	EMSG3-MSG03
1663	003310	000001	MSG4C: .WORD	EMSG4-MSG04
1664	003312	000100	MSG5C: .WORD	EMSG5-MSG05
1665	003314	000000	MSG6C: .WORD	0

; THE NUMBER OF BYTES IN EACH MESSAGE

1666				
1667	003316		MSGAD::	
1668	003316	003334	.WORD	MSG00
1669	003320	003464	.WORD	MSG01
1670	003322	003465	.WORD	MSG02
1671	003324	003466	.WORD	MSG03
1672	003326	003467	.WORD	MSG04
1673	003330	003470	.WORD	MSG05
1674	003332	003570	.WORD	OPSLBF

1675				
1676	003334	041	041	042
	003337	043	044	045
	003342	046	047	050
	003345	051	052	053
	003350	054	055	057
	003353	060	061	062

MSG00:: .ASCII \ ! " # \$ % & ' () * + , - / 0 1 2 3 4 5 6 7 8 9 ; : < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z \

003356	063	064	065		
003361	066	067	070		
003364	071	072	073		
003367	074	075	076		
003372	077	100	101		
003375	102	103	104		
003400	105	106	107		
003403	110	111	112		
003406	113	114	115		
003411	116	117	120		
003414	121	122	123		
003417	124	125	126		
003422	127	130	131		
003425	132				
1677 003426	133	135	136	.ASCII	\ [] +-ABCDEFGHIJKLMN OPQRSTUVWXYZ \ ; ALPHANUMERIC
003431	055	101	102		
003434	103	104	105		
003437	106	107	110		
003442	111	112	113		
003445	114	115	116		
003450	117	120	121		
003453	122	123	124		
003456	125	126	127		
003461	130	131	132		
1678 003464				EMSG0::	
1679 003464	377			MSG01::	.BYTE 377 ; MESSAGE OF ALL ONES
1680 003465				EMSG1::	
1681 003465	000			MSG02::	.BYTE 0 ; MESSAGE OF ALL ZEROS
1682 003466				EMSG2::	
1683 003466	252			MSG03::	.BYTE 252 ; MESSAGE OF ALTERNATING ONES
1684 003467				EMSG3::	
1685 003467	125			MSG04::	.BYTE 125 ; MESSAGE OF ALTERNATING ZEROS
1686 003470				EMSG4::	
1687 003470				MSG05::	
1688 003470	177603	157427	031011	.WORD	177603,157427,031011,047321,163715,105221
003476	047321	163715	105221		
1689 003504	143325	142304	040041	.WORD	143325,142304,040041,104116,052606,172334
003512	104116	052606	172334		
1690 003520	105025	123754	111337	.WORD	105025,123754,111337,111523,030030,145064
003526	111523	030030	145064		
1691 003534	137642	143531	063617	.WORD	137642,143531,063617,135075,066730,026575
003542	135075	066730	026575		
1692 003550	052012	053627	070071	.WORD	052012,053627,070071,151172,165044,031605
003556	151172	165044	031605		
1693 003564	166632	016147		.WORD	166632,016147
1694 003570				EMSG5::	
1695 003570				OPSLBF:	.BLKB 66 ; BUFFER FOR OPERATOR SELECTED MESSAGE TYPE
1696					
1697					
1698 003672	000000			CFLAG:	.WORD 0 ; ACTION ROUTINE CMD ARGUMENT FLAG
1699					
1700				.;CLOCK TABLES, EVENT LOG AND POINTERS	
1701 003674	000000			CLKCSR:	.WORD 0 ; CLOCK CSR ADDRESS
1702 003676	000000			CLKBR:	.WORD 0 ; CLOCK INTERRUPT LEVEL
1703 003700	000000			CLKVEC:	.WORD 0 ; CLOCK INTERRUPT VECTOR
1704 003702	000074			CLKHZ:	.WORD 60. ; CLOCK'S FREQUENCY IN HERTZ
1705 003704	000000			CLKEN:	.WORD 0 ; CLOCK'S CSR VALUE TO INTRPT. ENABLE IT

```

1706
1707 003706 000000          TIMMIN: .WORD 0          ; PLACE TO KEEP TIME-SINCE-START
1708 003710 000000          TIMSEC: .WORD 0
1709 003712 000000          TIMTCK: .WORD 0          ; PLACE TO KEEP NO. OF TICKS/SEC.
1710
1711 003714 000000          TIMER1: .WORD 0          ; EVENT TIMER #1 (TICKS)
1712 003716 000000          TIMER2: .WORD 0          ; EVENT TIMER #2 (TICKS)
1713 003720 000000          TIMERS: .WORD 0          ; EVENT TIMER #3 (SECONDS)
1714          .EVEN
1715
1716
1717
1718          ;
1719          ;         TABLE OF START ADDRESS OF RECIEVE RING BUFFERS
1720          ;
1721 003722          RRNGTB::
1722
1723 003722 026006          .WORD RRG001
1724 003724 030764          .WORD RRG002
1725 003726 033742          .WORD RRG003
1726 003730 036720          .WORD RRG004
1727 003732 041676          .WORD RRG005
1728 003734 044654          .WORD RRG006
1729
1730
1731          ;
1732          ;         TABLE OF START ADDRESS OF TRANSMIT RING BUFFERS
1733          ;
1734 003736          XRNGTB::
1735
1736 003736 004162          .WORD XRG001
1737 003740 007140          .WORD XRG002
1738 003742 012116          .WORD XRG003
1739 003744 015074          .WORD XRG004
1740 003746 020052          .WORD XRG005
1741 003750 023030          .WORD XRG006
1742
1743          ;
1744          ;         POINTERS TO DESCRIPTOR RING ENTRIES
1745 003752 003772          XRSRT:: .WORD XRING          ; FIRST ENTRY IN TRANSMIT RING
1746 003754 004066          RRSRT:: .WORD RRING          ; FIRST ENTRY IN RECIEVE RING
1747 003756 003772          XRCUR:: .WORD XRING          ; CURRENT ENTRY IN TRANSMIT RING
1748 003760 004066          RRCUR:: .WORD RRING          ; CURRENT ENTRY IN RECIEVE RING
1749 003762 003772          XRGNT:: .WORD XRING          ; NEXT ENTRY IN TRANSMIT RING
1750 003764 004066          RRGNT:: .WORD RRING          ; NEXT ENTRY IN RECIEVE RING
1751 003766 004054          XRGLST:: .WORD XRING+50.          ; LAST ENTRY IN TRANSMIT RING
1752 003770 004150          RRGLST:: .WORD RRING+50.          ; LAST ENTRY IN RECEIVE RING
1753          ;         VALUE = NO.NTR X 10.
1754
1755
1756
1757
1758
1759
1760 003772 000006          XRING:: .REPT 6
1761          .NLIST
1762          .RNGFRM XRG00,B,0
1763          .LIST
1764          .ENDR
1765          .WORD RPKLEN          ; SEGMENT LENGTH
1766          .WORD XRG001          ; SEGMENT BUFFER ADDRESS
    
```

003776	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004000	000000	.WORD	0	:	STATUS	
004002	000000	.WORD	0	:	SEQUENCE NUMBER	
004004	002756	.WORD	RPKLEN	:	SEGMENT LENGTH	
004006	007140	.WORD	XRG002	:	SEGMENT BUFFER ADDRESS	
004010	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004012	000000	.WORD	0	:	STATUS	
004014	000000	.WORD	0	:	SEQUENCE NUMBER	
004016	002756	.WORD	RPKLEN	:	SEGMENT LENGTH	
004020	012116	.WORD	XRG003	:	SEGMENT BUFFER ADDRESS	
004022	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004024	000000	.WORD	0	:	STATUS	
004026	000000	.WORD	0	:	SEQUENCE NUMBER	
004030	002756	.WORD	RPKLEN	:	SEGMENT LENGTH	
004032	015074	.WORD	XRG004	:	SEGMENT BUFFER ADDRESS	
004034	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004036	000000	.WORD	0	:	STATUS	
004040	000000	.WORD	0	:	SEQUENCE NUMBER	
004042	002756	.WORD	RPKLEN	:	SEGMENT LENGTH	
004044	020052	.WORD	XRG005	:	SEGMENT BUFFER ADDRESS	
004046	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004050	000000	.WORD	0	:	STATUS	
004052	000000	.WORD	0	:	SEQUENCE NUMBER	
004054	002756	.WORD	RPKLEN	:	SEGMENT LENGTH	
004056	023030	.WORD	XRG006	:	SEGMENT BUFFER ADDRESS	
004060	000000	.WORD	0	:	OWNERSHIP AND STATUS BITS	
004062	000000	.WORD	0	:	STATUS	
004064	000000	.WORD	0	:	SEQUENCE NUMBER	
1765						
1769						
1770	004066	000006				
1771			RRING:: .REPT 6			
1772			.NLIST			
1773			RNGFRM RRG00,B,100000			
1774			.LIST			
	004066	002756	.ENDR			
	004070	026006	.WORD	RPKLEN	:	SEGMENT LENGTH
	004072	100000	.WORD	RRG001	:	SEGMENT BUFFER ADDRESS
	004074	000000	.WORD	100000	:	OWNERSHIP AND STATUS BITS
	004076	000000	.WORD	0	:	STATUS
			.WORD	0	:	SEQUENCE NUMBER
	004100	002756	.WORD	RPKLEN	:	SEGMENT LENGTH
	004102	030764	.WORD	RRG002	:	SEGMENT BUFFER ADDRESS
	004104	100000	.WORD	100000	:	OWNERSHIP AND STATUS BITS
	004106	000000	.WORD	0	:	STATUS
	004110	000000	.WORD	0	:	SEQUENCE NUMBER
	004112	002756	.WORD	RPKLEN	:	SEGMENT LENGTH
	004114	033742	.WORD	RRG003	:	SEGMENT BUFFER ADDRESS
	004116	100000	.WORD	100000	:	OWNERSHIP AND STATUS BITS
	004120	000000	.WORD	0	:	STATUS

```

004122 000000 .WORD 0 ; SEQUENCE NUMBER
004124 002756 .WORD RPKLEN ; SEGMENT LENGTH
004126 036720 .WORD RRG004 ; SEGMENT BUFFER ADDRESS
004130 100000 .WORD 100000 ; OWNERSHIP AND STATUS BITS
004132 000000 .WORD 0 ; STATUS
004134 000000 .WORD 0 ; SEQUENCE NUMBER

004136 002756 .WORD RPKLEN ; SEGMENT LENGTH
004140 041676 .WORD RRG005 ; SEGMENT BUFFER ADDRESS
004142 100000 .WORD 100000 ; OWNERSHIP AND STATUS BITS
004144 000000 .WORD 0 ; STATUS
004146 000000 .WORD 0 ; SEQUENCE NUMBER

004150 002756 .WORD RPKLEN ; SEGMENT LENGTH
004152 044654 .WORD RRG006 ; SEGMENT BUFFER ADDRESS
004154 100000 .WORD 100000 ; OWNERSHIP AND STATUS BITS
004156 000000 .WORD 0 ; STATUS
004160 000000 .WORD 0 ; SEQUENCE NUMBER

1775
1779
1780 004162 XRG001::.BLKB XPKLEN ; XMIT RING BUFFERS
1781 007140 XRG002::.BLKB XPKLEN
1782 012116 XRG003::.BLKB XPKLEN
1783 015074 XRG004::.BLKB XPKLEN
1784 020052 XRG005::.BLKB XPKLEN
1785 023030 XRG006::.BLKB XPKLEN
1786
1787 026006 RRG001::.BLKB RPKLEN ; RECIEVE RING BUFFERS
1788 030764 RRG002::.BLKB RPKLEN
1789 033742 RRG003::.BLKB RPKLEN
1790 036720 RRG004::.BLKB RPKLEN
1791 041676 RRG005::.BLKB RPKLEN
1792 044654 RRG006::.BLKB RPKLEN
1793
1794
1795
1796
1797
1798
1799
1800
*****
1801
1802
1803 047632 000000 PCSR0::.WORD ;PCSR0 OF CURRENT SLOT
1804 047634 000000 PCSR1::.WORD ; ADDRESS OF PCSR0 (PORT COMMAND FIELD
1805 047636 000000 PCSR2::.WORD ; 1 (STATE & SELF TEST FIELDS
1806 047640 000000 PCSR3::.WORD ; 2 (PCB ADDRESS LO 15 BITS
1807 ; 3 (PCB ADDRESS HI 2 BITS
1808 047642 000000 PCSR0C::.WORD 0 ;PCSR0 CONTENTS
1809 047644 000000 PCSR1C::.WORD 0 ;PCSR1 CONTENTS
1810 047646 000000 PCSR2C::.WORD 0 ;PCSR2 CONTENTS
1811 047650 000000 PCSR3C::.WORD 0 ;PCSR3 CONTENTS
1812
1813
1814 047652 000000 UNACSR::.WORD 0 ;CSR

```

GLOBAL DATA SECTION

```

1815 047654 00000J UNAVEC:::WORD 0 ;VECTOR
1816 047656 000000 UNAPRI:::WORD 0 ;PRIORITY
1817
1818 047660 000000 FRESIZ:::WORD 0 ;POINTER TO WORD CONTAINING SIZE OF FREE MEMORY
1819 047662 000000 FREMEM:::WORD 0 ;POINTER TO FREE MEMORY SPACE
1820
1821 047664 000000 UNIT:::WORD 0 ;CURRENT UNIT NUMBER BEING TESTED
1822
1823 ; PORT CONTROL BLOCK FUNCTION STRUCTURES
1824
1825 ;PORT CONTROL BLOCK
1826 047666 000000 PCBB0:::WORD 0 ; PORT FUNCTION
1827 047670 000000 PCBB2:::WORD 0 ; PORT FUNCTION DEPENDENT PARAMETERS
1828 047672 000000 PCBB4:::WORD 0 ; PORT FUNCTION DEPENDENT PARAMETERS
1829 047674 000000 PCBB6:::WORD 0 ; PORT FUNCTION DEPENDENT PARAMETERS
1830
1831 ; FUNCTION TABLE
1832
1833 047676 047746 FUNTAB:::WORD $PNOP ; NO OP
1834 047700 000000 .WORD 0 ; FILL IN THE HOLE
1835 047702 047750 .WORD $RDDE ; READ DEFAULT PHYSICAL ADDRESS
1836 047704 000000 .WORD 0 ; FILL IN ANOTHER HOLE
1837 047706 047760 .WORD $RDPH ; READ PHYSICAL ADDRESS
1838 047710 047770 .WORD $WDPH ; WRITE PHYSICAL ADDRESS
1839 047712 050000 .WORD $RDMC ; READ MULTICAST ADDRESS LIST
1840 047714 050040 .WORD $WDMC ; WRITE MULTICAST ADDRESS LIST
1841 047716 050100 .WORD $RDRN ; READ DESCRIPTOR RINGS
1842 047720 050124 .WORD $WDRN ; WRITE DESCRIPTOR RINGS
1843 047722 050150 .WORD $RDCN ; READ COUNTERS
1844 047724 050260 .WORD $CLRC ; READ AND CLEAR COUNTERS
1845 047726 050270 .WORD $RDMO ; READ MODE
1846 047730 050300 .WORD $WDMO ; WRITE MODE
1847 047732 050310 .WORD $RDST ; READ STATUS
1848 047734 050320 .WORD $CLRS ; READ AND CLEAR STATUS
1849 047736 050330 .WORD $DMEM ; DUMP INTERNAL MEMORY
1850 047740 050350 .WORD $LMEM ; LOAD INTERNAL MEMORY
1851 047742 050360 .WORD $RDSY ; READ SYS ID PARAMETERS
1852 047744 050370 .WORD $WTSY ; WRITE SYS ID PARAMETERS
1853
1854 ;*
1855 ; PNOP == 0 ; PORT NO-OPERATION
1856 ;-
1857 .EVEN
1858 047746 000000 $PNOP:::WORD 0 ; NO-OP
1859
1860 ;*
1861 ; RDDEFA == BIT01 ; READ DEFAULT PHYSICAL ADDRESS
1862 ;-
1863 .EVEN
1864
1865 047750 000002 $RDDE:::WORD 2 ; PCBB+0 FUNCTION READ DFFAULT
1866 047752 000000 DEPADR:::WORD 0 ; PCBB+2 PHYSICAL ADDRESS
1867 047754 000000 .WORD 0 ; PCBB+4
1868 047756 000000 .WORD 0 ; PCBB+6
1869
1870 ;*
1871 ; ROPHYA == BIT02 ; READ PHYSICAL ADDRESS

```

```

1872
1873      :-
1874      .EVEN
1875 047760 000004      $RDPH::      .WORD 4      ; PCBB+0 READ CURRENT (ACTIVE)
1876 047762 000000      PHYADR::      .WORD 0      ; PCBB+2      .PHYSICAL ADDRESS
1877 047764 000000      .WORD 0      ; PCBB+4
1878 047766 000000      .WORD 0      ; PCBB+6
1879
1880      ;+
1881      ;      WDPHYA == BIT02!BIT00      ; WRITE PHYSICAL ADDRESS
1882      ;-
1883      .EVEN
1884 047770 000005      $WDPH::      .WORD 5      ; PCBB+0 WRITE PHYSICAL ADDRESS
1885 047772 000000      .WORD 0      ; PCBB+2
1886 047774 000000      .WORD 0      ; PCBB+4
1887 047776 000000      .WORD 0      ; PCBB+6
1888
1889      ;+
1890      ;      RDMULA == BIT02!BIT01      ; READ MULTICAST ADDRESS LIST
1891      ;-
1892
1893      .EVEN
1894 050000 000006      $RDMC::      .WORD 6      ; FUNCTION CODE
1895 050002 050010      .WORD UCB6    ; UCBB ADDRESS
1896 050004 000000      .WORD 0      ; PCBB+4
1897 050006 000000      .WORD 0      ; PCBB+6
1898
1899 050010      UCB6::      .BLKW 12.    ; ENOUGH ROOM FOR 4 ADDRESSES
1900
1901      ;+
1902      ;      WDMULA == BIT02!BIT01!BIT00      ; WRITE MULTICAST ADDRESS LIST
1903      ;-
1904
1905      .EVEN
1906 050040 000007      $WDMC::      .WORD 7      ; FUNCTION CODE
1907 050042 050050      .WORD UCB7    ; UCBB ADDRESS
1908 050044 000400      .WORD 400    ; LENGTH OF LIST = 1
1909 050046 000000      .WORD 0      ; PCBB+6
1910
1911 050050      UCB7::      .WORD 253    ; MULTICAST ADDRESS FOR LOOPBACK
1912 050052 001000      .WORD 1000
1913 050054 000000      .WORD 0
1914 050056      .BLKW 9.    ; ROOM FOR THREE MORE ADDRESSES
1915
1916      ;+
1917      ;      RDRNGS == BIT03      ; READ BOTH THE RCVR AND XMIT RINGS
1918      ;-
1919
1920      .EVEN
1921 050100 000010      $RDRN::      .WORD 10     ; FUNCTION CODE
1922 050102 050110      .WORD UCB10   ; UCBB ADDRESS
1923 050104 000000      .WORD 0      ; NULL
1924 050106 000000      .WORD 0      ; NULL
1925
1926      .EVEN
1927
1928 050110 003772      UCB10::      .WORD XRING  ; UCBB

```

```

GLOBAL DATA SECTION

1929 050112 002000          .WORD 2000          ; UCBB+2
1930 050114 000000          .WORD 0            ; UCBB+4
1931 050116 004066          .WORD RRING        ; UCBB+6
1932 050120 002000          .WORD 2000        ; UCBB+10
1933 050122 000000          .WORD 0           ; UCBB+12
1934
1935
1936
1937          ;
1938          ;          WDRNGS == BIT03!BIT00          ; WRITE BOTH THE RCVR AND XMIT RINGS
1939          ;
1940          .EVEN
1941
1942 050124 000011          $WDRN::          .WORD 11          ; FUNCTION CODE
1943 050126 050134          .WORD UCB11       ; UCBB ADDRESS
1944 050130 000000          .WORD 0           ; NULL
1945 050132 000000          .WORD 0           ; NULL
1946
1947          .EVEN
1948
1949 050134          UCB11::
1950 050134 003772          .WORD XRING       ; TRANSMIT RING BASE ADDRESS
1951 050136 000          .BYTE 0           ; HI BITS OF TRANSMIT RING BASE ADDRESS
1952 050137 005          .BYTE 5           ; FIVE WORDS PER RING ENTRY (1 FOR PORT DRIVER)
)
1953 050140 000005          .WORD NO.NTR     ; EIGHT TRANSMIT DESCRIPTORS IN THE RING
1954
1955 050142 004066          .WORD RRING       ; RECEIVE RING BASE ADDRESS
1956 050144 000          .BYTE 0           ; HI BITS OF RECEIVE RING BASE ADDRESS
1957 050145 005          .BYTE 5           ; FIVE WORDS PER RING ENTRY (1 FOR PORT DRIVER)
)
1958 050146 000006          .WORD NO.NTR     ; EIGHT RECEIVE DESCRIPTORS IN THE RING
1959
1960
1961
1962          ;
1963          ;          RDCNTS == BIT03!BIT01          ; READ COUNTERS
1964          ;
1965          .EVEN
1966
1967 050150 000012          $RDCN::          .WORD 12          ; FUNCTION
1968 050152 050160          .WORD UCB12       ; UCBB ADDRESS
1969
1970          ;          DEFAULT COUNT OF COUNTER LIST
1971          ;          ;          40 (OCTAL)
1972 050154 000000          .WORD 0           ; NULL
1973
1974 050156 000100          .WORD 100        ; (# OF WORDS IN LIST = UPPER BYTE)
1975          ;          ;          MAX NUMBER VALUE = 32 (DECIMAL) *
1976          .EVEN
1977
1978
1979 050160          UCB13::
1980 050160 000000          UCB12::          .WORD 0           ; UCBB
1981 050162 000000          .WORD 0           ; UCBB+2
1982 050164 000000          .WORD 0           ; UCBB+4
1983 050166 000000          .WORD 0           ; UCBB+6
1984 050170 000000          .WORD 0           ; UCBB+10
1985 050172 000000          .WORD 0           ; UCBB+12

```


GLOBAL DATA SECTION

```

1986 050174 000000
1987 050176 000000
1988 050200 000000
1989 050202 000000
1990 050204 000000
1991 050206 000000
1992 050210 000000
1993 050212 000000
1994 050214 000000
1995 050216 000000
1996 050220 000000
1997 050222 000000
1998 050224 000000
1999 050226 000000
2000 050230 000000
2001 050232 000000
2002 050234 000000
2003 050236 000000
2004 050240 000000
2005 050242 000000
2006 050244 000000
2007 050246 000000
2008 050250 000000
2009 050252 000000
2010 050254 000000
2011 050256 000000
2012
2013
2014
2015
2016
2017
2018
2019 050260 000013
2020 050262 050160
2021
2022 050264 000000
2023 050266 000010
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036 050270 000014
2037 050272 000000
2038
2039
2040 050274 000000
2041 050276 000000
2042

```

```

        .WORD 0      ; UCBB+14
        .WORD 0      ; UCBB+16
        .WORD 0      ; UCBB+20
        .WORD 0      ; UCBB+22
        .WORD 0      ; UCBB+24
        .WORD 0      ; UCBB+26
        .WORD 0      ; UCBB+30
        .WORD 0      ; UCBB+32
        .WORD 0      ; UCBB+34
        .WORD 0      ; UCBB+36
        .WORD 0      ; UCBB+40
        .WORD 0      ; UCBB+42
        .WORD 0      ; UCBB+44
        .WORD 0      ; UCBB+46
        .WORD 0      ; UCBB+50
        .WORD 0      ; UCBB+52
        .WORD 0      ; UCBB+54
        .WORD 0      ; UCBB+56
        .WORD 0      ; UCBB+60
        .WORD 0      ; UCBB+62
        .WORD 0      ; UCBB+64
        .WORD 0      ; UCBB+66
        .WORD 0      ; UCBB+70
        .WORD 0      ; UCBB+72
        .WORD 0      ; UCBB+74
        .WORD 0      ; UCBB+76

;+
; CLRCNTS == BIT03!BIT01!BIT00 ; READ AND CLEAR COUNTERS
;-

.EVEN
$CLRC::      .WORD 13      ; FUNCTION
              .WORD UCBB13 ; UCBB ADDRESS
              .WORD 0      ; DEFAULT COUNT OF COUNTER LIST
              .WORD 40     ; NULL
              .WORD 40     ; (# OF WORDS IN LIST * UPPER BYTE)
              .WORD 40     ; MAX NUMBER VALUE * 32 (DECIMAL) *
              .WORD 40     ; 40 (OCTAL)

;( FOR UCBB13:: SEE UCB 12 ABOVE)

;+
; RDMODE == BIT03!BIT02      ; READ INTERNAL LINK MODE REGISTER
;-

.EVEN
$RDMO::      .WORD 14     ; FUNCTION CODE
              .WORD 0      ; A 16 BIT COPY OF THE
              .WORD 0      ; BITS TO READ THE UNA INTERNAL
              .WORD 0      ; MODE REGISTER
              .WORD 0      ; NULL
              .WORD 0      ; NULL

```

```

GLOBAL DATA SECTION

2043
2044      ;+
2045      ;          WDMODE == BIT03!BIT02!BIT00 ; WRITE INTERNAL LINK MODE REGISTER
2046      ;-
2047      .EVEN
2048 050300 000015      $WDMO::          .WORD 15      ; FUNCTION CODE
2049 050302 000000      .WORD 0          ; A 16 BIT COPY OF THE
2050      .WORD 0          ; BITS TO WRITE THE UNA INTERNAL
2051      .WORD 0          ; MODE REGISTER
2052 050304 000000      .WORD 0          ; NULL
2053 050306 000000      .WORD 0          ; NULL
2054
2055
2056      ;+
2057      ;          RDSTA == BIT03!BIT02!BIT01 ; READ PORT STATUS
2058      ;-
2059      .EVEN
2060
2061 050310 000016      $RDST::          .WORD 16      ; FUNCTION CODE
2062 050312 000000      STATUS::          .WORD 0        ; A LIST OF ERRORS AND STATUS
2063 050314 000000      .WORD 0          ; LOWER BYTE = # OF MULTICAST ADRS
2064      .WORD 0          ; MAXIMUM SUPPORTED BY UNA
2065      .WORD 0          ; UPPER BYTE = # OF MULTICAST ADRS
2066      .WORD 0          ; CURRENTLY SUPPORTED BY UNA
2067 050316 000000      .WORD 0          ; WORD = MAXIMUM # OF WORDS IN
2068      .WORD 0          ; UCB FOR COUNTERS
2069      .WORD 0          ; AS CURRENTLY PERCEIVED
2070      .WORD 0          ; BY THE UNA
2071
2072
2073      ;+
2074      ;          CLRSTA == BIT03!BIT02!BIT01!BIT0
2075      ;-
2076      .EVEN
2077 050320 000017      $CLRS::          .WORD 17      ; FUNCTION CODE
2078 050322 000000      .WORD 0        ; A LIST OF ERRORS AND STATUS
2079 050324 000000      .WORD 0          ; LOWER BYTE = # OF MULTICAST ADRS
2080      .WORD 0          ; MAXIMUM SUPPORTED BY UNA
2081      .WORD 0          ; UPPER BYTE = # OF MULTICAST ADRS
2082      .WORD 0          ; CURRENTLY SUPPORTED BY UNA
2083 050326 000000      .WORD 0          ; WORD = MAXIMUM # OF WORDS IN
2084      .WORD 0          ; UCB FOR COUNTERS
2085      .WORD 0          ; AS CURRENTLY PERCEIVED
2086      .WORD 0          ; BY THE UNA
2087
2088
2089      ;+
2090      ;          DMPMEM == BIT04
2091      ;-
2092      .EVEN
2093 050330 000020      $DMEM::          .WORD 20      ; FUNCTION CODE
2094 050332 050340      .WORD UCB20   ; UCBB ADDRESS
2095 050334 000000      .WORD 0
2096 050336 000000      .WORD 0
2097
2098 050340      UCB20::          .WORD 0
2099 050340 000000      UCB21::          .WORD 0          ; FUNCTION LENGTH (NO OF WORDS TO XFER)

```

```

2100 050342 000000          .WORD 0          ; HDBB - HOST MEMORY DATA BLOCK ADDRESS
2101 050344 000000          .WORD 0          ; MUST BE ZERO
2102 050346 000000          .WORD 0          ; IDBB - INTERNAL DATA BLOCK BASE ADDRESS
2103
2104
2105          ;          LDMEM == BIT04!BIT00          ; LOAD UNA INTERNAL MEMORY
2106          ;-
2107
2108          .EVEN
2109 050350 000021          $LMEM::          .WORD 21          ; FUNCTION CODE
2110 050352 05034C          .WORD UCB21      ; UCBB ADDRESS
2111 050354 000000          .WORD 0
2112 050356 000000          .WORD 0
2113
2114
2115          ;          RDSYS == BIT04!BIT01          ; READ SYSTEM ID
2116          ;-
2117
2118          .EVEN
2119 050360 000022          $RDSY::          .WORD 22          ; FUNCTION CODE
2120 050362 050400          .WORD UCB22      ; UCBB ADDRESS
2121 050364 000000          .WORD 0
2122 050366 000033          .WORD 27         ; LENGTH OF ID MESSAGE
2123
2124          ;          WTSYS == BIT04!BIT01!BIT00      ; WRITE SYSTEM ID
2125          ;-
2126
2127          .EVEN
2128 050370 000023          $WTSY::          .WORD 23          ; FUNCTION CODE
2129 050372 050400          .WORD UCB23      ; UCBB ADDRESS
2130 050374 000000          .WORD 0
2131 050376 000033          .WORD 27         ; LENGTH OF ID MESSAGE
2132
2133          UCB22:
2134 050400 000000          .WORD 0          ; UDBB+0
2135 050402 000000          .WORD 0          ; UDBB+2
2136 050404 000000          .WORD 0          ; UDBB+4
2137 050406 000000          .WORD 0          ; UDBB+6
2138 050410 000000          .WORD 0          ; UDBB+10
2139 050412 000000          .WORD 0          ; UDBB+12
2140 050414 000000          .WORD 0          ; UDBB+14
2141 050416 000000          .WORD 0          ; UDBB+16
2142 050420 000000          .WORD 0          ; UDBB+20
2143 050422 000000          .WORD 0          ; UDBB+22
2144 050424 000000          .WORD 0          ; UDBB+24
2145 050426 000000          .WORD 0          ; UDBB+26
2146 050430 000000          .WORD 0          ; UDBB+30
2147 050432 000000          .WORD 0          ; UDBB+32
2148 050434 000000          .WORD 0          ; UDBB+34
2149 050436 000000          .WORD 0          ; UDBB+36
2150 050440 000000          .WORD 0          ; UDBB+40
2151 050442 000000          .WORD 0          ; UDBB+42
2152 050444 000000          .WORD 0          ; UDBB+44
2153 050446 000000          .WORD 0          ; UDBB+46
2154 050450 000000          .WORD 0          ; UDBB+50
2155 050452 000000          .WORD 0          ; UDBB+52
2156 050454 000000          .WORD 0          ; UDBB+54

```

```

2157 050456 000000 .WORD 0 ;UDBB+56
2158 050460 000000 .WORD 0 ;UDBB+60
2159 050462 000000 .WORD 0 ;UDBB+62
2160 050464 000000 .WORD 0 ;UDBB+64
2161
2162 050466 000000 UDBB:: .WORD 0 ;UNIBUS DATA BLOCK BASE
2163 050470 000000 .WORD 0 ;+2
2164 050472 000000 .WORD 0 ;+4
2165 050474 000000 .WORD 0 ;+6
2166
2167
2168 ;
2169 ; SUMMARY DATA COUNTERS
2170 ;
2171 050476 000000 S.REC:: .WORD 0 ; MESSAGES RECEIVED
2172 050500 000000 S.NREC:: .WORD 0 ; MESSAGES NOT RECEIVED
2173 050502 000000 S.LEN:: .WORD 0 ; LENGTH ERRORS
2174 050504 000000 S.COMP:: .WORD 0 ; COMPARE ERRORS
2175 050506 000000 S.BYTE:: .WORD 0 ; BYTES COMPARED
2176 050510 000000 S.XFER:: .WORD 0 ; BYTES TRANSFERED
2177
2178 ;
2179 ; DEUNA DRIVER AND ASSOCIATED SUBROUTINES DATA
2180 ;
2181
2182 050512 000000 FATFLG:: .WORD 0 ; FATAL ERROR FLAG
2183 050514 000000 PCEFLG:: .WORD 0 ; PORT COMMAND ERROR FLAG
2184 050516 000000 NIRCNT:: .WORD 0 ; UNA RECIEVE MESSAGE COUNTER
2185 050520 000000 XFLAG:: .WORD 0 ; FRAME TRANSMITTED FLAG
2186 050522 000000 DNIFLG:: .WORD 0 ; DONE INTERRUPT FLAG
2187 050524 000000 RBFCNT:: .WORD 0 ; RECIEVE BUFFERS LOST COUNTER
2188 050526 000000 BDCOUNT:: .WORD 0 ; UNEXPLAINED INTERRUPTS COUNTER
2189 050530 000000 ERRFLG:: .WORD 0 ; ERROR FLAG
2190 050532 000000 TIMEOUT:: .WORD 0 ; TIME OUT COUNTER
2191 050534 000000 RETRYS:: .WORD 0 ; COUNTER FOR FRAMES FAILING DUE TO RTRY ERROR
2192 050536 000000 RCVERR:: .WORD 0 ; COUNTS NO. OF BUFFERS RECEIVED WITH ERRORS
2193 050540 000000 RCVBUF:: .WORD 0 ; COUNTS NO. OF GOOD BUFFERS RECEIVED
2194 050542 000000 COUNT:: .WORD 0 ; USED IN BLDDBUF SUBROUTINE AS COUNTER
2195 050544 000220 PROTO0:: .WORD 000220 ; PROTOCALL TYPE FOR LOOPBACK MESSAGES
2196 050546 001140 PROTO2:: .WORD 001140 ; PROTOCALL TYPE FOR REMOTE CONSOLE
2197 050550 000000 TEMP:: .WORD 0 ; USED IN XMIT AS TEMPORARY STORAGE
2198 050552 000000 TEMP1:: .WORD 0 ; USED FOR TEMPORARY STORAGE
2199 050554 000000 TEMP2:: .WORD 0 ; USED FOR TEMPORARY STORAGE
2200 050556 000000 TEMP3:: .WORD 0 ; USED FOR TEMPORARY STORAGE
2201 050560 000000 XFER:: .WORD 0 ; STORES 'BYTES TRANSFERED'
2202 050562 000000 CPYCNT:: .WORD 0 ; 'NO. OF COPIES' COUNTER FOR LOOPING
2203 050564 000000 PCCALL:: .WORD 0 ; STORES PC OF CALLING ROUTINE FOR ERROR REPORTS
2204 050566 000000 BUFLN:: .WORD 0 ; STORES TRANSMIT BUFFER LENGTH
2205 050570 000000 CMPBUF:: .WORD 0 ; STORES LOCATION OF DATA BUFFER TO BE COMPARED
2206 050572 .PATCH:: .BLKW 40 ; 40 WORDS FOR PROGRAM PATCH
2207
2208 ;
2209 ; REQUEST ID MESSAGE FORMAT
2210 ;
2211
2212 050712 REQID:: .WORD 3 ; BYTE COUNT (=3 FOR REQUEST ID)
2213 050712 000000

```

```

2214 050714 000005 .WORD 5 ; FUNCTION CODE FOR REQUEST ID
2215 050716 051115 .WORD "MR ; RECEIPT NUMBER
2216
2217
2218 ; LOOP DIRECT MESSAGE
2219 ;
2220
2221 .EVEN
2222
2223 050720 LOPDIR::
2224 050720 000000 .WORD 0 ; SKIP COUNT
2225 050722 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2226 050724 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2227 050732 000001 .WORD 1 ; FUNCTION = REPLY
2228 050734 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2229
2230
2231 ; TRANSMIT ASSIST MESSAGE
2232 ;
2233
2234 050742 TASIST::
2235 050742 000000 .WORD 0 ; SKIP COUNT
2236 050744 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2237 050746 000000 000000 000000 .WORD 0,0,0 ; TRANSMIT ASSIST ADDRESS
2238 050754 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2239 050756 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2240 050764 000001 .WORD 1 ; FUNCTION = REPLY
2241 050766 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2242
2243
2244 ; RECIEVE ASSIST MESSAGE
2245 ;
2246
2247 050774 RASIST::
2248 050774 000000 .WORD 0 ; SKIP COUNT
2249 050776 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2250 051000 000000 000000 000000 .WORD 0,0,0 ; TRANSMIT ASSIST ADDRESS
2251 051006 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2252 051010 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2253 051016 000001 .WORD 1 ; FUNCTION = REPLY
2254 051020 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2255
2256
2257 ; FULL ASSIST MESSAGE
2258 ;
2259
2260 051026 FASIST::
2261 051026 000000 .WORD 0 ; SKIP COUNT
2262 051030 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2263 051032 000000 000000 000000 .WORD 0,0,0 ; TARGET NODE ADDRESS
2264 051040 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2265 051042 000000 000000 000000 .WORD 0,0,0 ; ASSIST NODE ADDRESS
2266 051050 000002 .WORD 2 ; FUNCTION = FORWARD DATA
2267 051052 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2268 051060 000001 .WORD 1 ; FUNCTION = REPLY
2269 051062 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2270

```

2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286
 2287 051070
 2288
 2289
 2290 051070
 2291 051074
 2292 051100
 2293 051102
 2294 051116
 2295 051120
 2296 051134
 2297 051140
 2298 051154
 2299 051156
 2300 051170
 2301 051174
 2302 051200
 2303 051212
 2304 051216
 2305 051234
 2306 051236
 2307 051250
 2308 051252
 2309 051254
 2310 051256
 2311 051272
 2312 051276
 2313 051316
 2314 051322
 2315 051340
 2316 051344
 2317 051362
 2318 051366
 2319 051402
 2320 051404
 2321 051424
 2322 051430
 2323 051432
 2324
 2325
 2326
 2327 051434

.SBTTL COMMAND LINE ACTION TREE

;SAMPLE CLI TREE NODE (ALWAYS AT LEAST 1 WORD)

```

|-----|
| ! ACTION ! CHAR CODE !
|-----|
| ! MISS DISPLACEMENT !           ONLY IF "MISS" ARGUMENT DEFINED
|-----|
| ! NEXT MODE DISPLMNT !         ONLY IF "ASCII" ARGUMENT DEFINED
|-----|
| ! ASCIZ MATCH STRING !         ONLY IF "ASCII" ARGUMENT DEFINED
| !      (.EVEN)      !
|-----|
    
```

CLITRE:

;FIRST KEYWORD

```

N10$: CLI CLISPA,0,N10$ ;SKIP ANY LEADING SPACES
      CLI <'?'>,HELP,N12$ ;IS THE FIRST NON-SP CHAR. A "?"
      CLI CLIEXI,0 ; IF YES DO "HLP" AND EXIT
N12$: CLI CLISTR,HELP,N14$,<'HELP'> ;ELSE IS FIRST WORD A "HELP"
      CLI CLIEXI,0 ; IF YES DO "HLP" AND EXIT
N14$: CLI CLISTR,NOTNUF,N16$,<'NODE'> ;ELSE IS FIRST WORD A "NODE"
      CLI CLIBR,0,N80$ ; IF YES, BR N80$
N16$: CLI CLISTR,BUILD,N18$,<'BUILD'> ;ELSE IS FIRST WORD A "BUILD"
      CLI CLIEXI,0 ; IF YES DO "BUILD" AND EXIT
N18$: CLI CLISTR,NOTNUF,N20$,<'RUN'> ;ELSE IS FIRST WORD A "RUN"
      CLI CLIBR,0,N180$ ; IF YES, BR N180$
N20$: CLI <'S'>,NOTNUF,N25$ ;ELSE IS FIRST CHAR. A "S"
      CLI CLISTR,0,N22$,<'HOW'> ; IF YES IS REST OF WORD "HOW"
      CLI CLIBR,0,N100$ ; IF YES, BR N100$
N22$: CLI CLISTR,SUMMARY,N23$,<'UMMARY'> ; ELSE IS REST OF WORD "UMMARY"
      CLI CLIEXI,0 ; IF YES, DO "SUMM" AND EXIT
N23$: CLI CLISTR,CSAVE,N24$,<'AVE'> ; ELSE IS REST OF WORD "AVE"
      CLI CLIEXI,0 ; IF YES, DO "SAVE" AND EXIT
N24$: CLI CLIERR,0 ; ELSE "ILL COMMAND"
      CLI CLIEXI,0 ; EXIT
N25$: CLI CLISTR,NOTNUF,N26$,<'CLEAR'> ;ELSE IS FIRST WORD A "CLEAR"
      CLI CLIBR,0,N120$ ; IF YES, BR N120$
N26$: CLI CLISTR,NOTNUF,N28$,<'IDENTIFY'> ;ELSE IS FIRST WORD "IDENTIFY"
      CLI CLIBR,C,N140$ ; IF YES, GET ADDR, BR N140$
N28$: CLI CLISTR,NOTNUF,N29$,<'MESSAGE'> ;ELSE IS FIRST WORD "MESSAGE"
      CLI CLIBR,0,N160$ ; IF YES, BR N160$
N29$: CLI CLISTR,CUNSAV,N30$,<'UNSAVE'> ;ELSE IS FIRST WORD "UNSAVE"
      CLI CLIBR,0,N210$ ; IF YES, BR TO N210$
N30$: CLI CLISTR,EXIT,N31$,<'EXIT'> ;ELSE IS FIRST WORD "EXIT"
      CLI CLIEXI,0 ; IF YES EXIT
N31$: CLI CLISTR,NOTNUF,N32$,<'FUNCTION'> ;ELSE IS FIRST WORD "FUNCTION"
      CLI CLIBR,0,N200$ ; IF YES, BR N200$
N32$: CLI CLIERR,0 ;OTHERWISE "ILL CMD",
      CLI CLIEXI,0 ; EXIT
    
```

;SECOND KEYWORD (ADR/TYPE) FOR NODE COMMAND

```

N80$: CLI CLISPA,0,N81$ ;SKIP ANY LEADING SPACES
    
```

```

2328 051440      N81$:  CLI      CLIBR,CSAVR4,N82$
2329 051444      N82$:  CLI      CLIBR,NOUE,N90$
2330
2331 051450      N90$:  CLI      CLIBIF,0,N32$
2332 051454      N95$:  CLI      CLIEXI,0
2333
2334              ;SECOND KEYWORD FOR SHOW COMMAND
2335
2336 051456      N100$: CLI      CLISPA,0,N101$
2337 051462      N101$: CLI      CLISTR,CNODE,N102$,<'NODES'>
2338 051476      N102$: CLI      CLIBR,0,N110$
2339 051502      N102$: CLI      CLISTR,CSHMSG,N104$,<'MESSAGE'>
2340 051520      N104$: CLI      CLIBR,0,N110$
2341 051524      N104$: CLI      CLISTR,CCNTR,N106$,<'COUNTERS'>
2342 051544      N106$: CLI      CLIBR,0,N110$
2343 051550      N106$: CLI      CLIBR,0,N32$
2344 051554      N110$: CLI      CLIEXI,0
2345
2346              ;SECOND KEYWORD FOR CLEAR COMMAND
2347
2348 051556      N120$: CLI      CLISPA,0,N121$
2349 051562      N121$: CLI      CLISTR,0,N130$,<'NODE'>
2350 051576      N122$: CLI      CLISPA,0,N122$
2351 051602      N122$: CLI      <'/'>,CSAVR4,N32$
2352 051606      N123$: CLI      <'A'>,0,N123$
2353 051612      N123$: CLI      CLISTR,CNODAL,N124$,<'LL'>
2354 051624      N123$: CLI      CLIBR,0,N135$
2355 051630      N123$: CLI      <'N'>,0,N124$
2356 051634      N124$: CLI      CLIDF,0,N32$
2357 051640      N124$: CLI      CLIPR,0,N135$
2358 051644      N126$: CLI      CLIDF,0,N126$
2359 051650      N126$: CLI      CLIDF,0,N135$
2360 051654      N130$: CLI      CLISTR,CCLMSG,N132$,<'MESSAGE'>
2361 051672      N132$: CLI      CLIBR,0,N135$
2362 051676      N132$: CLI      CLISTR,CCLSUM,N134$,<'SUMMARY'>
2363 051714      N134$: CLI      CLIBR,0,N135$
2364 051720      N134$: CLI      LLERR,0
2365 051722      N135$: CLI      CLTEXTI,0
2366
2367              ;ADDRESS FOR IDENTIFY COMMAND
2368
2369 051724      N140$: CLI      CLISPA,0,N141$
2370 051730      N141$: CLI      CLIPR,CSAVR4,N142$
2371 051734      N142$: CLI      CLIALN,0,N32$
2372 051740      N143$: CLI      CLIPR,CEXADR,N143$
2373 051744      N143$: CLI      CLIEXI,IDENT
2374
2375              ;REMAINING COMMAND LINE FOR MESSAGE COMMAND
2376
2377 051746      N160$: CLI      CLISPA,0,N161$
2378 051752      N161$: CLI      <'/'>,0,N178$
2379 051756      N161$: CLI      CLISTR,0,N170$,<'TYPE'>
2380 051772      N162$: CLI      <'='>,0,N32$
2381 051776      N162$: CLI      CLISTR,CALPHA,N162$,<'ALPHA'>
2382 052012      N162$: CLI      CLIBR,0,N168$
2383 052016      N162$: CLI      CLISTR,CONES,N163$,<'ONES'>
2384 052032      N162$: CLI      CLIBR,0,N168$

```

```

;SAVE STRING POINTER LOCATION
;PARSE THROUGH ADDRESS,CHECK
;FOR TARGET OR ASSIST, DO NODE
;TAKE ERROR BRANCH IF ERROR EXISTS
;EXIT

```

```

;SKIP LEADING SPACES
;IS NEXT WORD "NODES"
; IF YES, SET FLAG, BR N110$
; ELSE IS NEXT WORD "MESSAGE"
; IF YES, SET FLAG, BR N110$
; ELSE IS NEXT WORD "COUNTERS"
; GO TO COUNTERS ROUTINE
;ELSE "ILL COMMAND"
;EXIT

```

```

;SKIP LEADING SPACES
;IS NEXT WORD "NODE"
; IF YES SKIP SPACES
; LOOK FOR DELIMITER, ELSE "ILL COM"
; IS NEXT CHAR. AN "A"
; IF YES, IS WORD "ALL"
; IF YES, SET FLAG, BR N135$
; ELSE IS NEXT CHAR. AN "N"
; IF YES, STORE NODE LOGICAL NAME
; BR TO CLR. NODE LOGICAL ROUTINE
; ELSE, EXTRACT ADDRESS
; SET FLAG, BR N135$
; ELSE IS NEXT WORD "MESSAGE"
; IF YES, SET FLAG, BR N135$
; ELSE IS NEXT WORD "SUMMARY"
; IF YES, CLEAR TABLE AND EXIT
;ELSE, "ILL COMMAND",
;EXIT

```

```

;SKIP LEADING SPACES
;SAVE POINTER TO FIRST CHAR. OF ADDRESS
;CHECK THAT ADDRESS HAS LEGAL CHAR.S
;GET ADDRESS
;DO "IDENTIFY", EXIT

```

```

;SKIP LEADING SPACES
;IF CHAR. "/", CONT., ELSE BR N178$
;IS NEXT WORD "TYPE"
; IF YES, FOLLOWED BY " "?
; IF "ALPHA", SET FLAG
; CONTINUE AT N168$
; IF "ONES", SET FLAG
; CONTINUE AT N168$

```

```

2385 052036 N163$: CLI CLISTR,CZEROS,N164$,<'ZEROS'> ; IF "ZEROS", SET FLAG
2386 052052 CLI CLIBR,0,N168$ ; CONTINUE AT N168$
2387 052056 N164$: CLI CLISTR,C1ALT,N165$,<'1ALT'> ; IF "1ALT", SET FLAG
2388 052072 CLI CLIBR,0,N168$ ; CONTINUE AT N168$
2389 052076 N165$: CLI CLISTR,COALT,N166$,<'OALT'> ; IF "OALT", SET FLAG
2390 052112 CLI CLIBR,0,N168$ ; CONTINUE AT N168$
2391 052116 N166$: CLI CLISTR,CCITT,N167$,<'CCITT'> ; IF "CCITT", SET FLAG
2392 052132 CLI CLIBR,0,N168$ ; CONTINUE AT N168$
2393 052136 N167$: CLI <'>,CSAVR4,N32$ ; IF "OPERATOR", SET FLAG
2394 052142 CLI CLIBR,COPRSL,N168$ ; AND INPUT SPECIFIED STRING
2395 052146 N168$: CLI CLIBR,CTYPE,N160$ ; DO "TYPE", CHECK FOR MORE INPUT
2396 052152 N170$: CLI CLISTR,0,N175$,<'SIZE'> ; ELSE IS WORD "SIZE"
2397 052166 CLI <'>,0,N32$ ; IF YES, FOLLOWED BY "="?
2398 052172 CLI CLIDEC,Csize,N32$ ; STORE NUMBER IN M$SIZE
2399 052176 CLI CLIBR,0,N160$ ; CHECK FOR MORE INFO
2400 052202 N175$: CLI CLISTR,0,N32$,<'COPIES'> ; ELSE IS WORD "COPIES"
2401 052220 CLI <'>,0,N32$ ; IF YES, FOLLOWED BY "="?
2402 052224 CLI CLIDEC,CCPYS,N32$ ; STORE NUMBER IN M$CPYS
2403 052230 CLI CLIBR,0,N160$ ; CHECK FOR MORE INFO
2404 052234 N178$: CLI CLIBR,0,N32$ ; ELSE "ILL COMMAND"
2405
2406 ;SECOND KEYWORD FOR RUN COMMAND
2407
2408 052240 N180$: CLI CLISPA,0,N181$ ;SKIP LEADING SPACES
2409 052244 N181$: CLI CLISTR,CLUPPR,N182$,<'LOOPPAIR'> ; IS NEXT WORD "LOOPPAIR"
2410 052264 CLI CLIBR,0,N185$ ; IF YES, SET "LOOPPAIR" FLAG
2411 052270 N132$: CLI CLISTR,CNALL,N183$,<'ALL'> ; ELSE IS NEXT WORD "ALL"
2412 052302 CLI CLIBR,0,N185$ ; IF YES, SET "ALL" FLAG
2413 052306 N183$: CLI CLISTR,COIR,N184$,<'DIRECT'> ; ELSE IS NEXT WORD "DIRECT"
2414 052324 CLI CLIBR,0,N185$ ; IF YES, SET "DIRECT" FLAG
2415 052330 N184$: CLI CLISTR,CPTARN,N32$,<'PATTERN'> ; ELSE IS NEXT WORD "PATTERN"
2416 052346 N185$: CLI CLIBR,CDEFLT,N186$ ;SEE IF DEFAULT OF 1 PASS
2417 052352 N186$: CLI CLISTR,0,N32$,<' /PASS'> ;PARSE THROUGH SWITCH
2418 052366 CLI <'>,0,N32$ ;PARSE THROUGH ""
2419 052372 CLI CLIDEC,0,N32$ ;GET PASS COUNT
2420 052376 N190$: CLI CLIEXI,CRUN ;RUN TEST AND EXIT
2421
2422 ;REMAINING COMMAND LINE FOR FUNCTION COMMAND
2423
2424 052400 N200$: CLI CLISPA,0,N201$ ; SKTP SPACES
2425 052404 N201$: CLI CLIOCT,CFUNCT,N32$ ; GET OCTAL NUMBER AND DO FUNCT
2426 052410 CLI CLIEXI,0 ; EXIT
2427
2428 ;REMAINING COMMAND LINE FOR UNSAVE COMMAND
2429
2430 052412 N210$: CLI CLISPA,0,N212$ ;SKIP SPACES
2431 052416 N212$: CLI <'>,0,N215$ ;PARSE THROUGH '/'
2432 052422 CLI CLIEXI,CUNSVF ;SAVE POINTER TO FILE NAME
2433 052424 N215$: CLI CLIEXI,0
2434
2435 *****
2436 ;
2437 ; THE ERRTOB MACRO IS REQUIRED IF YOU INTEND TO REPORT ERRORS USING
2438 ; THE "ERROR" MACRO. THE ERRTOB MACRO EXPANDS INTO FOUR WORDS THAT
2439 ; ARE USED BY THE RUNTIME SERVICES DURING AN ERROR CALL: ERROR TYPE,
2440 ; ERROR NUMBER, ADDRESS OF ERROR MESSAGE AND ADDRESS OF MESSAGE
2441 ; BLOCK. THERE MUST BE ONLY ONE ERRTOB IN ANY PROGRAM. THIS SECTION
2442 ; IS OPTIONAL. REMOVE IF IT IF YOU ARE NOT GOING TO USE THE ERROR

```


2443
2444
2445
2447
2448

052426
052426 000000
052430 000000
052432 000000
052434 000000

; MACRO. CHANGE THE POINTER MACRO TO REFLECT THIS SECTION'S DEL-
; ETION IF YOU REMOVE IT.

;*****

ERRTBL

L\$ERRTBL::

ERRTYP:: .WORD 0
ERRNBR:: .WORD 0
ERRMSG:: .WORD 0
ERRBLK:: .WORD 0

```

2450          ,SBTTL  GLOBAL TEXT SECTION
2451
2452          ;**
2453          ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
2454          ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
2455          ; MORE THAN ONE TEST.
2456          ;--
2457 052436      012      015      116  CLI$PM: .ASCIZ  <12><15>/NIE>/          ;NIE PROMPT
                111      105      076
                052441      000
                052444      000
2458 052445      045      116      045  CLIERM: .ASCIZ  /%N%?ILL CMD-BAD SYNTAX?/
                052450      101      077      111
                052453      114      114      040
                052456      103      115      104
                052461      055      102      101
                052464      104      040      123
                052467      131      116      124
                052472      101      130      077
                052475      000
2459 052476      045      116      045  CLINUF: .ASCIZ  /%N%?INCOMPLETE COMMAND?/
                052501      101      077      111
                052504      116      103      117
                052507      115      120      114
                052512      105      124      105
                052515      040      103      117
                052520      115      115      101
                052523      116      104      077
                052526      000
2460 052527      045      116      045  CLINBG: .ASCIZ  /%N%?NUMBER TOO BIG?/
                052532      101      077      116
                052535      125      115      102
                052540      105      122      040
                052543      124      117      117
                052546      040      102      111
                052551      107      077      000
2461 052554      045      116      045  CLIBRX: .ASCIZ  /%N%?BAD RADIX?/
                052557      101      077      102
                052562      101      104      040
                052565      122      101      104
                052570      111      130      077
                052573      000
2462 052574      045      116      045  LDRESP: .ASCIZ  /%N%ANODE %T% HAS RESPONDED./
                052577      101      116      117
                052602      104      105      040
                052605      045      124      045
                052610      101      040      110
                052613      101      123      040
                052616      122      105      123
                052621      120      117      116
                052624      104      105      104
                052627      056      000
2463 052631      045      116      045  RECERR: .ASCIZ  /%N%APACKET RECEIVED WITH UNA ERROR./
                052634      101      120      101
                052637      103      113      105
                052642      124      040      122
                052645      105      103      105
                052650      111      126      105

```

	052653	104	040	127	
	052656	111	124	110	
	052661	040	125	116	
	052664	101	040	105	
	052667	122	122	117	
	052672	122	056	000	
2464	052675	045	116	045	RTRYER: .ASCIZ /#N#ATRANSMISSION ABORTED -- EXCESSIVE COLLISIONS./
	052700	101	124	122	
	052703	101	116	123	
	052706	115	111	123	
	052711	123	111	117	
	052714	116	040	101	
	052717	102	117	122	
	052722	124	105	104	
	052725	040	055	055	
	052730	040	105	130	
	052733	103	105	123	
	052736	123	111	126	
	052741	105	040	103	
	052744	117	114	114	
	052747	111	123	111	
	052752	117	116	123	
	052755	056	000		
2465	052757	045	116	045	BLDMSG: .ASCIZ /#N#D2#A NODE ADDRESSES ADDED, ELAPSED TIME: #D2#A MINUTE./
	052762	104	062	045	
	052765	101	040	116	
	052770	117	104	105	
	052773	040	101	104	
	052776	104	122	105	
	053001	123	123	105	
	053004	123	040	101	
	053007	104	104	105	
	053012	104	054	040	
	053015	105	114	101	
	053020	120	123	105	
	053023	104	040	124	
	053026	111	115	105	
	053031	072	040	045	
	053034	104	062	045	
	053037	101	040	115	
	053042	111	116	125	
	053045	124	105	056	
	053050	000			
2466	053051	045	116	045	ILADMS: .ASCII /#N#ACANNOT USE BROADCAST ADDRESS (FF-FF-FF-FF-FF-FF)/
	053054	101	103	101	
	053057	116	116	117	
	053062	124	040	125	
	053065	123	105	040	
	053070	102	122	117	
	053073	101	104	103	
	053076	101	123	124	
	053101	040	101	104	
	053104	104	122	105	
	053107	123	123	040	
	053112	050	106	106	
	053115	055	106	106	
	053120	055	106	106	

	053123	055	106	106	
	053126	055	106	106	
	053131	055	106	106	
	053134	051			
2467	053135	045	116	045	ILADM1: .ASCIZ /#N#AFOR LOOP TESTING. ADDRESS WAS NOT ADDED TO NODE TABLE.#N/
	053140	101	106	117	
	053143	122	040	114	
	053146	117	117	120	
	053151	040	124	105	
	053154	123	124	111	
	053157	116	107	056	
	053162	040	101	104	
	053165	104	122	105	
	053170	123	123	040	
	053173	127	101	123	
	053176	040	116	117	
	053201	124	040	101	
	053204	104	104	105	
	053207	104	040	124	
	053212	117	040	116	
	053215	117	104	105	
	053220	040	124	101	
	053223	102	114	105	
	053226	056	045	116	
	053231	000			
2468	053232	045	116	045	CADRER: .ASCIZ /#N#PLEASE ENTER TWELVE HEXADECIMAL DIGITS./
	053235	101	120	114	
	053240	105	101	123	
	053243	105	040	105	
	053246	116	124	105	
	053251	122	040	124	
	053254	127	105	114	
	053257	126	105	040	
	053262	110	105	130	
	053265	101	104	105	
	053270	103	111	115	
	053273	101	114	040	
	053276	104	111	107	
	053301	111	124	123	
	053304	056	000		
2469	053306	045	116	045	CADERR: .ASCIZ /#N#ATWELVE HEX-DIGITS REQUIRED FOR ADDRESS./
	053311	101	124	127	
	053314	105	114	126	
	053317	105	040	110	
	053322	105	130	055	
	053325	104	111	107	
	053330	111	124	123	
	053333	040	122	105	
	053336	121	125	111	
	053341	122	105	104	
	053344	040	106	117	
	053347	122	040	101	
	053352	104	104	122	
	053355	105	123	123	
	053360	056	000		
2470	053362	045	116	045	NULSTR: .ASCIZ /#N#AA ZERO LENGTH STRING WAS ENTERED./
	053365	101	101	040	

	053370	132	105	122				
	053373	117	040	114				
	053376	105	116	107				
	053401	124	110	040				
	053404	123	124	122				
	053407	111	116	107				
	053412	040	127	101				
	053415	123	040	105				
	053420	116	124	105				
	053423	122	105	104				
	053426	056	000					
2471	053430	045	116	045	NODADR:	.ASCIZ	/#N#S2#T/	
	053433	123	062	045				
	053436	124	000					
2472	053440	045	123	061	LOGNAM:	.ASCIZ	/#S14#AN#D2/	
	053443	064	045	101				
	053446	116	045	104				
	053451	062	000					
2473	053453	045	123	061	NODTYP:	.ASCIZ	/#S1#T/	
	053456	064	045	124				
	053461	000						
2474	053462	045	116	045	NTBHDR:	.ASCIZ	\#N#ANODE PHYSICAL ADDRESS	NODE LOGICAL NAME
	053465	101	116	117				TYPE(A/T)\
	053470	104	105	040				
	053473	120	110	131				
	053476	123	111	103				
	053501	101	114	040				
	053504	101	104	104				
	053507	122	105	123				
	053512	123	040	040				
	053515	040	040	040				
	053520	116	117	104				
	053523	105	040	114				
	053526	117	107	111				
	053531	103	101	114				
	053534	040	116	101				
	053537	115	105	040				
	053542	040	040	040				
	053545	040	124	131				
	053550	120	105	050				
	053553	101	057	124				
	053556	051	000					
2475	053560	045	116	045	TABFUL:	.ASCIZ	/#N#ATHE #T#A TABLE IS FILLED TO CAPACITY!//	
	053563	101	124	110				
	053566	105	040	045				
	053571	124	045	101				
	053574	040	124	101				
	053577	102	114	105				
	053602	040	111	123				
	053605	040	106	111				
	053610	114	114	105				
	053613	104	040	124				
	053616	117	040	103				
	053621	101	120	101				
	053624	103	111	124				
	053627	131	041	000				
2476	053632	045	116	045	TABEMT:	.ASCIZ	/#N#ATHE #T#A TABLE IS CURRENTLY EMPTY!//	

RIVER

RIVER

	053635	101	124	110	
	053640	105	040	045	
	053643	124	045	101	
	053646	040	124	101	
	053651	102	114	105	
	053654	040	111	123	
	053657	040	103	125	
	053662	122	122	105	
	053665	116	124	114	
	053670	131	040	105	
	053673	115	120	124	
	053676	131	041	000	
2477	053701	116	117	104	NOD: .ASCIZ /NODE/
	053704	105	000		
2478	053706	123	125	115	SUMM: .ASCIZ /SUMMARY/
	053711	115	101	122	
	053714	131	000		
2479	053716	045	114	045	CLRMSG: .ASCIZ /NNWATHE MESSAGE PARAMETERS HAVE BEEN RESET TO:/
	053721	101	124	110	
	053724	105	040	115	
	053727	105	123	123	
	053732	101	107	105	
	053735	040	120	101	
	053740	122	101	115	
	053743	105	124	105	
	053746	122	123	040	
	053751	110	101	126	
	053754	105	040	102	
	053757	105	105	116	
	053762	040	122	105	
	053765	123	105	124	
	053770	040	124	117	
	053773	072	000		
2480	053775	045	116	045	CPYLMT: .ASCIZ /NNWATHE NUMBER OF COPIES MUST BE BETWEEN 1 AND 255./
	054000	101	124	110	
	054003	105	040	116	
	054006	125	115	102	
	054011	105	122	040	
	054014	117	106	040	
	054017	103	117	120	
	054022	111	105	123	
	054025	040	115	125	
	054030	123	124	040	
	054033	102	105	040	
	054036	102	105	124	
	054041	127	105	105	
	054044	116	040	061	
	054047	040	101	116	
	054052	104	040	062	
	054055	065	065	056	
	054060	000			
2481	054061	045	116	045	SIZLMT: .ASCIZ /NNWATHE MESSAGE SIZE [DATA] MUST BE BETWEEN 32 AND 1466 BYTES./
	054064	101	124	110	
	054067	105	040	115	
	054072	105	123	123	
	054075	101	107	105	
	054100	040	123	111	

GLOBAL TEXT SECTION	054103	132	105	040	
	054106	133	104	101	
	054111	124	101	135	
	054114	040	115	125	
	054117	123	124	040	
	054122	102	105	040	
	054125	102	105	124	
	054130	127	105	105	
	054133	116	040	063	
	054136	062	040	101	
	054141	116	104	040	
	054144	061	064	060	
	054147	066	040	102	
	054152	131	124	105	
	054155	123	056	000	
2482	054160	045	116	045	NOCMPR: .ASCIZ /#N#ATHE ADDRESS MARKED FOR DELETION WAS NOT IN THE TABLE./
	054163	101	124	110	
	054166	105	040	101	
	054171	104	104	122	
	054174	105	123	123	
	054177	040	115	101	
	054202	122	113	105	
	054205	104	040	106	
	054210	117	122	040	
	054213	104	105	114	
	054216	105	124	111	
	054221	117	116	040	
	054224	127	101	123	
	054227	040	116	117	
	054232	124	040	111	
	054235	116	040	124	
	054240	110	105	040	
	054243	124	101	102	
	054246	114	105	056	
	054251	000			
2483	054252	045	116	045	UNBOND: .ASCIZ /#N#AAN UNBOUNDED "OPERATOR INPUT" STRING WAS ENTERED./
	054255	101	101	116	
	054260	040	125	116	
	054263	102	117	125	
	054266	116	104	105	
	054271	104	040	042	
	054274	117	120	105	
	054277	122	101	124	
	054302	117	122	040	
	054305	111	116	120	
	054310	125	124	042	
	054313	040	123	124	
	054316	122	111	116	
	054321	107	040	127	
	054324	101	123	040	
	054327	105	116	124	
	054332	105	122	105	
	054335	104	056	000	
2484	054340	045	116	045	ADRDEL: .ASCIZ /#N#ATHE ADDRESS HAS BEEN DELETED FROM THE NODE TABLE./
	054343	101	124	110	
	054346	105	040	101	
	054351	104	104	122	

	054354	105	123	123	
	054357	040	110	101	
	054362	123	040	102	
	054365	105	105	116	
	054370	040	104	105	
	054373	114	105	124	
	054376	105	104	040	
	054401	106	122	117	
	054404	115	040	124	
	054407	110	105	040	
	054412	116	117	104	
	054415	105	040	124	
	054420	101	102	114	
2485	054423	105	056	000	
	054426	045	116	045	LOGDEL: .ASCIZ /#N#ANODE N#D1#A HAS BEEN DELETED FROM THE NODE TABLE./
	054431	101	116	117	
	054434	104	105	040	
	054437	116	045	104	
	054442	061	045	101	
	054445	040	110	101	
	054450	123	040	102	
	054453	105	105	116	
	054456	040	104	105	
	054461	114	105	124	
	054464	105	104	040	
	054467	106	122	117	
	054472	115	040	124	
	054475	110	105	040	
	054500	116	117	104	
	054503	105	040	124	
	054506	101	102	114	
2486	054511	105	056	000	
	054514	045	116	045	TABCLR: .ASCIZ /#N#ATHE #T#A TABLE HAS BEEN CLEARED./
	054517	101	124	110	
	054522	105	040	045	
	054525	124	045	101	
	054530	040	124	101	
	054533	102	114	105	
	054536	040	110	101	
	054541	123	040	102	
	054544	105	105	116	
	054547	040	103	114	
	054552	105	101	122	
	054555	105	104	056	
	054560	000			
2487	054561	045	116	045	UNSMMSG: .ASCIZ /#N#ATHE NODE TABLE HAS BEEN #T/
	054564	101	124	110	
	054567	105	040	116	
	054572	117	104	105	
	054575	040	124	101	
	054600	102	114	105	
	054603	040	110	101	
	054606	123	040	102	
	054611	105	105	116	
	054614	040	045	124	
	054617	000			
2488	054620	123	101	126	SAVED: .ASCIZ /SAVED./

	054623	105	104	056	
	054626	000			
2489	054627	122	105	123	RESTOR: .ASCIZ /RESTORED./
	054632	124	117	122	
	054635	105	104	056	
	054640	000			
2490	054641	045	116	040	MSGPRM: .ASCIZ /#N#ATHE CURRENT MESSAGE PARAMETERS ARE:/
	054644	101	124	110	
	054647	105	040	103	
	054652	125	122	122	
	054655	105	116	124	
	054660	040	115	105	
	054663	123	123	101	
	054666	107	105	040	
	054671	120	101	122	
	054674	101	115	105	
	054677	124	105	122	
	054702	123	040	101	
	054705	122	105	072	
	054710	000			
2491	054711	045	116	045	MSG1: .ASCIZ /#N#ATHE COLLECTION OF ALL NODE ADDRESSES COULD TAKE AS LONG AS 40 MINUTES./
	054714	101	124	110	
	054717	105	040	103	
	054722	117	114	114	
	054725	105	103	124	
	054730	111	117	116	
	054733	040	117	106	
	054736	040	101	114	
	054741	114	040	116	
	054744	117	104	105	
	054747	040	101	104	
	054752	104	122	105	
	054755	123	123	105	
	054760	123	040	103	
	054763	117	125	114	
	054766	104	040	124	
	054771	101	113	105	
	054774	040	101	123	
	054777	040	114	117	
	055002	116	107	040	
	055005	101	123	040	
	055010	064	060	040	
	055013	115	111	116	
	055016	125	124	105	
	055021	123	054	000	
2492	055024	045	116	045	MSG11: .ASCIZ /#N#AHOWEVER, IF NO NEW NODES ARE ADDED TO THE TABLE FOR A 10 MINUTE PERIOD/
	055027	101	110	117	
	055032	127	105	126	
	055033	105	122	054	
	055040	040	111	106	
	055043	040	116	117	
	055046	040	116	105	
	055051	127	040	116	
	055054	117	104	105	
	055057	123	040	101	
	055062	122	105	040	
	055065	101	104	104	

	055070	105	104	040	
	055073	124	117	040	
	055076	124	110	105	
	055101	040	124	101	
	055104	102	114	105	
	055107	040	106	117	
	055112	122	040	101	
	055115	040	061	060	
	055120	040	115	111	
	055123	116	125	124	
	055126	105	040	120	
	055131	105	122	111	
	055134	117	104	000	
2493	055137	045	116	045	MSG12: .ASCIZ /#N#ATHE COLLECTION WILL STOP.#N/
	055142	101	124	110	
	055145	105	040	103	
	055150	117	114	114	
	055153	105	103	124	
	055156	111	117	116	
	055161	040	127	111	
	055164	114	114	040	
	055167	123	124	117	
	055172	120	056	045	
	055175	116	000		
2494	055177	045	116	045	MSG2: .ASCIZ /#N#AYOU ENTERED #T#A NODE: #T/
	055202	101	131	117	
	055205	125	040	105	
	055210	116	124	105	
	055213	122	105	104	
	055216	040	045	124	
	055221	045	101	040	
	055224	115	117	104	
	055227	105	072	040	
	055232	045	124	000	
2495	055235	045	116	045	MSG3: .ASCIZ /#N#ATHE SPECIFIED ADDRESS IS: #T/
	055240	101	124	110	
	055243	105	040	123	
	055246	120	105	103	
	055251	111	106	111	
	055254	105	104	040	
	055257	101	104	104	
	055262	122	105	123	
	055265	123	040	111	
	055270	123	072	040	
	055273	045	124	000	
2496	055276	045	116	045	MSG4: .ASCIZ /#N#ATYPE=#T#A,SIZE=#D4#A,COPIES=#D3/
	055301	101	124	131	
	055304	120	105	075	
	055307	045	124	045	
	055312	101	054	123	
	055315	111	132	105	
	055320	075	045	104	
	055323	064	045	101	
	055326	054	103	117	
	055331	120	111	105	
	055334	123	075	045	
	055337	104	063	000	

2497					
2498	055342	045	116	045	HDMSG1: .EVEN
	055345	101	040	105	HDMSG1: .ASCIZ /#N#A ETHERNET DEFAULT ADDRESS (HEX): #T/
	055350	124	110	105	
	055353	122	116	105	
	055356	124	040	104	
	055361	105	106	101	
	055364	125	114	124	
	055367	040	101	104	
	055372	104	122	105	
	055375	123	123	040	
	055400	050	110	105	
	055403	130	051	072	
	055406	040	040	045	
	055411	124	000		
2499	055413	045	116	062	HDMSG2: .ASCIZ /#N2#A ROM MICROCODE VERSION (DECIMAL): #D3/
	055416	045	101	040	
	055421	122	117	115	
	055424	040	115	111	
	055427	103	122	117	
	055432	103	117	104	
	055435	105	040	126	
	055440	105	122	123	
	055443	111	117	116	
	055446	040	050	104	
	055451	105	103	111	
	055454	115	101	114	
	055457	051	072	040	
	055462	045	104	063	
	055465	000			
2500	055466	045	116	062	HDMSG3: .ASCIZ /#N2#A SWITCH PACK SET FOR :/
	055471	045	101	040	
	055474	123	127	111	
	055477	124	103	110	
	055502	040	120	101	
	055505	103	113	040	
	055510	123	105	124	
	055513	040	106	117	
	055516	122	040	072	
	055521	000			
2501	055522	045	116	045	HDMSG4: .ASCIZ /#N#A REMOTE AND POWER UP BOOT ENABLED/
	055525	101	040	040	
	055530	040	040	040	
	055533	040	040	040	
	055536	122	105	115	
	055541	117	124	105	
	055544	040	101	116	
	055547	104	040	120	
	055552	117	127	105	
	055555	122	040	125	
	055560	120	040	102	
	055563	117	117	124	
	055566	040	105	116	
	055571	101	102	114	
	055574	105	104	000	
2502	055577	045	116	045	HDMSG5: .ASCIZ /#N#A REMOTE BOOT ENABLED WITH ROM/
	055602	101	040	040	

	055605	040	040	040		
	055610	040	040	040		
	055613	122	105	115		
	055616	117	124	105		
	055621	040	102	117		
	055624	117	124	040		
	055627	105	116	101		
	055632	102	114	105		
	055635	104	040	127		
	055640	111	124	110		
	055643	040	122	117		
	055646	115	000			
2503	055650	045	116	045	HMSG6: .ASCIZ /%N%A	REMOTE BOOT ENABLED/
	055653	101	040	040		
	055656	040	040	040		
	055661	040	040	040		
	055664	122	105	115		
	055667	117	124	105		
	055672	040	102	117		
	055675	117	124	040		
	055700	105	116	101		
	055703	102	114	105		
	055706	104	000			
2504	055710	045	116	045	HMSG7: .ASCIZ /%N%A	REMOTE BOOT DISABLED/
	055713	101	040	040		
	055716	040	040	040		
	055721	040	040	040		
	055724	122	105	115		
	055727	117	124	105		
	055732	040	102	117		
	055735	117	124	040		
	055740	104	111	123		
	055743	101	102	114		
	055746	105	104	000		
2505	055751	045	116	045	HMSG8: .ASCIZ /%N%A	SELF TEST LOOP ENABLED/
	055754	101	040	040		
	055757	040	040	040		
	055762	040	040	040		
	055765	123	105	114		
	055770	106	040	124		
	055773	105	123	124		
	055776	040	114	117		
	056001	117	120	040		
	056004	105	116	101		
	056007	102	114	105		
	056012	104	000			
2506	056014	045	116	045	HMSG9: .ASCIZ /%N%A	SELF TEST LOOP DISABLED/
	056017	101	040	040		
	056022	040	040	040		
	056025	040	040	040		
	056030	123	105	114		
	056033	106	040	124		
	056036	105	123	124		
	056041	040	114	117		
	056044	117	120	040		
	056047	104	111	123		
	056052	101	102	114		

	056055	105	104	000	
2507					
2508	056060	045	116	045	HELP1: .EVEN
	056063	101	103	117	.ASCIZ \N*ACOMMAND SUMMARY FOR THE NETWORK INTERCONNECT EXERCISER (NIE)\
	056066	115	115	101	
	056071	116	104	040	
	056074	123	125	115	
	056077	115	101	122	
	056102	131	040	106	
	056105	117	122	040	
	056110	124	110	105	
	056113	040	116	105	
	056116	124	127	117	
	056121	122	113	040	
	056124	111	116	124	
	056127	105	122	103	
	056132	117	116	116	
	056135	105	103	124	
	056140	040	105	130	
	056143	105	122	103	
	056146	111	123	105	
	056151	122	040	050	
	056154	116	111	105	
	056157	051	000		
2509	056161	045	116	045	HELP2: .ASCIZ \N*A(IT IS ONLY NECESSARY TO TYPE THE LETTERS IN BRACKETS)\
	056164	101	050	111	
	056167	124	040	111	
	056172	123	040	117	
	056175	116	114	131	
	056200	040	116	105	
	056203	103	105	123	
	056206	123	101	122	
	056211	131	040	124	
	056214	117	040	124	
	056217	131	120	105	
	056222	040	124	110	
	056225	105	040	114	
	056230	105	124	124	
	056233	105	122	123	
	056236	040	111	116	
	056241	040	102	122	
	056244	101	103	113	
	056247	105	124	123	
	056252	051	000		
2510	056254	045	116	062	HELP3: .ASCIZ \N2*A[H]ELP OR ? - TYPES THIS HELP TEXT.\
	056257	045	101	133	
	056262	110	135	105	
	056265	114	120	040	
	056270	117	122	040	
	056273	077	011	055	
	056276	040	124	131	
	056301	120	105	123	
	056304	040	124	110	
	056307	111	123	040	
	056312	110	105	114	
	056315	120	040	124	
	056320	105	130	124	

	056323	056	000			
2511	056325	045	116	062	HELP4: .ASCIZ \N2*A(E)XIT	- RETURN TO THE SUPERVISOR.\
	056330	045	101	133		
	056333	105	135	130		
	056336	111	124	011		
	056341	011	055	040		
	056344	122	105	124		
	056347	125	122	116		
	056352	040	124	117		
	056355	040	124	110		
	056360	105	040	123		
	056363	125	120	105		
	056366	122	126	111		
	056371	123	117	122		
	056374	056	000			
2512	056376	045	116	062	HELP5: .ASCIZ \N2*A(SH)OW [N]ODES	- PRINTS INFORMATION IN CURRENT NODE TABLE.\
	056401	045	101	133		
	056404	123	110	135		
	056407	117	127	040		
	056412	133	116	135		
	056415	117	104	105		
	056420	123	011	055		
	056423	040	120	122		
	056426	111	116	124		
	056431	123	040	111		
	056434	116	106	117		
	056437	122	115	101		
	056442	124	111	117		
	056445	116	040	111		
	056450	116	040	103		
	056453	125	122	122		
	056456	105	116	124		
	056461	040	116	117		
	056464	104	105	040		
	056467	124	101	102		
	056472	114	105	056		
	056475	000				
2513	056476	045	116	062	HELP6: .ASCIZ \N2*A(SH)OW [M]ESSAGE	- PRINTS THE SELECTED MESSAGE TYPE, SIZE AND COPIES.
	056501	045	101	133		
	056504	123	110	135		
	056507	117	127	040		
	056512	133	115	135		
	056515	105	123	123		
	056520	101	107	105		
	056523	011	055	040		
	056526	120	122	111		
	056531	116	124	123		
	056534	040	124	110		
	056537	105	040	123		
	056542	105	114	105		
	056545	103	124	105		
	056550	104	040	115		
	056553	105	123	123		
	056556	101	107	105		
	056561	040	124	131		
	056564	120	105	054		
	056567	040	123	111		

GLOBAL TEXT SECTION

	056572	132	105	040	
	056575	101	116	104	
	056600	040	103	117	
	056603	120	111	105	
	056606	123	056	000	
2514	056611	045	116	062	HELP7: .ASCIZ \N2A[SHOW [C]OUNTERS - PRINTS THE LOW LEVEL COUNTERS OF THE HOST NODE.\
	056614	045	101	133	
	056617	123	110	135	
	056622	117	127	040	
	056625	133	103	135	
	056630	117	125	116	
	056633	124	105	122	
	056636	123	040	055	
	056641	040	120	122	
	056644	111	116	124	
	056647	123	040	124	
	056652	110	105	040	
	056655	114	117	127	
	056660	040	114	105	
	056663	126	105	114	
	056666	040	103	117	
	056671	125	116	124	
	056674	105	122	123	
	056677	040	117	106	
	056702	040	124	110	
	056705	105	040	110	
	056710	117	123	124	
	056713	040	116	117	
	056716	104	105	056	
	056721	000			
2515	056722	045	116	062	HELP8: .ASCIZ \N2A[R]UN [L]OOPPAIR/PASS=NN - RUNS THE LOOPPAIR TEST.\
	056725	045	101	133	
	056730	122	135	125	
	056733	116	040	133	
	056736	114	135	117	
	056741	117	120	120	
	056744	101	111	122	
	056747	057	120	101	
	056752	123	123	075	
	056755	116	116	040	
	056760	055	040	122	
	056763	125	110	123	
	056766	040	124	110	
	056771	105	040	114	
	056774	117	117	120	
	056777	120	101	111	
	057002	122	040	124	
	057005	105	123	124	
	057010	056	000		
2516	057012	045	116	062	HELP9: .ASCIZ \N2A[R]UN [A]LL/PASS=NN - RUNS THE NODE-TO-NODE TEST.\
	057015	045	101	133	
	057020	122	135	125	
	057023	116	040	133	
	057026	101	135	114	
	057031	114	057	120	
	057034	101	123	123	
	057037	075	116	116	

	057042	040	055	040	
	057045	122	125	116	
	057050	123	040	124	
	057053	110	105	040	
	057056	116	117	104	
	057061	105	055	124	
	057064	117	055	116	
	057067	117	104	105	
	057072	040	124	105	
	057075	123	124	056	
	057100	000			
2517	057101	045	116	062	HELP10: .ASCIZ \N2%A(R)UN (D)IRECT/PASS=NN - RUNS THE LOOP DIRECT TEST.\
	057104	045	101	133	
	057107	122	135	125	
	057112	116	040	133	
	057115	104	135	111	
	057120	122	105	103	
	057123	124	057	120	
	057126	101	123	123	
	057131	075	116	116	
	057134	040	055	040	
	057137	122	125	116	
	057142	123	040	124	
	057145	110	105	040	
	057150	114	117	117	
	057153	120	040	104	
	057156	111	122	105	
	057161	103	124	040	
	057164	124	105	123	
	057167	124	056	000	
2518	057172	045	116	062	HELP11: .ASCIZ \N2%A(R)UN (P)ATTERN/PASS=NN - RUNS THE MESSAGE PATTERN TEST.\
	057175	045	101	133	
	057200	122	135	125	
	057203	116	040	133	
	057206	120	135	101	
	057211	124	124	105	
	057214	122	116	057	
	057217	120	101	123	
	057222	123	075	116	
	057225	116	040	055	
	057230	040	122	125	
	057233	116	123	040	
	057236	124	110	105	
	057241	040	115	105	
	057244	123	123	101	
	057247	107	105	040	
	057252	120	101	124	
	057255	124	105	122	
	057260	116	040	124	
	057263	105	123	124	
	057266	056	000		
2519	057270	045	116	062	HELP12: .ASCIZ \N2%A(M)ESSAGE/(T)YPE=A/(S)IZE=N/(C)OPIES=M - ALLOWS THE OPERATOR TO\
	057273	045	101	133	
	057276	115	135	105	
	057301	123	123	101	
	057304	107	105	057	
	057307	133	124	135	

	057312	131	120	'05	
	057315	075	101	057	
	057320	133	123	135	
	057323	111	132	105	
	057326	075	116	057	
	057331	133	103	135	
	057334	117	120	111	
	057337	105	123	075	
	057342	115	040	055	
	057345	040	101	114	
	057350	114	117	127	
	057353	123	040	124	
	057356	110	105	040	
	057361	117	120	105	
	057364	122	101	124	
	057367	117	122	040	
	057372	124	117	000	
2520	057375	045	116	045	HELP13: .ASCIZ \N*N*AMQDIY THE DEFAULT MESSAGE TYPE, SIZE AND COPY PARAMETERS.\
	057400	101	115	117	
	057403	104	111	106	
	057406	131	040	124	
	057411	110	105	040	
	057414	104	105	106	
	057417	101	125	114	
	057422	124	040	115	
	057425	105	123	123	
	057430	101	107	105	
	057433	040	124	131	
	057436	120	105	054	
	057441	040	123	111	
	057444	132	105	040	
	057447	101	116	104	
	057452	040	103	117	
	057455	120	131	040	
	057460	120	101	122	
	057463	101	115	105	
	057466	124	105	122	
	057471	123	056	000	
2521	057474	045	116	062	HELP14: .ASCIZ \N2*A[N]ODE ADR/TYPE - ENTERS A PHYSICAL ADDRESS INTO THE NODE\
	057477	045	101	133	
	057502	116	135	117	
	057505	104	105	040	
	057510	101	104	122	
	057513	057	124	131	
	057516	120	105	040	
	057521	055	040	105	
	057524	116	124	105	
	057527	122	123	040	
	057532	101	040	120	
	057535	110	131	123	
	057540	111	103	101	
	057543	114	040	101	
	057546	104	104	122	
	057551	105	123	123	
	057554	040	111	116	
	057557	124	117	040	
	057562	124	110	105	

	057565	040	116	117	
	057570	104	105	000	
2522	057573	045	116	045	HELP15: .ASCIZ \NNWATABLE, THE TYPE CAN BE EITHER [T]ARGET (DEFAULT) OR [A]SSIST,\
	057576	101	124	101	
	057601	102	114	105	
	057604	056	040	040	
	057607	124	110	105	
	057612	040	124	131	
	057615	120	105	040	
	057620	103	101	116	
	057623	040	102	105	
	057626	040	105	111	
	057631	124	110	103	
	057634	122	040	133	
	057637	124	135	101	
	057642	122	107	105	
	057645	124	040	050	
	057650	104	105	106	
	057653	101	125	114	
	057656	124	051	040	
	057661	117	122	040	
	057664	133	101	135	
	057667	123	123	111	
	057672	123	124	056	
	057675	000			
2523	057676	045	116	062	HELP16: .ASCIZ \NN2WA(SU)MMARY - PRINTS A SUMMARY OF THE TEST RESULTS,\
	057701	045	101	133	
	057704	123	125	135	
	057707	115	115	101	
	057712	122	131	040	
	057715	055	040	120	
	057720	122	111	116	
	057723	124	123	040	
	057726	101	040	123	
	057731	125	115	115	
	057734	101	122	131	
	057737	040	117	106	
	057742	040	124	110	
	057745	105	040	124	
	057750	105	123	124	
	057753	040	122	105	
	057756	123	125	114	
	057761	124	123	056	
	057764	000			
2524	057765	045	116	062	HELP17: .ASCIZ \NN2WA(B)UILD - BUILDS A TABLE OF REMOTE NODE PHYSICAL ADDRESSES BY\
	057770	045	101	133	
	057773	102	135	125	
	057776	111	114	104	
	060001	040	055	040	
	060004	102	125	111	
	060007	1:4	104	123	
	060012	040	101	040	
	060015	124	101	102	
	060020	114	105	040	
	060023	117	1	040	
	060026	122	105	115	
	060031	117	124	105	

GLOBAL TEXT SECTION

	060034	040	116	117	
	060037	104	105	040	
	060042	120	110	131	
	060045	123	111	103	
	060050	101	114	040	
	060053	101	104	104	
	060056	122	105	123	
	060061	123	105	123	
	060064	040	102	131	
	060067	000			
2525	060070	045	116	045	HELP18: .ASCIZ \MNA LISTENING TO ID MESSAGES ON THE NI.\
	060073	101	114	111	
	060076	123	124	105	
	060101	116	111	116	
	060104	107	040	124	
	060107	117	040	111	
	060112	104	040	115	
	060115	105	123	123	
	060120	101	107	105	
	060123	123	040	117	
	060126	116	040	124	
	060131	110	105	040	
	060134	116	111	056	
	060137	000			
2526	060140	045	116	062	HELP19: .ASCIZ \MNA[C]LEAR [N]ODE/AOR - REMOVES THE NODE SPECIFIED BY EITHER ADR\
	060143	045	101	133	
	060146	103	135	114	
	060151	105	101	122	
	060154	040	133	116	
	060157	135	117	104	
	060162	105	057	101	
	060165	104	122	040	
	060170	055	040	122	
	060173	105	115	117	
	060176	126	105	123	
	060201	040	124	110	
	060204	105	040	116	
	060207	117	104	105	
	060212	040	123	120	
	060215	105	103	111	
	060220	106	111	105	
	060223	104	040	102	
	060226	131	040	105	
	060231	111	124	110	
	060234	105	122	040	
	060237	101	104	122	
	060242	000			
2527	060243	045	116	045	HELP20: .ASCIZ \MNAOR NODE LOGICAL NAME FROM THE NODE TABLE.\
	060246	101	117	122	
	060251	040	116	117	
	060254	104	105	040	
	060257	114	117	107	
	060262	111	103	101	
	060265	114	040	116	
	060270	101	115	105	
	060273	040	106	122	
	060276	117	115	040	

D6

	060301	124	110	105	
	060304	040	116	117	
	060307	104	105	040	
	060312	124	101	102	
	060315	114	105	056	
	060320	000			
2528	060321	045	116	062	HELP21: .ASCIZ \N2WA(C)LEAR [N]ODE/[AL]L - CLEARS THE NODE TABLE.\
	060324	045	101	133	
	060327	103	135	114	
	060332	105	101	122	
	060335	040	133	116	
	060340	133	117	104	
	060343	105	057	133	
	060346	101	114	135	
	060351	114	040	055	
	060354	040	103	114	
	060357	105	101	122	
	060362	123	040	124	
	060365	110	105	040	
	060370	116	117	104	
	060373	105	040	124	
	060376	101	102	114	
	060401	105	056	000	
2529	060404	045	116	062	HELP22: .ASCIZ \N2WA(C)LEAR [M]ESSAGE - SETS ALL MESSAGE PARAMETERS TO DEFAULT.\
	060407	045	101	133	
	060412	103	135	114	
	060415	105	101	122	
	060420	040	133	115	
	060423	135	105	123	
	060426	123	101	107	
	060431	105	040	055	
	060434	040	123	105	
	060437	124	123	040	
	060442	101	114	114	
	060445	040	115	105	
	060450	123	123	101	
	060453	107	105	040	
	060456	120	101	122	
	060461	101	115	105	
	060464	124	105	122	
	060467	123	040	124	
	060472	117	040	104	
	060475	105	106	101	
	060500	125	114	124	
	060503	056	000		
2530	060505	045	116	062	HELP23: .ASCIZ \N2WA(C)LEAR [S]UMMARY - CLEARS THE TABLE OF SUMMARY TEST DATA.\
	060510	045	101	133	
	060513	103	135	114	
	060516	105	101	122	
	060521	040	133	133	
	060524	135	125	115	
	060527	115	101	122	
	060532	131	040	055	
	060535	040	103	114	
	060540	105	101	122	
	060543	123	040	124	
	060546	110	105	040	

GLOBAL TEXT SECTION						
	060551	124	101	102		
	060554	114	105	040		
	060557	117	106	040		
	060562	123	125	115		
	060565	115	101	122		
	060570	131	040	124		
	060573	105	123	124		
	060576	040	104	101		
	060601	124	101	056		
	060604	000				
E	2531	060605	045	116	062	HELP24: .ASCIZ \N2*A[I]IDENTIFY ADR - USES THE REQUEST ID FUNCTION TO IDENTIFY A REMOTE NOD
		060610	045	101	133	
		060613	111	135	104	
		060616	105	116	124	
		060621	111	106	131	
		060624	040	101	104	
		060627	122	040	055	
		060632	040	125	123	
		060635	105	123	040	
		060640	124	110	105	
		060643	040	122	105	
		060646	121	125	105	
		060651	123	124	040	
		060654	111	104	040	
		060657	106	125	116	
		060662	103	124	111	
		060665	117	116	040	
		060670	124	117	040	
		060673	111	104	105	
		060676	116	124	111	
		060701	106	131	040	
		060704	101	040	122	
		060707	105	115	117	
		060712	124	105	040	
		060715	116	117	104	
		060720	105	040	117	
		060723	116	040	124	
		060726	110	105	040	
		060731	116	111	056	
		060734	000			
	2532	060735	045	116	062	HELP25: .ASCIZ \N2*A[SA]VE - SAVES THE CONTENTS OF THE NODE TABLE.\
		060740	045	101	133	
		060743	123	101	135	
		060746	126	105	040	
		060751	055	040	123	
		060754	101	126	105	
		060757	123	040	124	
		060762	110	105	040	
		060765	103	117	116	
		060770	124	105	116	
		060773	124	123	040	
		060776	117	106	040	
		061001	124	110	105	
		061004	040	116	117	
		061007	104	105	040	
		061012	124	101	102	
		061015	114	105	056	

2533	061020	000			
	061021	045	116	062	HELP26: .ASCIZ \N2%A[U]NSAVE - REPLACES THE CURRENT NODE TABLE WITH THE SAVED ONE.\
	061024	045	101	133	
	061027	125	135	116	
	061032	123	101	126	
	061035	105	040	055	
	061040	040	122	105	
	061043	120	114	101	
	061046	103	105	123	
	061051	040	124	110	
	061054	105	040	103	
	061057	125	122	122	
	061062	105	116	124	
	061065	040	116	117	
	061070	104	105	040	
	061073	124	101	102	
	061076	114	105	040	
	061101	127	111	124	
	061104	110	040	124	
	061107	110	105	040	
	061112	123	101	126	
	061115	105	104	040	
	061120	117	116	105	
	061123	056	000		
2534	061125	045	116	045	HELP27: .ASCIZ \N#S8#ANOTES: 1) ADR IS THE PHYSICAL ADDRESS OF A NODE ON THE NI.\
	061130	123	070	045	
	061133	101	116	117	
	061136	124	105	123	
	061141	072	040	061	
	061144	051	040	101	
	061147	104	122	040	
	061152	111	123	040	
	061155	124	110	105	
	061160	040	120	110	
	061163	131	123	111	
	061166	103	101	114	
	061171	040	101	104	
	061174	104	122	105	
	061177	123	123	040	
	061202	117	106	040	
	061205	101	040	116	
	061210	117	104	105	
	061213	040	117	116	
	061216	040	124	110	
	061221	105	040	116	
	061224	111	056	000	
UL 2535	061227	045	116	045	HELP28: .ASCIZ \N#S8#A 2) PASS COUNT IS A DECIMAL NUMBER BETWEEN 1 AND 65534. A DEFA
	061232	123	070	045	
	061235	101	040	040	
	061240	040	040	040	
	061243	040	040	062	
	061246	051	040	120	
	061251	101	123	123	
	061254	040	103	117	
	061257	125	11	124	
	061262	040	111	123	
	061265	040	101	040	

	061270	104	105	103		
	061273	111	115	101		
	061276	114	040	116		
	061301	125	115	102		
	061304	105	122	040		
	061307	102	105	124		
	061312	127	105	105		
	061315	116	040	061		
	061320	040	101	116		
	061323	104	040	066		
	061326	065	065	063		
	061331	064	056	040		
	061334	101	040	104		
	061337	105	106	101		
	061342	125	114	124		
	061345	000				
2536	061346	045	116	045	HELP29: .ASCIZ \NNS8WA	VALUE OF 1 IS ASSUMED.\
	061351	123	070	045		
	061354	101	040	040		
	061357	040	040	040		
	061362	040	040	040		
	061365	040	040	126		
	061370	101	114	125		
	061373	105	040	117		
	061376	106	040	061		
	061401	040	111	123		
	061404	040	101	123		
	061407	123	125	115		
	061412	105	104	056		
	061415	000				
2537	061416	045	116	045	HELP30: .ASCIZ \NNS8WA	SPECIFYING -1 CAUSES THE TEST TO BE RUN INDEFINATELY.\
	061421	123	070	045		
	061424	101	040	040		
	061427	040	040	040		
	061432	040	040	040		
	061435	040	040	123		
	061440	120	105	103		
	061443	111	106	131		
	061446	111	116	107		
	061451	040	055	061		
	061454	040	103	101		
	061457	125	123	105		
	061462	123	040	124		
	061465	110	105	040		
	061470	124	105	123		
	061473	124	040	124		
	061476	117	040	102		
	061501	105	040	122		
	061504	125	116	040		
	061507	111	116	104		
	061512	105	106	111		
	061515	116	101	124		
	061520	105	114	131		
	061523	056	000			
2538					.EVEN	
2539					:	
2540					:	TEST MESSAGES AND ARGUMENTS

GLOBAL TEXT SECTION

```

2541
2542
2543 061526      045      116      045 PASABT: .ASCIZ /*NNA PASS ABORTED!/*
      061531      101      040      120
      061534      101      123      123
      061537      040      101      102
      061542      117      122      124
      061545      105      104      041
      061550      000
2544 061551      045      116      045 TSTMS1: .ASCIZ /*NNTNA TEST -- /
      061554      124      045      101
      061557      040      124      105
      061562      123      124      040
      061565      055      055      040
      061570      000
2545 061571      045      116      045 TSTMS2: .ASCIZ /*NNTNA NODE: *T/*
      061574      124      045      101
      061577      040      116      117
      061602      104      105      072
      061605      040      045      124
      061610      000
2546 061611      045      124      045 TSTMS3: .ASCIZ /*TNA ERROR/
      061614      101      040      105
      061617      122      122      117
      061622      122      000
2547 061624      045      116      045 TSTMS4: .ASCIZ /*NNTNA NODE: *T*NA NODE: *T/*
      061627      124      045      101
      061632      040      116      117
      061635      104      105      072
      061640      040      045      124
      061643      045      124      045
      061646      101      040      116
      061651      117      104      105
      061654      072      040      045
      061657      124      000
2548 061661      045      101      040 OK:      .ASCIZ /*A - RESPONSE OK/
      061664      055      040      122
      061667      105      123      120
      061672      117      116      123
      061675      105      040      117
      061700      113      000
2549 061702      045      116      045 OKRE:   .ASCIZ /*NNA - RECEIVE ASSIST - RESPONSE OK/
      061705      101      040      055
      061710      040      122      105
      061713      103      105      111
      061716      126      105      040
      061721      101      123      123
      061724      111      123      124
      061727      040      055      040
      061732      122      105      123
      061735      120      117      116
      061740      123      105      040
      061743      117      113      000
2550 061746      045      116      045 OKTR:   .ASCIZ /*NNA - TRANSMIT ASSIST - RESPONSE OK/
      061751      101      040      055
      061754      040      124      122
      061757      101      116      123

```


	061762	115	111	124	
	061765	040	101	123	
	061770	123	111	123	
	061773	124	040	055	
	061776	040	122	105	
	062001	123	120	117	
	062004	116	123	105	
	062007	040	117	113	
	062012	000			
2551	062013	045	116	045	OKFU: .ASCIZ /NNA - FULL ASSIST - RESPONSE OK/
	062016	101	040	055	
	062021	040	106	125	
	062024	114	114	040	
	062027	101	123	123	
	062032	111	123	124	
	062035	040	055	040	
	062040	122	105	123	
	062043	120	117	116	
	062046	123	105	040	
	062051	117	113	000	
2552	062054	045	116	045	MESPAT: .ASCIZ /NNAERROR OCCURED WITH TNA MESSAGE TYPE/
	062057	101	105	122	
	062062	122	117	122	
	062065	040	117	103	
	062070	103	125	122	
	062073	105	104	040	
	062076	127	111	124	
	062101	110	040	045	
	062104	124	045	101	
	062107	040	115	105	
	062112	123	123	101	
	062115	107	105	040	
	062120	124	131	120	
	062123	105	000		
2553	062125	045	101	040	MESPA1: .ASCIZ /NA DATA PATTERN: T/
	062130	104	101	124	
	062133	101	040	120	
	062136	101	124	124	
	062141	105	122	116	
	062144	072	040	045	
	062147	124	000		
2554	062151	101	114	114	ALLNOD: .ASCIZ /ALL NODE/
	062154	040	116	117	
	062157	104	105	000	
2555	062162	114	117	117	LUPAIR: .ASCIZ /LOPPAIR/
	062165	120	120	101	
	062170	111	122	000	
2556	062173	114	117	117	DIRECT: .ASCIZ /LOOP DIRECT/
	062176	120	040	104	
	062201	111	122	105	
	062204	103	124	000	
2557	062207	106	125	114	FULAST: .ASCIZ /FULL ASSIST/
	062212	114	040	101	
	062215	123	123	111	
	062220	123	124	000	
2558	062223	124	122	101	TRAST: .ASCIZ /TRANSMIT ASSIST/
	062226	116	123	115	

	062231	111	124	040		
	062234	101	123	123		
	062237	111	123	124		
	062242	000				
2559	062243	122	105	103	RECAST: .ASCIZ	/RECEIVE ASSIST/
	062246	105	111	126		
	062251	105	040	101		
	062254	123	123	111		
	062257	123	124	000		
2560	062262	115	105	123	PATRN: .ASCIZ	/MESSAGE PATTERN/
	062265	123	101	107		
	062270	105	040	120		
	062273	101	124	124		
	062276	105	122	116		
	062301	000				
2561	062302	116	117	040	NORESP: .ASCIZ	/NO RESPONSE/
	062305	122	105	123		
	062310	120	117	116		
	062313	123	105	000		
2562	062316	105	130	103	RETRY: .ASCIZ	/EXCESSIVE COLLISION/
	062321	105	123	123		
	062324	111	126	105		
	062327	040	103	117		
	062332	114	114	111		
	062335	123	111	117		
	062340	116	000			
2563	062342	114	105	116	LENGTH: .ASCIZ	/LENGTH/
	062345	107	124	110		
	062350	000				
2564	062351	104	101	124	COMPAR: .ASCIZ	/DATA COMPARISON/
	062354	101	040	103		
	062357	117	115	120		
	062362	101	122	111		
	062365	123	117	116		
2565	062370	000				
2566					.EVEN	
2567	062372	101	114	120	MSGTY0: .ASCIZ	/ALPHA/ ;MESSAGE TYPES
	062375	110	101	000		
2568	062400	117	116	105	MSGTY1: .ASCIZ	/ONES/
	062403	123	000			
2569	062405	132	105	122	MSGTY2: .ASCIZ	/ZEROS/
	062410	117	123	000		
2570	062413	061	101	114	MSGTY3: .ASCIZ	/1ALT/
	062416	124	000			
2571	062420	060	101	114	MSGTY4: .ASCIZ	/0ALT/
	062423	124	000			
2572	062425	103	103	111	MSGTY5: .ASCIZ	/CCITT/
	062430	124	124	000		
2573	062433	117	120	105	MSGTY6: .ASCIZ	/OPER SEL/
	062436	122	040	123		
	062441	105	114	000		
2574	062444	105	130	111	CMDTY1: .ASCIZ	/EXIT/ ;COMMAND TYPES
	062447	124	000			
2575	062451	123	125	115	CMDTY2: .ASCIZ	/SUMMARY/
	062454	115	101	122		
	062457	131	000			

```

2576 062461      102      125      111  CMDTY3: .ASCIZ  /BUILD/
      062464      114      104      000
2577 062467      123      110      117  CMDTY4: .ASCIZ  /SHOW/
      062472      127      000
2578 062474      122      125      116  CMDTY5: .ASCIZ  /RUN/
      062477      000
2579 062500      115      105      123  CMDTY6: .ASCIZ  /MESSAGE/
      062503      123      101      107
      062506      105      000
2580 062510      116      117      104  CMDTY7: .ASCIZ  /NODE/
      062513      105      000
2581 062515      103      114      105  CMDTY8: .ASCIZ  /CLEAR/
      062520      101      122      000
2582 062523      122      105      121  CMDTY9: .ASCIZ  /REQUEST ID/
      062526      125      105      123
      062531      124      040      111
      062534      104      000
2583 062536      116      117      104  ARGTY1: .ASCIZ  /NODES/           ; ARGUMENT TYPES
      062541      105      123      000
2584 062544      115      105      123  ARGTY2: .ASCIZ  /MESSAGES/
      062547      123      101      107
      062552      105      123      000
2585 062555      103      117      125  ARGTY3: .ASCIZ  /COUNTERS/
      062560      116      124      105
      062563      122      123      000
2586 062566      114      117      117  ARGTY4: .ASCIZ  /LOOPAIR/
      062571      120      120      101
      062574      111      122      000
2587 062577      101      114      114  ARGTY5: .ASCIZ  /ALL/
      062602      000
2588 062603      040      101      123  ARGTY6: .ASCIZ  / ASSIST/
      062606      123      111      123
      062611      124      000
2589 062613      124      101      122  ARGTY7: .ASCIZ  /TARGET/
      062616      107      105      124
      062621      000
2590
2591 062622      045      116      045  NOCLK:  .EVEN
      062625      101      102      101  .ASCIZ  /%N%ABAD CLOCK - PROGRAM WILL HANG ON "TIMEOUT"!!/
      062630      104      040      103
      062633      114      117      103
      062636      113      040      055
      062641      040      120      122
      062644      117      107      122
      062647      101      115      040
      062652      127      111      114
      062655      114      040      110
      062660      101      116      107
      062663      040      117      116
      062666      040      042      124
      062671      111      115      105
      062674      117      125      124
      062677      042      041      041
      062702      000

```

2592
2593
2594

;
; UNA COUNTER INFORMATION MESSAGES

```

2595
2596
2597 062703      045      116      045      CNTR00: .ASCIZ /#N#S5#ACONTENTS OF NODE #T#A INTERNAL COUNTERS:/
      062706      123      065      045
      062711      101      103      117
      062714      116      124      105
      062717      116      124      123
      062722      040      117      106
      062725      040      116      117
      062730      104      105      040
      062733      045      124      045
      062736      101      040      111
      062741      116      124      105
      062744      122      116      101
      062747      114      040      103
      062752      117      125      116
      062755      124      105      122
      062760      123      072      000
2598 062763      045      116      062      CNTR01: .ASCIZ /#N#ASECONDS SINCE LAST ZEROED:#S15#Z5/
      062766      045      101      123
      062771      105      103      117
      062774      116      104      123
      062777      040      123      111
      063002      116      103      105
      063005      040      114      101
      063010      123      124      040
      063013      132      105      122
      063016      117      105      104
      063021      072      045      123
      063024      061      065      045
      063027      132      065      000
2599 063032      045      116      045      CNTR02: .ASCIZ /#N#APACKETS RECEIVED:#S19#T/
      063035      101      120      101
      063040      103      113      105
      063043      124      123      040
      063046      122      105      103
      063051      105      111      126
      063054      105      104      072
      063057      045      123      061
      063062      071      045      124
      063065      000
2600 063066      045      116      045      CNTR03: .ASCIZ /#N#AMULTICAST PACKETS RECEIVED:#S9#T/
      063071      101      115      125
      063074      114      124      111
      063077      103      101      123
      063102      124      040      120
      063105      101      103      113
      063110      105      124      123
      063113      040      122      105
      063116      103      105      111
      063121      126      105      104
      063124      072      045      123
      063127      071      045      124
      063132      000
2601 063133      045      116      045      CNTR04: .ASCIZ /#N#APACKETS REC'D WITH ERROR - BITMAP:#S9#B3/
      063136      101      120      101
      063141      103      113      105

```

GLOBAL TEXT SECTION

	063144	124	123	040	
	063147	122	105	103	
	063152	047	104	040	
	063155	127	111	124	
	063160	110	040	105	
	063163	122	122	117	
	063166	122	040	055	
	063171	040	102	111	
	063174	124	115	101	
	063177	120	072	045	
	063202	123	071	045	
	063205	102	063	000	
2602	063210	045	116	045	CNTR05: .ASCIZ /#N#APACKETS RECEIVED WITH ERROR:#S13#Z5/
	063213	101	120	101	
	063216	103	113	105	
	063221	124	123	040	
	063224	122	105	103	
	063227	105	111	126	
	063232	105	104	040	
	063235	127	111	124	
	063240	110	040	105	
	063243	122	122	117	
	063246	122	072	045	
	063251	123	061	063	
	063254	045	132	065	
	063257	000			
2603	063260	045	116	045	CNTR06: .ASCIZ /#N#ADATA BYTES RECEIVED:#S16#T/
	063263	101	104	101	
	063266	124	101	040	
	063271	102	131	124	
	063274	105	123	040	
	063277	122	105	103	
	063302	105	111	126	
	063305	105	104	072	
	063310	045	123	061	
	063313	066	045	124	
	063316	000			
2604	063317	045	116	045	CNTR07: .ASCIZ /#N#AMULTICAST DATA BYTES RECEIVED:#S6#T/
	063322	101	115	125	
	063325	114	124	111	
	063330	103	101	123	
	063333	124	040	104	
	063336	101	124	101	
	063341	040	102	131	
	063344	124	105	123	
	063347	040	122	105	
	063352	103	105	111	
	063355	126	105	104	
	063360	072	045	123	
	063363	066	045	124	
	063366	000			
2605	063367	045	116	045	CNTR08: .ASCIZ /#N#ARECEIVED PACKETS LOST-INTERNAL:#S10#Z5/
	063372	101	122	105	
	063375	103	105	111	
	063400	126	105	104	
	063403	040	120	101	
	063406	103	113	105	

	063411	124	123	040	
	063414	114	117	123	
	063417	124	055	111	
	063422	116	124	105	
	063425	122	116	101	
	063430	114	072	045	
	063433	123	061	060	
	063436	045	132	065	
	063441	000			
2606	063442	045	116	045	CNTR09: .ASCIZ /#N#ARECEIVED PACKETS LOST -LOCAL: #S12#Z5/
	063445	101	122	105	
	063450	103	105	111	
	063453	126	105	104	
	063456	040	120	101	
	063461	103	113	105	
	063464	124	123	040	
	063467	114	117	123	
	063472	124	040	055	
	063475	114	117	103	
	063500	101	114	072	
	063503	045	123	061	
	063506	062	045	132	
	063511	065	000		
2607	063513	045	116	045	CNTR10: .ASCIZ /#N#APACKETS TRANSMITTED: #S16#T/
	063516	101	120	101	
	063521	103	113	105	
	063524	124	123	040	
	063527	124	122	101	
	063532	116	123	115	
	063535	111	124	124	
	063540	105	104	072	
	063543	045	123	061	
	063546	066	045	124	
	063551	000			
2608	063552	045	116	045	CNTR11: .ASCIZ /#N#AMULTICAST PACKETS TRANSMITTED: #S6#T/
	063555	101	115	125	
	063560	114	124	111	
	063563	103	101	123	
	063566	124	040	120	
	063571	101	103	113	
	063574	105	124	123	
	063577	040	124	122	
	063602	101	116	123	
	063605	115	111	124	
	063610	124	105	104	
	063613	072	045	123	
	063616	066	045	124	
	063621	000			
2609	063622	045	116	045	CNTR12: .ASCIZ /#N#APACKETS TRANSMITTED 3+ TRYS: #S8#T/
	063625	101	120	101	
	063630	103	113	105	
	063633	124	123	040	
	063636	124	122	101	
	063641	116	123	115	
	063644	111	124	124	
	063647	105	104	040	
	063652	063	053	040	

	063655	124	122	131	
	063660	123	072	045	
	063663	123	070	045	
	063666	124	000		
2610	063670	045	116	045	CNTR13: .ASCIZ /#N#APACKETS TRANSMITTED 2 TRYS:#S9#T/
	063673	101	120	101	
	063676	103	113	105	
	063701	124	123	040	
	063704	124	122	101	
	063707	116	123	115	
	063712	111	124	124	
	063715	105	104	040	
	063720	062	040	124	
	063723	122	131	123	
	063726	072	045	123	
	063731	071	045	124	
	063734	000			
2611	063735	045	116	045	CNTR14: .ASCIZ /#N#APACKETS DEFERRED:#S19#T/
	063740	101	120	101	
	063743	103	113	105	
	063746	124	123	040	
	063751	104	105	106	
	063754	106	105	122	
	063757	105	104	072	
	063762	045	123	061	
	063765	071	045	124	
	063770	000			
2612	063771	045	116	045	CNTR15: .ASCIZ /#N#ADATA BYTES TRANSMITTED:#S13#T/
	063774	101	104	101	
	063777	124	101	040	
	064002	102	131	124	
	064005	105	123	040	
	064010	124	122	101	
	064013	116	123	115	
	064016	111	124	124	
	064021	105	104	072	
	064024	045	123	061	
	064027	063	045	124	
	064032	000			
2613	064033	045	116	045	CNTR16: .ASCIZ /#N#AMULTICAST BYTES TRANSMITTED:#S8#T/
	064036	101	115	123	
	064041	114	124	111	
	064044	103	101	123	
	064047	124	040	102	
	064052	131	124	105	
	064055	123	040	124	
	064060	122	101	116	
	064063	123	115	111	
	064066	124	124	105	
	064071	104	072	045	
	064074	123	070	045	
	064077	124	000		
2614	064101	045	116	045	CNTR17: .ASCIZ /#N#ATRANSMIT PACKETS ABORTED-BITMAP:#S8#B6/
	064104	101	124	122	
	064107	101	116	123	
	064112	115	111	124	
	064115	040	120	101	

	064120	103	113	105	
	064123	124	123	040	
	064126	101	102	117	
	064131	122	124	105	
	064134	104	055	102	
	064137	111	124	115	
	064142	101	120	072	
	064145	045	123	070	
	064150	045	102	066	
	064153	000			
2615	064154	045	116	045	CNTR18: .ASCIZ /#N#ATRANSMIT PACKETS ABORTED:#S16#Z5/
	064157	101	124	122	
	064162	101	116	123	
	064165	115	111	124	
	064170	040	120	101	
	064173	103	113	105	
	064176	124	123	040	
	064201	101	102	117	
	064204	122	124	105	
	064207	104	072	045	
	064212	123	061	066	
	064215	045	132	065	
	064220	000			
2616	064221	045	116	045	CNTR19: .ASCIZ /#N#AXMIT COLLISION CHECK FAILURE:#S12#Z5/
	064224	101	130	115	
	064227	111	124	040	
	064232	103	117	114	
	064235	114	111	123	
	064240	111	117	116	
	064243	040	103	110	
	064246	105	103	113	
	064251	040	106	101	
	064254	111	114	125	
	064257	122	105	C72	
	064262	045	123	061	
	064265	052	045	132	
	064270	065	000		
2617					
2618					
2619					
2620					
2621					
2622	064272	125	116	101	EMSG01: .ASCIZ /UNA PORT COMMAND ERROR/
	064275	040	120	117	
	064300	122	124	040	
	064303	103	117	115	
	064306	115	101	116	
	064311	104	040	105	
	064314	122	122	117	
	064317	122	000		
2623	064321	125	116	101	EMSG02: .ASCIZ /UNA FATAL ERROR/
	064324	040	106	101	
	064327	124	101	114	
	064332	040	105	122	
	064335	122	117	122	
	064340	000			
2624	064341	125	116	105	EMSG03: .ASCIZ /UNEXPLAINED UNA INTERRUPT/

ERROR MESSAGES FOR DEUNA DRIVER

	064344	130	120	114	
	064347	101	111	116	
	064352	105	104	040	
	064355	125	116	101	
	064360	040	111	116	
	064363	124	105	122	
	064366	122	125	120	
	064371	124	000		
2625	064373	125	116	113	EMSG04: .ASCIZ /UNKNOWN UNA ERROR/
	064376	116	117	127	
	064401	116	040	125	
	064404	116	101	040	
	064407	105	122	122	
	064412	117	122	000	
2626	064415	125	116	101	EMSG05: .ASCIZ /UNA WON'T READ PCB ADDRESS/
	064420	040	127	117	
	064423	116	047	124	
	064426	040	122	105	
	064431	101	104	040	
	064434	120	103	102	
	064437	040	101	104	
	064442	104	122	105	
	064445	123	123	000	
2627	064450	125	116	101	EMSG06: .ASCIZ /UNABLE TO READ PHYSICAL ADDRESS/
	064453	102	114	105	
	064456	040	124	117	
	064461	040	122	105	
	064464	101	104	040	
	064467	120	110	131	
	064472	123	111	103	
	064475	101	114	040	
	064500	101	104	104	
	064503	122	105	123	
	064506	123	000		
2628	064510	125	116	101	EMSG07: .ASCIZ /UNA WILL NOT GO INTO RUNNING STATE/
	064513	040	127	111	
	064516	114	114	040	
	064521	116	117	124	
	064524	040	107	117	
	064527	040	111	116	
	064532	124	117	040	
	064535	122	125	116	
	064540	116	111	116	
	064543	107	040	123	
	064546	124	101	124	
	064551	105	000		
2629	064553	124	111	115	EMSG08: .ASCIZ /TIMEOUT!--TRANSMIT FLAG NOT SET/
	064556	105	117	125	
	064561	124	041	055	
	064564	055	124	122	
	064567	101	116	123	
	064572	115	111	124	
	064575	040	106	114	
	064600	101	107	040	
	064603	116	117	124	
	064606	040	123	105	
	064611	124	000		

```

GLOBAL TEXT SECTION

2630 064613    105    122    122  MSG09: .ASCIZ  /ERROR DURING TRANSMIT PDMD COMMAND/
      064616    117    122    040
      064621    104    125    122
      064624    111    116    107
      064627    040    124    122
      064632    101    116    123
      064635    115    111    124
      064640    040    120    104
      064643    115    104    040
      064646    103    117    115
      064651    115    101    116
      064654    104    000
2631 064656    124    122    101  MSG10: .ASCIZ  /TRANSMIT RING BOOKKEEPING ERROR/
      064661    116    123    115
      064664    111    124    040
      064667    122    111    116
      064672    107    040    102
      064675    117    117    113
      064700    113    105    105
      064703    120    111    116
      064706    107    040    105
      064711    122    122    117
      064714    122    000
2632 064716    122    105    103  MSG11: .ASCIZ  /RECEIVE RING BOOKKEEPING ERROR/
      064721    105    111    126
      064724    105    040    122
      064727    111    116    107
      064732    040    102    117
      064735    117    113    113
      064740    105    105    120
      064743    111    116    107
      064746    040    105    122
      064751    122    117    122
      064754    000
2633 064755    115    105    123  MSG14: .ASCIZ  /MESSAGE SIZE TOO BIG FOR MAX. PACKET LENGTH/
      064760    123    101    107
      064763    105    040    123
      064766    111    132    105
      064771    040    124    117
      064774    117    040    102
      064777    111    107    040
      065002    106    117    122
      065005    040    115    101
      065010    130    056    040
      065013    120    101    103
      065016    113    105    124
      065021    040    114    105
      065024    116    107    124
      065027    110    000
2634 065031    104    116    111  MSG15: .ASCIZ  /DNI DID NOT SET FROM RESET/
      065034    040    104    111
      065037    104    040    116
      065042    117    124    040
      065045    123    105    124
      065050    040    106    122
      065053    117    115    040
      065056    122    105    123

```

GLOBAL TEXT SECTION

	065061	105	124	000	
2635	065064	125	116	101	EMSG16: .ASCIZ /UNA WILL NOT READ DESCRIPTOR RINGS/
	065067	040	127	111	
	065072	114	114	040	
	065075	116	117	124	
	065100	040	122	105	
	065103	101	104	040	
	065106	104	105	123	
	065111	103	122	111	
	065114	120	124	117	
	065117	122	040	122	
	065122	111	116	107	
	065125	123	000		
2636	065127	103	101	116	EMSG18: .ASCIZ /CAN'T GET INITIAL STATUS INFO FROM UNA/
	065132	047	124	040	
	065135	107	105	124	
	065140	040	111	116	
	065143	111	124	111	
	065146	101	114	040	
	065151	123	124	101	
	065154	124	125	123	
	065157	040	111	116	
	065162	106	117	040	
	065165	106	122	117	
	065170	115	040	125	
	065173	116	101	000	
2637	065176	115	105	123	EMSG19: .ASCIZ /MESSAGE DATA COMPARISON ERROR/
	065201	123	101	107	
	065204	105	040	104	
	065207	101	124	101	
	065212	040	103	117	
	065215	115	120	101	
	065220	122	111	123	
	065223	117	116	040	
	065226	105	122	122	
	065231	117	122	000	
2638	065234	124	117	124	EMSG20: .ASCIZ /TOTAL DATA COMPARE ERRORS/
	065237	101	114	040	
	065242	104	101	124	
	065245	101	040	103	
	065250	117	115	120	
	065253	101	122	105	
	065256	040	105	122	
	065261	122	117	122	
	065264	123	000		
2639	065266	045	116	045	EMSG22: .ASCIZ /N=ANO RESPONSE FROM NODE./
	065271	101	116	117	
	065274	040	122	105	
	065277	123	120	117	
	065302	116	123	105	
	065305	040	106	122	
	065310	117	115	040	
	065313	116	117	104	
	065316	105	056	000	
2640	065321	105	122	122	EMSG23: .ASCIZ /ERROR WHILE ATTEMPTING TO WRITE MODE/
	065324	117	122	040	
	065327	127	110	111	

	065332	114	105	040	
	065335	101	124	124	
	065340	105	115	120	
	065343	124	111	116	
	065346	107	040	124	
	065351	117	040	127	
	065354	122	111	124	
	065357	105	040	115	
	065362	117	104	105	
	065365	000			
2641	065366	124	122	101	MSG24: .ASCIZ /TRANSMIT ERROR, ALL PACKETS NOT TRANSMITTED/
	065371	116	123	115	
	065374	111	124	040	
	065377	105	122	122	
	065402	117	122	054	
	065405	040	101	114	
	065410	114	040	120	
	065413	101	103	113	
	065416	105	124	123	
	065421	040	116	117	
	065424	124	040	124	
	065427	122	101	116	
	065432	123	115	111	
	065435	124	124	105	
	065440	104	000		
2642	065442	105	122	122	MSG25: .ASCIZ /ERROR WHILE ATTEMPTING TO WRITE MULTICAST ADDRESS LIST/
	065445	117	122	040	
	065450	127	110	111	
	065453	114	105	040	
	065456	101	124	124	
	065461	105	115	120	
	065464	124	111	116	
	065467	107	040	124	
	065472	117	040	127	
	065475	122	111	124	
	065500	105	040	115	
	065503	125	114	124	
	065506	111	103	101	
	065511	123	124	040	
	065514	101	104	104	
	065517	122	105	123	
	065522	123	040	114	
	065525	111	123	124	
	065530	000			
2643	065531	105	122	122	MSG30: .ASCIZ /ERROR WHILE ATTEMPTING PORT FUNCTION/
	065534	117	122	040	
	065537	127	110	111	
	065542	114	105	040	
	065545	101	124	124	
	065550	105	115	120	
	065553	124	111	116	
	065556	107	040	120	
	065561	117	122	124	
	065564	040	106	125	
	065567	116	103	124	
	065572	111	117	116	
	065575	000			

GLOBAL TEXT SECTION

2644	065576	125	116	101	EMSG31: .ASCIZ /UNABLE TO READ INTERNAL COUNTERS/
	065601	102	114	105	
	065604	040	124	117	
	065607	040	122	105	
	065612	101	104	040	
	065615	111	116	124	
	065620	105	122	116	
	065623	101	114	040	
	065626	103	117	125	
	065631	116	124	105	
	065634	122	123	000	
2645	065637	045	116	045	EMSG32: .ASCIZ \N#ILLEGAL TARGET/ASSIST PAIR IN NODE TABLE\
	065642	101	111	114	
	065645	114	105	107	
	065650	101	114	040	
	065653	124	101	122	
	065656	107	105	124	
	065661	057	101	123	
	065664	123	111	123	
	065667	124	040	120	
	065672	101	111	122	
	065675	040	111	116	
	065700	040	116	117	
	065703	104	105	040	
	065706	124	101	102	
	065711	114	105	000	
2646	065714	124	111	115	EMSG33: .ASCIZ /TIMEOUT ERROR/
	065717	105	117	125	
	065722	124	040	105	
	065725	122	122	117	
	065730	122	000		
2647	065732	015	012	124	EMSG34: .ASCIZ <15><12>/TIMEOUT OCCURED BEFORE LOOPBACK REPLY/
	065735	111	115	105	
	065740	117	125	124	
	065743	040	117	103	
	065746	103	125	122	
	065751	105	104	040	
	065754	102	105	106	
	065757	117	122	105	
	065762	040	114	117	
	065765	117	120	102	
	065770	101	103	113	
	065773	040	122	105	
	065776	120	114	131	
	066001	000			
2648	066002	045	101	106	EMSG35: .ASCIZ /#AFAILING NODE ADDRESS: #T#N/
	066005	101	111	114	
	066010	111	116	107	
	066013	040	116	117	
	066016	104	105	040	
	066021	101	104	104	
	066024	122	105	123	
	066027	123	072	040	
	066032	045	124	045	
	066035	116	000		
2649	066037	045	101	104	EMSG36: .ASCIZ /#ADATA PATTERN: #T#N/
	066042	101	124	101	

	066045	040	120	101	
	066050	124	124	105	
	066053	122	116	072	
	066056	040	045	124	
	066061	045	116	000	
2650	066064	045	101	106	EMSG37: .ASCIZ /#AFAILING TARGET NODE ADDRESS: #T#N/
	066067	101	111	114	
	066072	111	116	107	
	066075	040	124	101	
	066100	122	107	105	
	066103	124	040	116	
	066106	117	104	105	
	066111	040	101	104	
	066114	104	122	105	
	066117	123	123	072	
	066122	040	045	124	
	066125	045	116	000	
2651	066130	045	101	106	EMSG38: .ASCIZ /#AFAILING ASSIST NODE ADDRESS: #T#N/
	066133	101	111	114	
	066136	111	116	107	
	066141	040	101	123	
	066144	123	111	123	
	066147	124	040	116	
	066152	117	104	105	
	066155	040	101	104	
	066160	104	122	105	
	066163	123	123	072	
	066166	040	045	124	
	066171	045	116	000	
2652	066174	015	012	124	EMSG40: .ASCIZ <15><12>/TIMEOUT OCCURED - LOOP MESSAGE TYPE - RECEIVE ASSIST/
	066177	111	115	105	
	066202	117	125	124	
	066205	040	117	103	
	066210	103	125	122	
	066213	105	104	040	
	066216	045	040	114	
	066221	117	117	120	
	066224	040	115	105	
	066227	123	123	101	
	066232	107	105	040	
	066235	124	131	120	
	066240	105	040	055	
	066243	040	122	105	
	066246	103	105	111	
	066251	126	105	040	
	066254	101	123	123	
	066257	111	123	124	
	066262	000			
2653	066263	015	012	124	EMSG41: .ASCIZ <15><12>/TIMEOUT OCCURED - LOOP MESSAGE TYPE - TRANSMIT ASSIST/
	066266	111	115	105	
	066271	117	125	124	
	066274	040	117	103	
	066277	103	125	122	
	066302	105	104	040	
	066305	055	040	114	
	066310	117	117	120	
	066313	040	115	105	

J7

GLOBAL TEXT SECTION

	066316	123	123	101	
	066321	107	105	040	
	066324	124	131	120	
	066327	105	040	055	
	066332	040	124	122	
	066335	101	116	123	
	066340	115	111	124	
	066343	040	101	123	
	066346	123	111	123	
	066351	124	000		
2654	066353	015	012	124	EMSG42: .ASCIZ <15><12>/TIMEOUT OCCURED - LOOP MESSAGE TYPE - FULL ASSIST/
	066356	111	115	105	
	066361	117	125	124	
	066364	040	117	103	
	066367	103	125	122	
	066372	105	104	040	
	066375	055	040	114	
	066400	117	117	120	
	066403	040	115	105	
	066406	123	123	101	
	066411	107	105	040	
	066414	124	131	120	
	066417	105	040	055	
	066422	040	106	125	
	066425	114	114	040	
	066430	101	123	123	
	066433	111	123	124	
	066436	000			
2655					
2656					.EVEN
2657	066440	045	116	045	SIMSG1: .ASCIZ /#N#A NODE DEFAULT ADDRESS: #T/
	066443	101	040	040	
	066446	116	117	104	
	066451	105	040	104	
	066454	105	106	101	
	066457	125	114	124	
	066462	040	101	104	
	066465	104	122	105	
	066470	123	123	072	
	066473	040	045	124	
	066476	000			
2658	066477	045	116	045	SIMSG2: .ASCIZ /#N#S#ARECEIPT NUMBER: #06/
	066502	123	070	045	
	066505	101	122	105	
	066510	103	105	111	
	066513	120	124	040	
	066516	116	125	115	
	066521	102	105	122	
	066524	072	040	045	
	066527	117	066	000	
2659	066532	045	116	045	SIMSG3: .ASCIZ /#N#A MAINTENANCE VERSION: #22/
	066535	101	040	040	
	066540	040	115	101	
	066543	111	116	124	
	066546	105	116	101	
	066551	116	103	105	
	066554	040	126	105	

GLOBAL TEXT SECTION

	066557	122	123	111	
	066562	117	116	072	
	066565	040	045	132	
	066570	062	000		
2660	066572	045	116	045	SIMSG4: .ASCIZ /#N#S19#AECO: #Z2/
	066575	123	061	071	
	066600	045	101	105	
	066603	103	117	072	
	066606	040	045	132	
	066611	062	000		
2661	066613	045	116	045	SIMSG5: .ASCIZ /#N#S14#AUSER ECO: #Z2/
	066616	123	061	064	
	066621	045	101	125	
	066624	123	105	122	
	066627	040	105	103	
	066632	117	072	040	
	066635	045	132	062	
	066640	000			
2662	066641	045	116	045	SIMSG6: .ASCIZ /#N#S14#AFUNCTION: #02/
	066644	123	061	064	
	066647	045	101	106	
	066652	125	116	103	
	066655	124	111	117	
	066660	116	072	040	
	066663	045	117	062	
	066666	000			
2663	066667	045	116	045	SIMSG7: .ASCIZ /#N#S16#ADEVICE: #02/
	066672	123	061	066	
	066675	045	101	104	
	066700	105	126	111	
	066703	103	105	072	
	066706	040	045	117	
	066711	062	000		
2664					
2665					.EVEN
2666	066714	045	116	045	PCMSG:: .ASCIZ /#N#APC OF CALLING ROUTINE = #06/
	066717	101	120	103	
	066722	040	117	106	
	066725	040	103	101	
	066730	114	114	111	
	066733	116	107	040	
	066736	122	117	125	
	066741	124	111	116	
	066744	105	040	075	
	066747	040	045	117	
	066752	066	000		
2667					.EVEN
2668	066754	045	116	045	CMPE1: .ASCIZ /#N#ABYTE NUMBER:#D4#A(D) EXPECTED=#06#A(O) RECIEVED=#06#A(O)/
	066757	101	102	131	
	066762	124	105	040	
	066765	116	125	115	
	066770	102	105	122	
	066773	072	045	104	
	066776	064	045	101	
	067001	050	104	051	
	067004	040	105	130	
	067007	120	105	107	

	067012	124	105	104	
	067015	075	045	117	
	067020	066	045	101	
	067023	050	117	051	
	067026	040	122	105	
	067031	103	111	105	
	067034	126	105	104	
	067037	075	045	117	
	067042	066	045	101	
	067045	050	117	051	
	067050	000			
2669	067051	045	116	045	CMPER2: .ASCIZ /#N#ATOTAL MISMATCHES IN MESSAGE = #D4/
	067054	101	124	117	
	067057	124	101	114	
	067062	040	115	111	
	067065	123	115	101	
	067070	124	103	110	
	067073	105	123	040	
	067076	111	116	040	
	067101	115	105	123	
	067104	123	101	107	
	067107	105	040	075	
	067112	040	045	104	
	067115	064	000		
2670	067117	045	116	045	LGERMS: .ASCIZ /#N#ALENGTH ERROR -- BYTES EXPECTED: #06#A BYTES RECEIVED: #06/
	067122	101	114	105	
	067125	116	107	124	
	067130	110	040	105	
	067133	122	122	117	
	067136	122	040	055	
	067141	055	040	102	
	067144	131	124	105	
	067147	123	040	105	
	067152	130	120	105	
	067155	103	124	105	
	067160	104	072	040	
	067163	045	117	066	
	067166	045	101	040	
	067171	102	131	124	
	067174	105	123	040	
	067177	122	105	103	
	067202	105	111	126	
	067205	105	104	072	
	067210	040	045	117	
	067213	066	000		
2671	067215	045	116	062	SUMMS1: .ASCIZ /#N2#S8#ANODE: #T/
	067220	045	123	070	
	067223	045	101	116	
	067226	117	104	105	
	067231	072	040	045	
	067234	124	000		
2672	067236	045	116	045	SUMMS2: .ASCIZ /#N#ARX NOT COMPLETE RX COMPLETE LENGTH ERRORS/
	067241	101	122	130	
	067244	040	116	117	
	067247	124	040	103	
	067252	117	115	120	
	067255	114	105	124	

	067260	105	040	040	
	067263	040	040	122	
	067266	130	040	103	
	067271	117	115	120	
	067274	114	105	124	
	067277	105	040	040	
	067302	040	040	114	
	067305	105	116	107	
	067310	124	110	040	
	067313	105	122	122	
	067316	117	122	123	
	067321	000			
2673	067322	045	116	045	SUMMS3: .ASCIZ /*N*S6*Z5*S12*Z5*S10*Z5/
	067325	123	066	045	
	067330	132	065	045	
	067333	123	061	062	
	067336	045	132	065	
	067341	045	123	061	
	067344	060	045	132	
	067347	065	000		
2674	067351	045	116	045	SUMMS4: .ASCIZ /*N*ACOMPARE ERRORS BYTES COMPARED BYTES XFER/
	067354	101	103	117	
	067357	115	120	101	
	067362	122	105	040	
	067365	105	122	122	
	067370	117	122	123	
	067373	040	040	040	
	067376	040	102	131	
	067401	124	105	123	
	067404	040	103	117	
	067407	115	120	101	
	067412	122	105	104	
	067415	040	040	040	
	067420	040	102	131	
	067423	124	105	123	
	067426	040	130	106	
	067431	105	122	000	
2675	067434	045	116	045	SUMMS5: .ASCIZ /*N*S6*Z5*S8*T/
	067437	123	066	045	
	067442	132	065	045	
	067445	123	070	045	
	067450	124	000		
2676	067452	045	123	065	SUMMS6: .ASCIZ /*S5*T/
	067455	045	124	000	
2677					.EVEN
2678					
2679					

```

2681          .SBTTL GLOBAL ERROR REPORT SECTION
2682
2683          ;++
2684          ; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
2685          ; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION. PRINTB
2686          ; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
2687          ;--
2688
2689
2690          BGNMSG ERR1
2691          067460 012737 000020 003672          MOV      #CEXIT,CFLAG          ERR1::
2692          067466          PRINTX #PCMSG,PCCALL
2693          067466 013746 050564          MOV      PCCALL,-(SP)
2694          067472 012746 066714          MOV      #PCMSG,-(SP)
2695          067476 012746 000002          MOV      #2,-(SP)
2696          067502 010600          MOV      SP,RO
2697          067504 104415          TRAP    C#PNTX
2698          067506 062706 000006          ADD     #6,SP
2699          067512          DOCLN          TRAP    C#DCLN
2700          067512 104444          ENDMSG
2701          067514          L10002: TRAP    C#MSG
2702          067514 104423
2703          BGNMSG ERR2          ERR2::
2704          067516          MOV      R1,-(SP)
2705          067520 013701 002370          MOV      P#TYPE,R1
2706          067524 006301          ASL     R1
2707          2700 067526 062701 003262          ADD     #MSGTAB,R1
2708          2701 067532          PRINTX #EMSG35,#STRBUF
2709          067532 012746 002322          MOV      #STRBUF,-(SP)
2710          067536 012746 066002          MOV      #EMSG35,-(SP)
2711          067542 012746 000002          MOV      #2,-(SP)
2712          067546 010600          MOV      SP,RO
2713          067550 104415          TRAP    C#PNTX
2714          067552 062706 000006          ADD     #6,SP
2715          2702 067556          PRINTX #EMSG36,(R1)          MOV      (R1),-(SP)
2716          067556 011146          MOV      #EMSG36,-(SP)
2717          067560 012746 066037          MOV      #2,-(SP)
2718          067564 012746 000002          MOV      SP,RO
2719          067570 010600          TRAP    C#PNTX
2720          067572 104415          ADD     #6,SP
2721          067574 062706 000006          MOV      (SP)+,R1
2722          2703 067600 012601          ENDMSG          L10003: TRAP    C#MSG
2723          2704 067602          BGNMSG ERR3          ERR3::
2724          067602          PRINTX #EMSG37,#STRBUF
2725          067602 104423
2726          2705          BGNMSG ERR3
2727          2706          PRINTX #EMSG37,#STRBUF
2728          2707          067604 012746 002322          MOV      #STRBUF,-(SP)
2729          067610 012746 066064          MOV      #EMSG37,-(SP)
2730          067614 012746 000002          MOV      #2,-(SP)
2731          067620 010600          MOV      SP,RO
    
```

067622	104415						
067624	062706	000006				TRAP	C#PNTX
2708 067630			PRINTX	@EMSG38, @STRBU1		ADD	@6, SP
067630	012746	002344				MOV	@STRBU1, -(SP)
067634	012746	066130				MOV	@EMSG38, -(SP)
067640	012746	000002				MOV	@2, -(SP)
067644	010600					MOV	SP, RO
067646	104415					TRAP	C#PNTX
067650	062706	000006				ADD	@6, SP
2709 067654			ENDMSG				
067654							
067654	104423					L10004:	
2710						TRAP	C#MSG
2712							
2713							
2714							
2715							
2716							
2717							
2718							
2719							
2720							
2721							
2722							
2723							
2724							
2726							
2727							

: *****
: THESE MESSAGE AREAS ARE USED TO OUTPUT SUPPLEMENTARY INFORMATION
: AFTER AN ERROR CALL. THEY ARE INVOKED BY APPENDING THE NAME
: OF THE AREA TO AN ERROR CALL: ERRXXX 1,ERRORMESSAGE,AREANAME.
: THE CORRESPONDING MESSAGE AREA IS SET UP IN THIS SECTION:
: BGNMSG AREANAME
: [CODE]
: ENDMSG
:
: THE AREAS IN THIS SECTION ARE FOR MESSAGES USED IN MORE THAN ONE
: TEST. USE THE PRINTB (PRINT BASIC) AND PRINTX (PRINT EXTENDED)
: MACROS.
: *****

2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2741
2742
2743
2744
2746
2747
2748
2750
2751
2752
2754
2755
2756
2758
2759
2760
2761
2763
2764
2765
2767
2768
2769
2771
2772
2773
2775
2776
2777
2778
2780
2781
2782
2784
2785
2786
2788
2789
2790
2792
2793
2794
2795
2796
2798
2799

```
.SBTTL GLOBAL SUBROUTINES SECTION

; **
; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
; THAT ARE USED IN MORE THAN ONE TEST.
; **

; **
; FUNCTIONAL DESCRIPTION:
; SUBROUTINE TO...

; ~~~~~
; COMPLETE THE "SUBROUTINE TO..." STATEMENT WITH A FUNCTIONAL
; DESCRIPTION OF THIS SUBROUTINE.
; ~~~~~

; INPUTS:

; ~~~~~
; LIST THE INPUT DATA THAT ARE EXPLICITLY PASSED TO THIS SUBROUTINE.
; ~~~~~

; IMPLICIT INPUTS:

; ~~~~~
; LIST THE INPUT DATA THAT ARE IMPLICITLY USED BY THIS SUBROUTINE;
; FOR EXAMPLE, DATA READ FROM COMMON AREAS.
; ~~~~~

; OUTPUTS:

; ~~~~~
; LIST THE OUTPUT DATA THAT ARE EXPLICITLY GIVEN BY THIS SUBROUTINE
; ~~~~~

; IMPLICIT OUTPUTS:

; ~~~~~
; LIST THE OUTPUT DATA THAT ARE IMPLICITLY GIVEN BY THIS SUBROUTINE;
; FOR EXAMPLE, DATA STORED IN COMMON AREAS.
; ~~~~~

; SUBORDINATE ROUTINES USED:

; ~~~~~
; LIST THE SUBROUTINES CALLED BY THIS SUBROUTINE.
; ~~~~~

; FUNCTIONAL SIDE EFFECTS:

; ~~~~~
; DESCRIBE ANY EFFECTS THIS SUBROUTINE MAY HAVE UPON OTHER
; MODULES OF THE DIAGNOSTIC PROGRAM. AN EXAMPLE OF THIS IS
; THE SUBROUTINE INHIBITS ALL INTERRUPTS WITH PRIORITY 7.
; ~~~~~

; CALLING SEQUENCE:
```

```

2800
2802
2803 ; *****
2804 ; GIVE THE EXACT CALLING SEQUENCE USED TO ACCESS THIS SUBROUTINE.
2805 ; FOR EXAMPLE:  MOV COUNT,R1  ;MOVE INPUT TO R1
2806 ;                JSR  PC,ROUTINE ;GO TO ROUTINE
2807 ;                BCS  ERROR     ;CARRY SET IF ROUTINE HAD ERROR
2808 ; *****
2809 ; --
2810
2811 ; *****
2812 ; INSERT THE CODE FOR THIS SUBROUTINE.  THE NAME OF THE SUBROUTINE SHOULD
2813 ; BE DEFINED WITH A DOUBLE-COLON (::); THIS WILL MAKE THE SUBROUTINE GLOBAL.
2814 ; *****
2815 ; *****
2816 ; *****
2817 ; *****
2818 ; *****
2819 ; *****
2820 ; BEGIN EACH SUBROUTINE AT THE TOP OF A NEW PAGE.
2821 ; *****
2822
2823
2824 .SBTTL CLKSET CLOCK SETUP SUBROUTINE
2825
2826 ; --+
2827 ; FUNCTIONAL DESCRIPTION:
2828 ; THIS SUBROUTINE SETS UP THE CLOCK INFORMATION TABLE FOLLOWING
2829 ; A "CLOCK" CALL EXECUTED IN THE INITIALIZATION CODE.  BUT SINCE
2830 ; THE "CLOCK" CALL SAYS NOTHING ABOUT AN LSI-11'S CLOCK, THE
2831 ; ROUTINE IS ONLY USED IF A LINE OR P-CLOCK IS FOUND.
2832 ;
2833 ; INPUTS - R1 - POINTS TO SUPERVISOR SPACE WHERE CLOCK INFO WAS RETURNED
2834 ;          R2 - POINTS TO "CLK" TABLE WHERE CLOCK INFO WILL BE KEPT
2835 ;
2836 ; OUTPUTS - "CLKCSR" GETS LOADED WITH THE CLOCK'D CSR ADDRESS
2837 ;           "CLKBR"  GETS LOADED WITH THE CLOCK'S INTERRUPT LEVEL
2838 ;           "CLKVEC" GETS LOADED WITH THE CLOCK'S INTERRUPT VECTOR
2839 ;           "CLKHZ"  GETS LOADED WITH THE LINE FREQ. (IN HERTZ)
2840 ;
2841 ; CALLING PROCEDURE:
2842 ;                JSR  PC,CLKSET  ; CALL CLOCK SETUP WITH R1 AND R2 SETUP
2843 ;
2844 ; --+
2845 067656 CLKSET:
2846 067656 012122 MOV (R1)+,(R2)+ ; LOAD CLOCK'S CSR ADDR. INTO "CLKCSR"
2847 067660 012112 MOV (R1)+,(R2)  ; LOAD CLOCK'S INTR. LEVEL INTO "CLKBR"
2848 067662 006312 ASL (R2)       ; ADJUST THE INTR. LEVEL FOR LOADING
2849 067664 006312 ASL (R2)       ; INTO THE PSW WITH A "SETVEC" CALL.
2850 067666 006312 ASL (R2)
2851 067670 006312 ASL (R2)
2852 067672 006322 ASL (R2)+
2853 067674 012122 MOV (R1)+,(R2)+ ; LOAD CLOCK'S INTR. VECTOR INTO "CLKVEC"
2854 067676 012122 MOV (R1)+,(R2)+ ; LOAD CLOCK'S FREQ. INTO "CLKHZ"
2855 067700 000207 RTS PC
2856

```

2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889

.SBTTL CLKINT CLOCK INTERRUPT SERVICE ROUTINE

```

---+
: FUNCTIONAL DESCRIPTION:
:   THIS IS THE CLOCK INTERRUPT SERVICE ROUTINE WHICH TAKES CARE
:   OF KEEPING THE "TIME-SINCE-START" AND COUNTING DOWN ANY OF THE
:   "EVENT" TIMERS. THE TIMERS ARE USED TO TIME COMPLETION OF
:   DEVICE REQUESTS. THE "TIME-SINCE-START" IS USED TO BE LOGGED
:   WITH EACH ENTRY INTO THE EVENT LOG.
:
: IMPLICIT INPUTS - TIMTCK - THE CURRENT NO. OF TICKS LEFT TO BE COUNTED UNTIL
:                   A SECOND HAS BEEN COUNTED OFF
:                   CLKHZ - THE NO. OF TICKS IN A SECOND, DETERMINED BY THE
:                   SYS. LINE FREQ.
:                   TIMMIN & TIMSEC - CURRENT VALUE OF "TIME-SINCE-START" IN
:                   MINUTES AND SECONDS
:                   TIMER 1,2 AND S - CURRENT VALUES OF "EVENT TIMERS"
:
: IMPLICIT OUTPUTS - NEW VALUE OF EVENT TIMER "1" & "2" DECREMENTED BY 1 TICK
:                   IF IT WAS NON-ZERO
:                   NEW VALUE OF EVENT TIMER "S" DECREMENTED BY 1 SECOND IF IT
:                   WAS NON-ZERO
:
: SIDE EFFECTS -    THE CLOCK IS DISABLED UPON ENTRY AND REENABLED WHEN LEAVING
:
: CALLING PROCEDURE - THIS ROUTINE IS CALLED WHEN THE CLOCK INTERRUPTS THRU
:                   "CLKVEC". THE ADDRESS OF THIS ROUTINE WAS LOADED
:                   INTO THE CLOCK'S INTERRUPT VECTOR WITH A "SETVEC" CALL
:
---+

```

2890 067702
2891 067702

BGNSRV CLKINT

CLKINT::

```

2892 067702 005077 113766 CLR      @CLKCSR      ; DISABLE THE CLOCK FROM INTERRUPTING
2893 067706 005337 003712 DEC      TIMTCK      ; DECREMENT THE NO. OF TICKS/SEC
2894 067712 001015          BNE          1$      ; GO CHECK TIMERS
2895 067714 013737 003702 003712 MOV      CLKHZ,TIMTCK ; RESET THE NO. OF TICKS/SEC.
2896 067722 005237 003710 INC      TIMSEC      ; INC. NO OF SECS-SINCE-START
2897 067726 022737 000074 003710 CMP      @60.,TIMSEC ; SEE IF WE'VE COUNTED 60 SEC.S YET
2898 067734 001004          BNE          1$      ; IF NOT, GO CHECK TIMERS
2899 067736 005237 003706 INC      TIMMIN      ; ELSE, INC. MINUTES-SINCE-START
2900 067742 005037 003710 CLR      TIMSEC      ; AND RESTART SECOND COUNTER
2901
2902 067746 005737 003714 1$:   TST      TIMER1      ; SEE IF TIMER1 TIMING ANYTHING
2903 067752 001402          BEQ      2$      ; IF=0, NO, CHECK NEXT TIMER
2904 067754 005337 003714 DEC      TIMER1      ; ELSE DECREMENT THE TIMER VALUE (BY 1 TICK)
2905 067760 005737 003716 2$:   TST      TIMER2      ; SEE IF TIMER2 TIMING ANYTHING
2906 067764 001402          BEQ      3$      ; IF=0, NO, CHECK NEXT TIMER
2907 067766 005337 003716 DEC      TIMER2      ; ELSE DECREMENT TIMER VALUE (BY 1 TICK)
2908 067772 005737 003720 3$:   TST      TIMERS      ; SEE IF TIMERS TIMING ANYTHING
2909 067776 001406          BEQ      4$      ; IF=0, NOTHING BE TIMED, LEAVE
2910 070000 023737 003702 003712 CMP      CLKHZ,TIMTCK ; SEE IF A SECOND HAS BEEN COUNTED OFF
2911 070006 001002          BNE          4$      ; OR IF NO
2912 070010 005337 003720 DEC      TIMERS      ; ELSE, DECREMENT TIMER VALUE (BY 1 SEC.)
2913 070014 013777 003704 113652 4$:   MOV      CLKEN,@CLKCSR ; REENABLE THE CLOCK TO INTERRUPT

```

2914 070022
070022
070022 000002

ENDSRV

L10005:
RTI

2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962

```

.SBTTL PREG14 PRESERVE REGISTERS 1 THROUGH 4 ACROSS SUBROUTINE CALLS
;*****
;INPUTS: THE RELATIVE ADDRESS OF THE CALLED ROUTINE MUST FOLLOW THE
; CALL TO THIS ROUTINE (SEE CALLING SEQUENCE).
;
;OUTPUTS: -REGISTERS 1 THROUGH 4 ARE PRESERVED ACROSS THE CALLED ROUTINE.
; -A CHECK IS MADE TO ENSURE THE HARDWARE STACK HASN'T OVER-RUN
; THE PARAMETER STACK.
;
;CALLING SEQUENCE: THIS IS BEST HANDLED BY THE "CALL" MACRO. THE ACTUAL
; CALL IS:
; JSR R4,PREG14
; .WORD [SUBROUTINE NAME]-ANCHOR
;
;COMMENTS: THIS ROUTINE IS RE-ENTRANT AND RELOCATABLE.
; THIS ROUTINE IS DRS COMPATIBLE.
;
;SUBORDINATE ROUTINES CALLED: THE ROUTINE SPECIFIED IN THE CALL.
;*****

```

```

PREG14: MOV R3,-(SP) ;R4 IS ALREADY ON THE R6 STACK.
MOV R2,-(SP) ;PUSH R3, R2, R1
MOV R1,-(SP) ;
;
MOV R4, PCCALL ;GET THE RELATIVE ADDRESS OF THE CALLED
MOV (R4)+,R1 ;ROUTINE.
;MAKE I) AN ABSOLUTE ADDRESS.
ADD PC,R1
ANCHOR: MOV R4,-(SP) ;SAVE THE RETURN TO THE CALLING ROUTINE.
;CHECK FOR STACK OVER-RUN.
CMP R5,SP
BLO 1$ ;HANDLE STACK OVER-RUN CONDITION.
HALT
1$: JSR PC,(R1) ;CALL THE SPECIFIED ROUTINE.
MOV (SP)+,R4 ;RESTORE THE RETURN TO THE CALLING ROUTINE.
MOV (SP)+,R1 ;RESTORE THE REGISTERS.
MOV (SP)+,R2
MOV (SP)+,R3
RTS R4 ;BACK TO THE CALLING ROUTINE.

```

050564


```

2964
2965      .SBTTL  WAIT      WAIT FOR DEUNA INTERRUPT WITH TIMEOUT
2966
2967      ;++
2968      ; FUNCTIONAL DESCRIPTION:
2969      ;       THIS SUBROUTINE WAITS FOR DNI INTERRUPTS FROM THE DEUNA
2970      ;       OR REPORTS A TIMEOUT.  IF A UNA INTERRUPT HAS OCCURED,
2971      ;       THE SUBROUTINE ERROR IS CALLED TO HANDLE IT.
2972      ;
2973      ;       SUCCESS OR FAILURE IS REPORTED VIA P1.
2974      ;
2975      ; INPUTS -
2976      ;
2977      ;       NONE
2978      ;
2979      ; OUTPUTS-
2980      ;       P1:      SUCCESS/FAILURE      0=SUCCESS/-1=FAILURE
2981      ;
2982      ; CALLING SEQUENCE:
2983      ;       CALL      WAIT
2984      ;       P$POP     P1
2985      ;--
2986 070066 012703 000012 WAIT::  MOV      #10,R3      ; MOVE NO. OF COUNTS TO R3
2987 070072 012704 003714      MOV      #TIMER1,R4   ; AND TIMER TO BE USED TO R4
2988 070076 005002      CLR      R2           ; LOCAL STATUS PARAMETER
2989 070100 010314      MOV      R3,(R4)     ; SET NUMBER OF TICKS. (GLOBAL)
2990 070102 005737 050530 1$:   TST      ERRFLG     ; CHECK IF ERROR OCCURED
2991 070106 001011      BNE      3$         ; BR IF YES
2992 070110 005737 050522      TST      DNIFLG     ; CHECK FOR DNI INTERRUPT
2993 070114 001403      BEQ      2$         ; BR IF INTERRUPT RECEIVED
2994 070116 005037 050522      CLR      DNIFLG
2995 070122 000410      BR      6$
2996 070124 005714      2$:   TST      (R4)     ; HAS TIMER EXPIRED?
2997 070126 001365      BNE      1$         ; BR IF NO TO WAIT FOR INTERRUPT
2998 070130 000403      BR      5$         ; BR TO 5$
2999 070132      3$:   CALL     ERROR     ; CALL ERROR ROUTINE
3000 070140 012702 177777 5$:   MOV      #-1,R2     ; INDICATE FAILURE
3001 070144      6$:   RETURN   R2     ; RETURN WITH SUCCESS/FAILURE INDICATION
3002
3003      .SBTTL  ERROR      HANDLE UNA INTERRUPT ERRORS
3004
3005      ;--+
3006      ; FUNCTIONAL DESCRIPTION:
3007      ;       THIS SUBROUTINE CHECKS THE ERROR FLAGS SET BY
3008      ;       UNAI SR THE INTERRUPT SERVICE ROUTINE AND PRINTS
3009      ;       OUT THE APPROPRIATE ERROR MESSAGES.
3010      ;
3011      ; INPUTS -
3012      ; IMPLICIT:  ERROR FLAGS SHOULD BE SET BY UNAI SR ROUTINES.
3013      ; OUTPUTS -
3014      ; IMPLICIT:  ERROR MESSAGES ARE PRINTED OUT TO THE OPERATOR CONSOLE.
3015      ;
3016      ; CALLING SEQUENCE:
3017      ;       CALL     ERROR
3018      ;
3019      ;--+
3020

```

```

3021 070150 005337 050530      ERROR:: DEC      ERRFLG      ;DECRIMENT ERROR COUNTER TO SHOW
3022                                ;THAT IT HAS BEEN HANDLED
3023 070154 005737 050514      TST      PCEFLG      ;SEE IF PORT COMMAND ERROR
3024 070160 001013              BNE      5$          ; IF YES, BRANCH
3025 070162 005737 050512      TST      FATFLG      ;SEE IF UNA FATAL ERROR
3026 070166 001017              BNE      10$         ; IF YES, BRANCH
3027 070170 005737 050526      TST      BCOUNT     ;SEE IF UNEXPLAINED INTERRUPT
3028 070174 001023              BNE      15$         ; IF YES, BRANCH
3029 070176              ERRDF      4,EMSG04,ERR1 ;ELSE UNKNOWN ERROR
                                TRAP      C$ERDF
                                .WORD      4
                                .WORD      EMSG04
                                .WORD      ERR1
3030 070206 000424              BR        20$
3031 070210 104455              5$:      ERRDF      1,EMSG01,ERR1 ; EXIT
                                ;PORT COMMAND ERROR
                                TRAP      C$ERDF
                                .WORD      1
                                .WORD      EMSG01
                                .WORD      ERR1
3032 070220 005337 050514      DEC      PCEFLG      ; INDICATE THAT IT WAS HANDLED
3033 070224 000415              BR        20$
3034 070226 104455              10$:     ERRDF      2,EMSG02,ERR1 ; EXIT
                                ;UNA FATAL ERROR
                                TRAP      C$ERDF
                                .WORD      2
                                .WORD      EMSG02
                                .WORD      ERR1
3035 070236 005337 050512      DEC      FATFLG      ; KEEP UP ON BOOK KEEPING
3036 070242 000406              BR        20$
3037 070244 104455              15$:     ERRDF      3,EMSG03,ERR1 ; EXIT
                                ;UNEXPLAINED INTERRUPT
                                TRAP      C$ERDF
                                .WORD      3
                                .WORD      EMSG03
                                .WORD      ERR1
3038 070254 005337 050526      DEC      BCOUNT     ; BOOK KEEPING
3039 070260              RETURN          ;RETURN

```

.SBTTL UNAINI INITIALIZE THE UNA

```

3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061

```

```

;*****
; SUBROUTINE TO
;           1) SETS UNA IN THE READY STATE
;           2) INITIALIZES ALL UNA GLOBAL DATA LOCATIONS
;              TO DEFAULT VALUES.
;
; CALLED BY:
;           CALL UNAINI
;
; INPUTS:   NONE
;
; OUTPUTS:  NONE
;
; SIDEAFFECTS: ALL GLOBAL LOCATIONS ARE ZEROED
;*****

```

```

3062
3063
3064
3065
3066
3067
3068 070262 013703 047632 UNAINI: MOV PCSRO, R3 ; MOVE ADDRESS OF PCSRO TO R3
3069 070266 012713 000040 MOV #RSET, (R3) ; HARDWARE RESET UNA
3070
3071 070272 005002 CLR R2 ; LOOP COUNTER INIT
3072 070274 011301 7$: MOV (R3), R1 ; READ PCSRO
3073 070276 032701 004000 BIT #DNT, R1 ; WAIT FOR COMMAND TO FINISH
3074 070302 001006 BNE 9$ ; BACK TIL DNI =1
3075 070304 005302 DEC R2 ; COUNT DOWN DELAY
3076 070306 001372 BNE 7$ ; BACK UNTIL TIMECUT
3077 070310 ERRDF 15,MSG15,ERR1 ; PRINT " DNI DID NOT SET FROM"
; TRAP C$ERDF
; .WORD 15
; .WORD MSG15
; .WORD ERR1
3078 ; " A RESET"
3079
3080 070320 012713 004000 9$: MOV #DNI, (R3) ; WRITE ONE TO CLEAR DNI
3081
3082 ;*****
3083 ; NOW ENABLE INTERRUPTS FOR THE ALL
3084 ; PORT COMMANDS ET AL
3085 ;*****
3086
3087 070324 112713 000100 MOV #INTE, (R3) ; ENABLE INTERRUPTS
3088
3089 ;*****
3090 ; LOAD ADDRESS OF PCB8
3091 ;*****
3092
3093 070330 012763 047666 000004 MOV #PCBB0,4(R3) ; LOWER 16 BITS OF ADRS
3094 070336 005063 000006 CLR 6(R3) ; UPPER 2
3095
3096 070342 CALL COMAND #GETPCB ; LOAD ADDRESS
3097 070354 P$POP R2 ; GET SUCCESS/FAILURE REPORT
3098 070356 001404 BEQ 10$ ; CONTINUE IF OK
3099 070360 ERRDF 5,MSG05,ERR1 ; ELSE REPORT ERROR
; TRAP C$ERDF
; .WORD 5
; .WORD MSG05
; .WORD ERR1
3100
3101
3102 070370 10$: ;*****
3103 ; INIT GLOBAL DATA
3104 070370 005037 047642 CLR PCSROC ; INIT CONTENTS OF CUR PCSRO
3105 070374 005037 047644 CLR PCSR1C ; PCSR1
3106 070400 005037 047646 CLR PCSR2C ; PCSR2
3107 070404 005037 047650 CLR PCSR3C ; PCSR3
3108
3109 ;*****
3110 ; INIT RING POINTERS AND

```

```

3111                                     |          RING ENTRIES          |
3112                                     |*****|
3113                                     |*****|
3114 070410 012737 003772 003756      MOV   #XRING,XRGCUR          ; SET POINTERS TO BEGINING OF RING
3115 070416 012737 003772 003762      MOV   #XRING,XRGNXT
3116 070424 012737 004066 003760      MOV   #RRING,RRGCUR
3117 070432 012737 004066 003764      MOV   #RRING,RRGNXT
3118
3119 070440 012702 000006                MOV   #NO.NTR,R2          ; RESET OWNERSHIP AND STATUS OF
3120 070444 013701 003756                MOV   XRGCUR,R1          ; RING ENTRIES
3121 070450 012761 000000 000004      12$: MOV   #0,4(R1)        ; FOR TRANSMIT RING...
3122 070456                CALL  GETXNX #XRGCUR
3123 070470 005302                DEC   R2
3124 070472 001364                BNE  12$
3125
3126 070474 012702 000006                MOV   #NO.NTR,R2          ; ...AND RECIEVE RING
3127 070500 013701 003760                MOV   RRGCUR,R1
3128 070504 012761 100000 000004      13$: MOV   #100000,4(R1)
3129 070512                CALL  GETRNX #RRGCUR
3130 070524 005302                DEC   R2
3131 070526 001364                BNE  13$
3132
3133                                     |*****|
3134                                     |          CLEAR UP ALL PCB AND UCB,          |
3135                                     |          SET BACK TO DEFAULT VALUES        |
3136                                     |*****|
3137
3138
3139 070530 012701 047750                MOV   #RDDE, R1          ; READ DEF PHY ADDR PCB
3140 070534 005061 000002                CLR  2(R1)
3141 070540 005061 000004                CLR  4(R1)
3142 070544 005061 000006                CLR  6(R1)
3143
3144 070550 012701 047760                MOV   #RDPH, R1         ; READ CURRENT PHY ADDR PCB
3145 070554 005061 000002                CLR  2(R1)
3146 070560 005061 000004                CLR  4(R1)
3147 070564 005061 000006                CLR  6(R1)
3148
3149 070570 012701 047770                MOV   #WDPH, R1         ; WRITE CURRENT PHY ADDR PCB
3150 070574 005061 000002                CLR  2(R1)
3151 070600 005061 000004                CLR  4(R1)
3152 070604 005061 000006                CLR  6(R1)
3153
3154 070610 012701 050100                MOV   #RDRN, R1         ; READ RING FORMAT
3155 070614 005061 000004                CLR  4(R1)
3156 070620 005061 000006                CLR  6(R1)
3157
3158 070624 012701 050110                MOV   #UCB10, R1        ; RING ENTRIES:
3159                .REPT 6
3160                CLR  (R1)+
3161                .ENDR
3162                ; INIT TO 0
3163 070644 012701 050124                MOV   #WDRN, R1         ; WRITE RING FORMAT
3164 070650 005061 000004                CLR  4(R1)
3165 070654 005061 000006                CLR  6(R1)
3166
3167 070660 012701 050150                MOV   #RDCN, R1        ; READ COUNTERS

```

```

3168 070664 005061 000004          CLR      4(R1)
3169
3170 070670 012701 050160          MOV      #UCB12, R1          ; COUNTER ENTRIES
3171 070674 012702 000040          MOV      #40,R2
3172 070700 005021          14$:   CLR      (R1)+
3173 070702 005302          DEC      R2
3174 070704 001375          BNE     14$
3175
3176 070706 012701 050260          MOV      #CLRC, R1          ; CLEAR COUNTERS
3177 070712 005061 000004          CLR      4(R1)
3178
3179 070716 012701 050270          MOV      #RDMO, R1          ; READ MODE
3180 070722 005061 000002          CLR      2(R1)
3181 070726 005061 000004          CLR      4(R1)
3182 070732 005061 000006          CLR      6(R1)
3183
3184 070736 012701 050300          MOV      #WDMO, R1          ; WRITE MODE
3185 070742 005061 000002          CLR      2(R1)
3186 070746 005061 000004          CLR      4(R1)
3187 070752 005061 000006          CLR      6(R1)
3188
3189 070756 012701 050310          MOV      #RDST, R1          ; READ STATUS
3190 070762 005061 000002          CLR      2(R1)
3191 070766 005061 000004          CLR      4(R1)
3192 070772 005061 000006          CLR      6(R1)
3193
3194 070776 012701 050320          MOV      #CLRS, R1          ; READ AND CLEAR STATUS
3195 071002 005061 000002          CLR      2(R1)
3196 071006 005061 000004          CLR      4(R1)
3197 071012 005061 000006          CLR      6(R1)
3198
3199 071016 005037 050516          CLR      NIRCNT          ; CLEAR BUFFERS RECEIVED COUNTER
3200 071022          CALL    FUNCT #WDRNGS    ; WRITE DESCRIPTOR RINGS
3201 071034          P#POP  R2              ; CHECK FOR ERROR
3202 071036 001404          BEQ     15$            ; IF OK, CONTINUE
3203 071040          ERRDF  16,MSG16,ERR1  ; ELSE REPORT ERROR
3203 071040 104455          TRAP   C$ERDF
3203 071042 000020          .WORD  16
3203 071044 065064          .WORD  MSG16
3203 071046 067460          .WORD  ERR1
3204
3205 071050          15$:   CALL    COMAND #STRT    ; PUT UNA IN RUNNING STATE
3206 071062          P#POP  R2              ; CHECK FOR ERROR
3207 071064 001404          BEQ     20$            ; IF OK, CONTINUE
3208 071066          ERRDF  7,MSG07,ERR1  ; ELSE REPORT ERROR
3208 071066 104455          TRAP   C$ERDF
3208 071070 000007          .WORD  7
3208 071072 064510          .WORD  MSG07
3208 071074 067460          .WORD  ERR1
3209
3210 071076          20$:   CALL    FUNCT #RDPHYA    ; READ UNA PHYSICAL ADDRESS
3211 071110          P#POP  R2              ; CHECK FOR ERROR
3212 071112 001404          BEQ     25$            ; IF OK, CONTINUE
3213 071114          ERRDF  6,MSG06,ERR1  ; ELSE, REPORT ERROR
3213 071114 104455          TRAP   C$ERDF
3213 071116 000006          .WORD  6
3213 071120 064450          .WORD  MSG06

```

```

071122 067460 .WORD ERR1
3214
3215 071124 012701 047670 25$: MOV #PCBB2, R1 ; STORE PHYSICAL ADDRESS
3216 071130 012704 047662 MOV #PHYADR, R4
3217 071134 012124 MOV (R1)+, (R4)+ ; MOVE FIRST TWO BYTES
3218 071136 012124 MOV (R1)+, (R4)+ ; AND SECOND TWO
3219 071140 011114 MOV (R1), (R4) ; AND DONE
3220
3221 071142 RETURN
3222
3223 .SBTTL UNAIISR UNA INTERRUPT SERVICE ROUTINE
3224
3225 ;
3226 ; THIS INTERRUPT ROUTINE WILL BE THE ONLY INTERRUPT ROUTINE
3227 ; WHICH THE UNA HARDWARE WILL VECTOR TO UPON HARDWARE INTERRUPT.
3228 ;
3229 ; THE REASON FOR SUCH INTERRUPTS ARE TO BE DETERMINED
3230 ; FROM THE APPROPRIATE BITS IN THE PCSRO.
3231 ;
3232 ; IN ADDITION ALL WRITE ONE TO CLEAR BITS OF THE PCSRO
3233 ; ARE CLEARED FROM THIS ROUTINE.
3234 ;-
3235 071144 UNAIISR:
3236 071144 010146 MOV R1, -(SP) ; SAVE R1
3237 071146 010246 MOV R2, -(SP) ; ...
3238 071150 010346 MOV R3, -(SP) ; ...
3239
3240 071152 005003 CLR R3 ; SETUP WRITE 1 TO CLR MASK
3241 071154 013701 047632 MOV PCSRO, R1 ; GET PCSRO ADDRESS
3242
3243 071160 011103 MOV (R1), R3 ; AND ITS CONTENTS
3244
3245 071162 012137 047642 MOV (R1)+, PCSROC ; SAVE PCSR'S FOR DEBUG
3246 071166 012137 047644 MOV (R1)+, PCSR1C
3247 071172 012137 047646 MOV (R1)+, PCSR2C
3248 071176 011137 047650 MOV (R1), PCSR3C
3249 071202 013701 047632 MOV PCSRO, R1
3250
3251 071206 000303 SWAB R3 ; REORIENT CONTENTS OF PCSRO
3252 071210 110361 000001 MOV R3, 1(R1) ; WRITE ONE TO CLEAR
3253 ; ONLY CLEAR UPPER BYTE
3254 071214 000303 SWAB R3 ; REORIENT CONTENTS OF PCSRO
3255
3256
3257 071216 032703 100400 BIT #SERIFATI, R3 ; ANY FATAL STATUS ??
3258 071222 001403 BEQ 10$
3259
3260 071224 005237 050512 INC FATFLG ; SET FLAG
3261 071230 000443 BR 90$ ; EXIT
3262
3263 071232 032703 040000 10$: BIT #PCEI, R3 ; PORT COMMAND ERROR INTERRUPT?
3264 071236 001402 BEQ 20$ ; NO
3265
3266 071240 005237 050514 INC PCEFLG ; YES, INCREMENT FLAG
3267
3268 071244 032703 020000 20$: BIT #RXI, R3 ; RECV INTERRUPT ??
3269 071250 001402 BEQ 30$ ; NO

```

```

3270 071252 005237 050516          INC      NIRCNT          ;YES, SET FLAG
3271
3272 071256 032703 010000          30$:    BIT      #TXI,R3          ;TRANSMIT INTERRUPT ??
3273 071262 001402                   BEQ      40$                   ;NO
3274 071264 005037 050520          CLR      XFLAG          ;YES, SET FLAG
3275
3276 071270 032703 004000          40$:    BIT      #DNI,R3          ;COMMAND DONE ??
3277 071274 001402                   BEQ      45$                   ;NO
3278 071276 005237 050522          INC      DNIFLG         ;YES, COUNT EACH DNI
3279
3280 071302 032703 002000          45$:    BIT      #RCBI,R3         ;RECIEVE BUFFER UNAVAILABLE?
3281 071306 001402                   BEQ      50$                   ;NO
3282 071310 005237 050524          INC      RBFCNT         ; COUNT THEM
3283
3284 071314 032703 034000          50$:    BIT      #RXI!TXI!DNI,R3  ;CHECK FOR NON-ERROR INTERRUPT
3285 071320 001007                   BNE      90$                   ;EXIT IF ONE OCCURED
3286 071322 032703 142000          BIT      #SERI!PCEI!RCBI,R3    ;CHECK FOR ERROR INTERRUPT
3287 071326 001002                   BNE      80$                   ;IF ONE OCCURED, INCR. ERRFLG
3288 071330 005237 050526          INC      BCOUNT         ;ELSE, NONSENSE INTERRUPT
3289 071334 005237 050530          80$:    INC      ERRFLG
3290 071340 012603          90$:    MOV      (SP)+,R3          ;RESTORE REGISTERS
3291 071342 012602                   MOV      (SP)+,R2          ;RESTORE REGISTERS
3292 071344 012601                   MOV      (SP)+,R1          ;RESTORE REGISTERS
3293 071346 000002                   RTI                          ;AUF WIEDERSEHEN
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

```

.SBTTL COMAND SUBR TO ISSUE A UNA PORT COMMAND

```

;---+
; FUNCTIONAL DESCRIPTION
; THIS SUBROUTINE ISSUES A UNA PORT COMMAND. ERRORS
; ARE HANDLED BY THE SUBROUTINE ERROR AND REPORTED IN
; P2 IF ONE OCCURED.
;
; INPUTS - P1 - THE UNA PORT COMMAND MNEMONIC OF THE
;          DESIRED COMMAND.
; OUTPUTS - P2 - SUCCESS REPORT. CONTAINS 0 FOR SUCCESS
;           -1 IF A UNA ERROR OCCURED. THIS PARAMETER
;           IS PASSED DIRECTLY FROM THE WAIT
;           ROUTINE AND IS UNTOUCHED BY COMAND.
;
; CALLING PROCEDURE - CALL COMAND #<COMMAND TYPE>
; SIDE EFFECTS - IF AN ERROR HAS OCCURED, THE ROUTINE ERROR WILL
;                BE CALLED.
; REGISTER USAGE - R1 CONTAINS THE COMMAND TYPE.
;---+

```

```

3319 071350
3320 071350          COMAND::
3321 071352 052701 000100          P#POP  R1          ;MOVE COMMAND TYPE TO R1
3322 071356 010177 156250          BIS   #INTE,R1    ;ADD INTERRUPT TO COMMAND
3323 071362          MOV   R1,#PCSR0 ;MOV COMMAND TO PCSRO
3324 071370          CALL WAIT      ;WAIT FOR DONE INTERRUPT
3325
3326          10$:    RETURN          ;RETURN - ERROR INFO STILL ON
; PARAMETER STACK FROM WAIT SUB.

```

```

3327 .SBTTL FUNCT SUBR TO PERFORM A UNA PORT FUNCTION
3328
3329
3330 ;---+
3331 ; FUNCTIONAL DESCRIPTION:
3332 ; THIS SUBROUTINE PERFORMS A UNA PORT FUNCTION. THE
3333 ; FUNCTION SPECIFIC PCB IS MOVED INTO THE UNA PCB.
3334 ;
3335 ; INPUTS - P1 - THE UNA PORT FUNCTION MNEMONIC OF THE
3336 ; DESIRED FUNCTION.
3337 ; OUTPUTS - P2 - SUCCESS REPORT. CONTAINS 0 FOR SUCCESS
3338 ; -1 IF A UNA ERROR OCCURED,
3339 ; THIS PARAMETER IS PASSED DIRECTLY FROM THE
3340 ; COMAND SUB AND IS NOT AFFECTED BY FUNCT.
3341 ;
3342 ; CALLING PROCEDURE - CALL FUNCT <FUNCTION TYPE>
3343 ; SIDE EFFECTS - IF AN ERROR HAS OCCURED, THE ROUTINE ERROR WILL
3344 ; BE CALLED.
3345 ; REGISTER USAGE - R1 CONTAINS THE FUNCTION TYPE, WHICH IS TRANSFORMED
3346 ; TO THE ADDRESS OF THE FUNCTION SPECIFIC PCB.
3347 ; R2 CONTAINS THE ADDRESS OF THE UNA PCB.
3348 ;
3349 ;---+
3350
3351 071372 FUNCT:: P$POP R1 ; GET FUNCTION TYPE INTO R1
3352 071374 006301 ASL R1 ; MULTIPLY BY TWO
3353 071376 062701 047676 ADD #FUNTAB,R1 ; ADD FUNCTION TABLE OFFSET
3354 ; R1 NOW CONTAINS ADDRESS OF ADDRESS
3355 ; OF FUNCTION SPECIFIC PCB
3356 071402 012702 047666 MOV #PCB80, R2 ; PUT UNA PCB INTO R2
3357 071406 011101 MOV (R1),R1 ; PUT ADDRESS OF PCB INTO R1
3358 071410 012122 MOV (R1)+,(R2)+ ; MOV PCB'S
3359 071412 012122 MOV (R1)+,(R2)+ ; MOV PCB'S
3360 071414 012122 MOV (R1)+,(R2)+ ; MOV PCB'S
3361 071416 012122 MOV (R1)+,(R2)+ ; MOV PCB'S
3362 071420 CALL COMAND #GETFNT ; ISSUE GET PORT FUNCTION COMMAND
3363 071432 RETURN ; SUCCESS INFO FROM COMAND SUBROUTINE
3364
3365 ; IS STILL ON PARAMETER STACK
3366 .SBTTL XMIT TRANSMIT UNA PACKETS
3367
3368 ;---+
3369 ; FUNCTIONAL DESCRIPTION:
3370 ; THIS SUBROUTINE SETS UP THE TRANSMIT DESCRIPTOR
3371 ; RING ENTRIES AND ISSUES A POLL DEMAND COMMAND TO
3372 ; THE UNA.
3373 ;
3374 ; INPUTS - NONE
3375 ;
3376 ; OUTPUTS - P1 - SUCCESS REPORT. 0 FOR SUCCESS, -1 FOR FAILURE
3377 ;
3378 ; CALL PROCEDURE - CALL XMIT
3379 ; P$POP P1
3380 ;
3381 ; SIDE EFFECTS - THE RING POINTER XRGXNT WILL BE UPDATED TO POINT THE NEXT
3382 ; AVAILABLE ENTRY AFTER THE TRANSMIT OPERATION.
3383 ;

```



```

3384      ; REGISTER USAGE - R1 POINTS TO TIMEOUT TIMER LOCATION
3385      ;
3386      ; R2 IS USED AS A POINTER IF RETRYS IS SET
3387      ; R3 IS USED TO PASS THE SUCCESS/FAILURE MESSAGE BACK
3388      ; R4 IS USED AS A POINTER TO RING ENTRIES OR STATUS INFO.
3389      ;---+
3390 071434 005037 050534 XMIT:: CLR RETRYS
3391 071440 013704 003756 11: MOV XRGCUR,R4 ; MOVE RING ENTRY LOCATION INTO R4
3392 071444 032764 100000 000004 BIT #OWN,4(R4) ; MAKE SURE WE OWN THIS
3393 071452 001125 BNE 401 ; ELSE, BOOKKEEPING ERROR
3394 071454 013714 050566 MOV BUFLN,(R4) ; MOVE BUFFER LENGTH INTO FIRST WORD OF
3395 ; NEXT AVAILABLE RING ENTRY
3396 071460 052764 101400 000004 BIS #OWNISTPIENP,4(R4) ; SET OWNERSHIP, START AND END OF PACKET BITS
3397 071466 012737 000001 050520 20: MOV #1,XFLAG ; SET TRANSMIT FLAG
3398 071474 CALL COMAND #PDMO ; ISSUE PDMO COMMAND
3399 071506 P:POP R3 ; CHECK FOR ERRORS
3400 071510 001126 BNE 501 ; IF YES, EXIT
3401 071512 012701 003716 22: MOV #TIMER2,R1 ; SET UP TO WAIT FOR TRANSMIT TO COMPLETE
3402 071516 012711 000100 MOV #100,(R1)
3403 071522 005737 050520 23: TST XFLAG ; SEE IF TRANSMIT DONE BIT SET
3404 071526 001403 BEQ 241 ; IF SET, SKIP WAIT LOOP
3405 071530 005711 TST (R1) ; ELSE, SEE IF TIMEOUT YET
3406 071532 001373 BNE 231 ; NO, WAIT
3407 071534 000506 BR 451 ; YES, EXIT
3408 071536 032764 100000 000004 24: BIT #OWN,4(R4) ; SEE WHO OWNS THIS ENTRY
3409 071544 001070 BNE 401 ; IF UNA STILL OWNS THIS, SOMETHINGS WRONG
3410 071546 032764 040000 000004 BIT #ERRS,4(R4) ; SEE IF ANY ERRORS
3411 071554 001013 BNE 301 ; IF YES, BRANCH AND TAKE CARE OF THEM
3412 071556 26: CALL GETXNX #XRGCUR ; UPDATE "TRANSMIT RING CURRENT" POINTER
3413 071570 005003 CLR R3 ; INDICATE SUCCESS
3414 071572 023737 003756 003762 CMP XRGCUR,XRGNXT ; SEE IF CURRENT POINTER = NEXT POINTER
3415 071600 001057 BNE 421 ; IF NO, ERROR
3416 071602 000473 BR 551 ; RETURN
3417 071604 032764 016000 000004 30: BIT #DEF!ONE!MORE,4(R4) ; WAS MESSAGE STILL SENT?
3418 071612 001361 BNE 261 ; IF YES, GO TO NEXT ONE
3419 071614 032764 002000 000006 BIT #RTY,6(R4) ; ELSE, DID UNA GIVE UP AFTER 16 TRIES
3420 071622 001434 BEQ 321 ; IF NOT, FATAL DEVICE ERROR, EXIT
3421 071624 005237 050534 INC RETRYS ; IF YES, KEEP COUNT OF THEM
3422 071630 022737 000003 050534 CMP #3,RETRYS ; GIVE UP AFTER 3 TIMES
3423 071636 100747 BMI 261 ; ELSE, SET UP TO RETRANSMIT
3424 071640 CALL GETXNX #XRGCUR ; UPDATE POINTERS
3425 071652 CALL GETXNX #XRGNXT
3426 071664 016402 000002 MOV 2(R4),R2 ; SET UP TO COPY DATA, BUFFER
3427 071670 013704 003756 MOV XRGCUR,R4 ; R2 POINTS TO OLD BUFFER
3428 071674 016403 000002 MOV 2(R4),R3 ; R3 POINTS TO NEW BUFFER
3429 071700 013704 050566 MOV BUFLN,R4 ; R4 COUNTS NUMBER OF BYTES TO COPY
3430 071704 112223 31: MOVB (R2)+,(R3)+ ; COPY DATA
3431 071706 005304 DEC R4
3432 071710 001375 BNE 311 ; HAVE WE COPIED ALL OF IT
3433 071712 000652 BR 11 ; IF YES, TRY AGAIN
3434 071714 104455 32: ERRDF 9,MSG09,ERR1 ; ELSE, FATAL DEVICE ERROR
      071714 104455 TRAP C:ERDF
      071716 000011 .WORD 9
      071720 064613 .WORD MSG09
      071722 067460 .WORD ERR1
3435 071724 000420 BR 501 ; EXIT
3436 071726 40: ERRDF 10,MSG10,ERR1 ; TRANSMIT RING BOOKKEEPING ERROR

```

```

071726 104455
071730 000012
071732 064656
071734 067460
3437 071736 000413
3438 071740 104455
071742 000014
071744 064656
071746 067460
3439 071750 000406
3440 071752 005237 050532
3441 071756 104456
071760 000010
071762 064553
071764 067460
3442 071766 012703 177777
3443 071772
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467 071776
3468 071776 005002
3469 072000 005737 050516
3470 072004 001476
3471 072006 013704 003760
3472 072012 016401 000004
3473 072016 012764 100000 000004
3474 072024 032701 100000
3475 072030 001060
3476 072032 016403 000002
3477 072036 016303 000014
3478 072042 020337 050544
3479 072046 001420
3480 072050 020337 050546
3481 072054 001415

TRAP C1ERDF
.WORD 10
.WORD MSG10
.WORD ERR1

42: BR 50:
ERRDF 12,MSG10,ERR1 ; BOOKKEEPING ERROR, XRGXNT SHOULD = XRGCUR
TRAP C1ERDF
.WORD 12
.WORD MSG10
.WORD ERR1

45: BR 50:
INC TIMEOUT
ERRHRD 8,MSG08,ERR1 ; REPORT ERROR

TRAP C1ERHRD
.WORD 8
.WORD MSG08
.WORD ERR1

50: MOV 0-1,R3 ; ERROR INDICATOR
55: RETURN R3 ; RETURN

.SBTTL RECEVE RECEIVE UNA RING BUFFERS

;---
; FUNCTIONAL DESCRIPTION
; THIS SUBROUTINE TAKES INCOMING DATA BUFFERS FROM
; THE UNA AND CHECKS FOR ERRORS. THIS PROCESS CONTINUES
; FOR ALL PENDING BUFFERS.
;
; INPUTS - NONE
; OUTPUTS - P1 - THE NUMBER OF PACKETS HANDLED BY THIS CALL TO RECEIVE
;
; CALLING PROCEDURE - CALL RECEVE
; P1POP P1
;
; SIDE EFFECTS - THE POINTERS RRGCUR AND RRGXNT ARE UPDATED.
;
; REGISTER USAGE - R1 IS USED TO HOLD CURRENT PACKET STATUS INFORMATION
; R2 COUNTS THE NUMBER OF PACKETS HANDLED
; R4 POINTS TO THE RING DESCRIPTOR ENTRY
;---

RECEVE:
1: CLR R2 ; CLEAR PACKETS HANDLED COUNTER
TST NIRCNT ; SEE IF ANY PACKETS TO RECEIVE
BEQ 60 ; IF NO, EXIT
MOV RRGCUR,R4 ; MOVE CURRENT RECEIVE RING POINTER TO R4
MOV 4(R4),R1 ; MOVE STATUS OF PACKET TO R1
MOV #100000,4(R4) ; RESET RING DESCRIPTOR
BIT @OWN,R1 ; SEE WHO OWNS THIS BUFFER
BNE 45 ; IF STILL OWNED BY UNA, ERROR
MOV 2(R4),R3 ; MOVE BUFFER ADDRESS INTO R3
MOV PROT0(R3),R3 ; MOVE PROTOTYPE INTO R3
CMP R3,PROT0 ; SEE IF IT IS AN ACCEPTABLE PROTOCALL TYPE
BEQ 101 ; IF YES, CONT.
CMP R3,PROT02 ; ELSE CHECK OTHER GOOD TYPE
BEQ 101 ; IF OK, CONT.

```

RECEIVE RECEIVE UNA RING BUFFERS

```

3482 072056          5#:  CALL  GETRNX  #RRGCR
3483 072070          CALL  GETRNX  #RRGNXT
3484 072102 005337 050516  DEC  NIRCNT
3485 072106 000435          BR   60#
3486 072110 032701 040000 10#:  BIT   #ERRS,R1
3487 072114 001413          BEQ  20#
3488 072116 005237 050536  INC  RCVERR
3489 072122          PRINTF #RECERR
      072122 012746 052631          MOV  #RECERR,-(SP)
      072126 012746 000001          MOV  #1,-(SP)
      072132 010600          MOV  SP,R0
      072134 104417          TRAP C#PNTF
      072136 062706 000004          ADD  #4,SP
3490 072142 000745          BR   5#
3491 072144 005237 050540 20#:  INC  RCVBUF
3492 072150 005202          INC  R2
3493 072152 005337 050516 25#:  DEC  NIRCNT
3494 072156          CALL  GETRNX  #RRGCR
3495 072170 000404          BR   60#
3496 072172          ERRDF 11,MSG11,ERR1
      072172 104455          ; RECEIVE RING BOOKKEEPING ERROR
      072174 000013          TRAP C#ERDF
      072176 064716          .WORD 11
      072200 067460          .WORD MSG11
3497 072202          .WORD ERR1
3498          60#:  RETURN R2
3499

```

```

; ELSE, THROW OUT AND UPDATE
;   CURRENT AND NEXT POINTERS
; DECREMENT RECEIVE COUNTER
; EXIT
; SEE IF ANY ERRORS
; FOR NO ERRORS BR TO 20#
; ELSE, INCREMENT RECEIVE ERROR COUNTER
; PRINT ERROR MESSAGE
; UPDATE POINTERS AND RETURN
; INCREMENT GOOD BUFFERS RECEIVED COUNTER
; KEEP COUNT OF HOW MANY BUFFERS RECEIVED
; KEEP BOOKKEEPING IN ORDER
; UPDATE "RECEIVE RING CURRENT" POINTER
; RETURN WITH NUMBER OF ENTRIES HANDLED

```

```

3504      |---+
3505      |      EDPACK                                ETHERNET DATA PACK
3506      |
3507      |      THIS ROUTINE WILL CONVERT A STRING OF HEX CHARACTERS INTO A RIGHT
3508      |      JUSTIFIED BINARY STREAM (WITH LEADING ZEROS), COMPATIBLE WITH
3509      |      ETHERNET CONVENTIONS. THE SOURCE STRING MUST BE FORMATTED USING
3510      |      EITHER A WORD BY WORD HEX DESCRIPTION OR A BYTE BY BYTE HEX
3511      |      DESCRIPTION. THE RETURNED STRING WILL BE BYTE ORIENTED AS
3512      |      REQUIRED BY THE ETHERNET:
3513      |      LO-BYTE-WORD0 HI-BYTE-WORD0 LO-BYTE-WORD1 HI-BYTE-WORD1, ETC.
3514      |
3515      |      INPUT ARGUMENTS          P1 - ADDRESS OF THE SOURCE (HEX) STRING
3516      |                               TO BE CONVERTED TO A BINARY STREAM.
3517      |                               P2 - ADDRESS OF THE DESIRED DESTINATION
3518      |                               BUFFER WHICH WILL ACCEPT BINARY DATA
3519      |                               P3 - LENGTH (IN BYTES) OF THE DESTINATION
3520      |                               BUFFER
3521      |      EXPLICIT OUTPUTS        P4 - ZERO IF SUCCESSFUL, -1 IF BUFFER TOO
3522      |                               LONG OR ODD NUMBER OF HEX CHARACTERS
3523      |      IMPLIED OUTPUTS        THE BUFFER AT P2 WILL CONTAIN A RIGHT
3524      |                               JUSTIFIED BINARY STREAM W/ LEADING ZEROS
3525      |                               AND CORRESPONDING TO HEX STRING AT R5.
3526      |      SUBORDINATE PROCEDURES: HXFORM, (STRIP NONHEX CHARACTERS)
3527      |                               HEXBIN, (HEX TO BINARY CONVERSION)
3528      |      CALLING PROCEDURE       CALL EDPACK P1,P2,P3      ;INPUT P1-P3 PARAMETERS
3529      |                               P#POP  P4                ;OUTPUT  P4 PARAMETER
3530      |      |---+
3531      |
3532      |      LOCDEST: .BLKB  74.                ;MAX NUMBER OF CHARACTERS THAT MAY BE ENTERED
3533      |      SOURCE: .WORD                    ;SOURCE ADDRESS
3534      |
3535      |      EDPACK: P#POP  SOURCE,R4,R3        ;R4-DESTINATION, R3-NUMBER OF CHARS REQD
3536      |                                       ;SOURCE-SRC ADDRESS, ORIENT-WORD/BYTE?
3537      |      CLR      R2                        ;ASSUME NO ERRORS, VALUE RETURNED
3538      |      ASL      R3                        ;NUMBER OF CHARACTERS REQUIRED W/ "0"'S
3539      |      CALL    HXFORM SOURCE,@LOCDEST,R3
3540      |      P#POP  R1,R2                      ;R1-ADDRESS OF LAST CHAR
3541      |                                       ;R2-SUCCESS/FAIL CODE (0/-1)
3542      |      TST      R2                        ;R1 WILL POINT TO RIGHTMOST CHARACTER
3543      |      BNE     9#                          ;RIGHT JUSTIFY BUFFER
3544      |                                       ;CONVERT HEX AT LOCDEST TO BINARY
3545      |      ASR      R3                        ;R3 BYTES IN OUTPUT BIT STREAM
3546      |      CALL    HEXBIN @LOCDEST,R4,R3
3547      |
3548      |      9#:  RETURN R2                      ;RETURN WITH SUCCESS/FAILURE INDICATION
3549      |

```

```

3554
3555      ;---+
3556      ;           HXFORM                               HEX FORMAT ROUTINE
3557      ;
3558      ;           THIS ROUTINE WILL ACCEPT A HEX STRING, AND STRIP OUT THE NON-HEX
3559      ;           CHARACTERS (WITHOUT FLAGGING AN ERROR FOR THE NON-HEX CHARACTERS).
3560      ;           A NULL WILL BE LEFT AS THE END OF STRING DELIMITER.
3561      ;
3562      ;           INPUT ARGUMENTS      P1- ADDRESS OF THE SOURCE ASCII STRING (NULL
3563      ;           DELIMITER AT END OF STRING)
3564      ;           P2- ADDRESS OF THE DESTINATION ASCII HEX STRING
3565      ;           (STRIPPED OF NON-HEX AND RIGHT JUSTIFIED)
3566      ;           EXPLICIT OUPUTS     P3- THE NUMBER OF HEX CHARACTERS DESIRED @R4
3567      ;           P4- WILL CONTAIN THE ADDRESS OF THE LAST
3568      ;           CHARACTER IN THE OUTPUT BUFFER.
3569      ;           P5- THE SUCCESS/FAILURE (0/-1) INDICATOR
3570      ;
3571      ;           IMPLICIT OUTPUTS     THE BUFFER AT R4 WILL CONTAIN FORMATTED
3572      ;           HEX (ASCII) STRING.
3573      ;           CALLING PROCEDURE   CALL HXFORM P1,P2,P3
3574      ;           P#POP P4,P5
3575      ;---
3575 072412 000000      HXN:      .WORD
3576 072414           HXFORM: P#POP      R3,R2,HXN      ;ADDRESS OF SOURCE STRING
3577           ;ADDRESS OF DESTINATION STRING
3578           ;NUMBER OF HEX CHARACTERS DESIRED
3579
3580 072424 010204      MOV      R2,R4      ;DESTINATION ADDRESS, R2: DESTINATION POINTER
3581 072426 063704 072412      ADD      HXN,R1      ;POINT TO END OF OUTPUT BUFFER (DESTINATION)
3582           ;DO WHILE NO NULL FOUND IN SOURCE STRING
3583 072432 112301      1$:      MOVB      (R3)+,R1      ;GET BYTE IN QUESTION (& POINT TO NEXT BYTE)
3584 072434 142701 000200      2$:      BICB      @200,R1      ;ENSURE HI BIT IS LO (SEVEN BIT ASCII)
3585           ;CHECK FOR VALID HEX CHARACTER
3586 072440 001414      BEQ      4$      ;THE NULL IS A VALID CHARACTER (BUT THE LAST)
3587 072442 120127 000060      CMPB      R1,@60      ;IF GREATER THAN "0" THEN RANGE OK
3588 072446 100413      CMI      6$      ;6$-OUT OF RANGE, BYTE VALUE TOO SMALL
3589 072450 120127 000072      CMPB      R1,@72      ;IF BYTE IS LESS THAN 72 AND >=60 THEN RANGE OK
3590 072454 100406      BMI      4$      ;RANGE OK IF >=60 AND <72 ELSE, CONTINUE CHECK
3591 072456 120127 000101      CMPB      R1,@101     ;BYTE MUST BE >=101 TO CONTINUE CHECK
3592 072462 100405      BMI      6$      ;IF BYTE >71 AND <101 THEN BYTE OUT OF RANGE.
3593 072464 122701 000106      CMPB      @106,R1     ;BYTE MUST BE <=106 TO BE OK, ELSE: NOT OK.
3594 072470 100402      BMI      6$      ;BYTE NOT OK, DON'T PLACE THIS BYTE IN OUTPUT.
3595           ;PLACE THE HEX BYTE IN THE OUTPUT BUFFER.
3596 072472 110122      4$:      MOVB      R1,(R2)+      ;BYTE IN RANGE, CONFIRMED BYTE OK. POINT TO
3597 072474 001405      BEQ      9$      ;NEXT BYTE DEST ADDRESS. IF NULL, THEN EXIT.
3598           ;(NO ERRORS)
3599           ;IF NOT NULL, THEN CHECK FOR BUFFER OVERFLOW.
3600           ;R4 POINTS TO LAST CHARACTER POSITION (OUTPUT)
3601 072476 020402      6$:      CMP      R4,R2      ;R2 - PRESENT WRITE ADDRESS
3602 072500 100354      BPL      1$      ;(SHOULD BE POSITIVE RESULT OR 0) (MORE TO DO)
3603 072502 012704 17777?      7$:      MOV      @-1,R4      ;SET ERROR CONDITION (EXIT WITH ERROR)
3604 072506 000404      BR      11$      ;ERROR DETECTED EXIT PATH -> (TOO MANY CHARS)
3605
3606           ;SUCCESSFUL EXIT PATH

```

G9

3607 072510 005302
3608 072512 020402
3609 072514 001372
3610 072516 005004
3611 072520
3612
3613

94: DEC R2
CMP R4,R2
BNE 74
CLR R4
114: RETURN R2,R4

;POINT TO THE LAST ACTUAL CHARACTER AT DEST BFR
;CHECK FOR MINIMUM OF 12 CHARACTERS.
;BRANCH IF LESS THAN 12, ERROR.
;INDICATE SUCCESS
;ADDRESS OF LAST CHARACTER (R2) IS P4
;ERROR INDICATOR (R4) IS P5

```

3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637 072526 000000
3638 072530 060 061 062
      072533 063 064 065
      072536 066 067 070
      072541 071 101 102
      072544 103 104 105
      072547 106 000
3639
3640
3641 072552
3642
3643
3644
3645 072562 060237 072526
3646
3647 072566 012704 072530
3648 072572 121124
3649 072574 001376
3650 072576 005201
3651 072600 162704 072531
3652
3653
3654 072604 006304
3655 072606 006304
3656 072610 006304
3657 072612 006304
3658 072614 010403
3659
3660 072616 012704 072530
3661 072622 121124
3662 072624 001376
3663 072626 005201
3664 072630 162704 072531
3665
3666
3667 072634 050403
3668
3669 072636 110322

```

```

      ; ---
      ;
      ; HEXBIN
      ;
      ; HEX TO BINARY CONVERSION PROCEDURE
      ;
      ; THIS PROCEDURE WILL CONVERT A STRING OF HEX (ASCII) CHARACTERS
      ; DIRECTLY TO A BINARY STREAM. THE DESTINATION BINARY STREAM WILL
      ; REQUIRE ONLY HALF AS MANY BYTES AS THE HEX STRING BECAUSE TWO
      ; HEX CHARACTERS ARE REQUIRED TO REPRESENT A SINGLE BINARY BYTE.
      ;
      ; INPUTS
      ; P1 - SOURCE STRING ADDRESS (DELIMITED BY A NULL)
      ; P2 - DESTINATION ADDRESS FOR THE BINARY DATA.
      ; P3 - THE NUMBER OF BINARY BYTES REQUIRED (HALF THE
      ; NUMBER OF CHARACTERS AT P1.
      ;
      ; OUTPUTS
      ; NO EXPLICIT OUTPUTS
      ; IMPLIED OUTPUTS
      ; THE BUFFER AT P2 WILL CONTAIN THE BINARY
      ; STREAM, CONVERTED DIRECTLY FROM THE BUFFER
      ; AT P1.
      ;
      ; SUBORDINATE PROCEDURES
      ; NONE
      ; CALLING PROCEDURE
      ; CALL HEXBIN P1,P2,P3
      ; ---
      ;
      ; .WORD
      ; CMPSTR: .ASCIZ /0123456789ABCDEF/
      ;
      ; .EVEN
      ;
      ; HEXBIN: P$POP R1,R2,HN
      ; R1=SOURCE STRING ADDRESS
      ; R2=DESTINATION STRING ADDRESS
      ; HN=NUMBER OF BYTES REQUIRED
      ;
      ; ADD R2,HN
      ; HN NOW POINTS TO THE LAST_BYTE_POSITION+1
      ;
      1$: MOV #CMPSTR,R4
      2$: CMPB (R1),(R4)+
      ; POINTER IN THE COMPARE STRING
      ; COMPARE CURRENT CHAR WITH A CHAR IN CMPSTR
      ; REPEAT UNTIL CHARACTER FOUND IN LIST
      ; POINT TO THE NEXT ASCII BYTE
      ; R4 NOW CONTAINS THE ACTUAL BINARY VALUE FOR
      ; THE NIBBLE DESCRIBED BY THE CURRENT BYTE.
      ; NOTE: NIBBLE IS THE HI PORTION OF THE BYTE
      ; MOVE NIBBLE TO THE HI END OF THE BYTE
      ;
      ; ASL R4
      ; ASL R4
      ; ASL R4
      ; ASL R4
      ; MOV R4,R3
      ; SAVE THE HI NIBBLE
      ;
      3$: MOV #CMPSTR,R4
      ; POINTER INTO COMPARE STRING
      ; COMPARE CURRENT CHAR WITH A CHAR IN CMPSTR
      ; REPEAT UNTIL MATCH FOUND IN CMPSTR LIST
      ; POINT TO THE NEXT ASCII BYTE
      ; R4 NOW CONTAINS THE ACTUAL BINARY VALUE FOR
      ; THE NIBBLE DESCRIBED BY THE CURRENT BYTE.
      ; NOTE: NIBBLE IS THE HI PORTION OF THE BYTE
      ; NOW THE TWO CHARACTERS HAVE MADE A SINGLE BYTE
      ; NOW PLACE THE COMPLETE BYTE IN THE DESTINATION
      ; AND POINT TO THE NEXT DESTINATION BYTE
      ;
      ; CMPB (R1),(R4)+
      ; BNE 3$
      ; INC R1
      ; SUB #CMPSTR+1,R4
      ;
      ; BIS R4,R3
      ;
      ; MOVB R3,(R2)+

```

3670 072640 020237 072526
3671 072644 100750
3672 072646
3673
3674
3675

CMP R2,HN
BMI 1#
RETURN

;IF THE DESTINATION POINTER (R2) REACHES THE
;LAST CHARACTER POSITION+1 (HN) THEN DONE.
;RETURN TO CALLER


```

3680
3681
3682          |---+
3683          |          BINHEX                      BINARY TO HEX CONVERSION PROCEDURE
3684          |          THIS PROCEDURE WILL CONVERT A BINARY DATA STREAM INTO A HEX STRING.
3685          |
3686          |          INPUTS  -  P1- BINARY DATA BUFFER ADDRESS
3687          |                   P2- NUMBER OF BYTES IN THE BUFFER
3688          |                   P3- ADDRESS OF OUTPUT BUFFER FOR HEX STRING.
3689          |                   HEX CHARACTER PAIRS SEPERATED BY "-" 'S
3690          |                   (NOTE: THIS BUFFER MUST BE AT LEAST 3*P2 BYTES LONG)
3691          |
3692          |          OUTPUTS - NONE
3693          |          IMPLICIT OUTPUTS  THE BUFFER AT P3 WILL CONTAIN THE HEX STRING
3694          |                               FOLLOWED BY A NULL CHARACTER.
3695          |          SUBORDINATE ROUTINES NONE
3696          |          CALLING PROCEDURE  CALL BINHEX P1,P2,P3
3697          |---+
3697 072650      060      061      062  HEXC:  .ASCII /0123456789ABCDEF/
3698 072653      063      064      065
3699 072656      066      067      070
3700 072661      071      101      102
3701 072664      103      104      105
3702 072667      106
3703 072670 000000
3704
3705 LST:      .WORD
3706
3707 BINHEX: P#POP  R1,LST,R2          ;R1 HAS THE INPUT BUFFER ADDRESS
3708                                     ;LST: HAS THE NUMBER OF BYTES IN INPUT BUFFER
3709                                     ;R2 HAS THE OUTPUT BUFFER ADDRESS
3710                                     ;LST IS NOW ADDRESS OF LAST SOURCE BYTE + 1
3711 1#:      ADD      R1,LST          ;GET THE CURRENT BYTE AND POINT TO NEXT BYTE
3712           MOVB   (R1)+,R3        ;SEPARATE NIBBLES AND GET CHARACTERS SEPARATELY
3713           MOVB   R3,R4           ;ONLY RIGHT BINARY NIBBLE REMAINS IN R3
3714           BIC   @177760,R3      ;SHIFT OVER FOR LEFT BINARY NIBBLE IN R4
3715           ASR   R4
3716           ASR   R4
3717           ASR   R4
3718           ASR   R4
3719           BIC   @177760,R4      ;ONLY LEFT BINARY NIBBLE REMAINS IN R4
3720                                     ;R4 IS THE MOST SIGNIFICANT NIBBLE (FIRST)
3721                                     ;R3 IS THE LEAST SIGNIFICANT NIBBLE (SECOND)
3722           MOVB   HEXC(R4),(R2)+  ;PUT THE ASCII BYTE INTO THE BUFFER HI POSITION
3723           MOVB   HEXC(R3),(R2)+  ;PUT THE ASCII BYTE INTO THE BUFFER LO POSITION
3724           MOVB   @'-,(R2)+      ;PUT - BETWEEN HEX PAIRS
3725           CMP   R1,LST          ;RESULT IS NEGATIVE UNTIL R1=LST
3726           BLO  1#              ;UNTIL R1=LST. (TRANSFER ALL SOURCE BYTES)
3727           CLRB  -(R2)          ;TERMINATE OUTPUT BUFFER WITH A NULL
3728           RETURN
3729
3730 .SBTTL BLDLD  BUILD LOOP DIRECT DATA BUFFERS FOR TRANSMIT.
3731
3732 |---+
3733 | FUNCTIONAL DESCRIPTION:
3734 | THIS SUBROUTINE BUILDS LOOP DIRECT PACKETS FOR TRANSMISSION
3735 | FROM THE UNA, SOURCE ADDRESS, DESTINATION ADDRESS,
3736 | PROT. TYPE, AND LOOP DIRECT HEADER INFO ARE ADDED
3737 | TO THE MESSAGE BUFFER. THE MESSAGE BUFFER IS BUILT
3738 | BY A CALL TO BLDBUF.

```

```

3732 ;
3733 ; INPUTS - P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
3734 ; IMPLICIT - P#SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
3735 ; XRGNXT POINTS TO THE NEXT AVAILABLE RING ENTRY
3736 ; PHYADR HOLDS THE CURRENT LOCAL UNA PHYSICAL ADDRESS
3737 ;
3738 ; OUTPUTS - BUILT MESSAGE PACKET.
3739 ;
3740 ; CALLING PROCEDURE - CALL BLDLD P1
3741 ;
3742 ; SIDE EFFECTS - THE MESSAGE PACKET IS BUILD AND CONTAINED IN THE
3743 ; BUFFER POINTED TO BY XRGNXT WHEN THE ROUTINE WAS ENTERED.
3744 ; XRGNXT IS UPDATED TO POINT TO THE NEXT RING ENTRY
3745 ;
3746 ; REGISTER USAGE - R1 HOLDS ADDRESS OF DESTINATION ADDRESS
3747 ; R2 IS A POINTER FOR THE LOOP DIRECT HEADER INFO
3748 ; R3 HOLDS THE PACKET LENGTH
3749 ; R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
3750 ;
3751 ; ---
3752 ;
3753 072760 BLDLD::
3754 072760 P#POP R1 ; PUT ADDRESS OF DEST. ADDRESS IN R1
3755 072762 013704 003762 MOV XRGNXT,R4 ; MOVE NEXT PACKET ADDRESS TO R4
3756 072766 032764 100000 000004 BIT @OWN,4(R4) ; CHECK OWNERSHIP BIT
3757 072774 001110 BNE 40$ ; IF DON'T OWN, BOOKKEEPING ERROR,
3758 072776 016404 000002 MOV 2(R4),R4 ; POINT R4 TO DATA BLOCK
3759 073002 013703 002372 MOV P#SIZE,R3 ; PUT MESSAGE SIZE INTO R3
3760 073006 062703 000040 ADD @40,R3 ; ADD HEADER AND LOOP DIRECT INFO TO LENGTH
3761 073012 010337 050560 MOV R3,XFER ; PUT BYTES TRANSFERED INTO WORD
3762 073016 162737 000016 050560 SUB @16,XFER ; AND CORRECT FOR HEADER
3763 073024 022703 002756 CMP @XPKLEN,R3 ; SEE IF LONGER THAN ONE PACKET
3764 073030 002477 BLT 45$ ; IF YES, ERROR
3765 073032 010337 050566 MOV R3,BUFLEN ; PUT PACKET LENGTH IN BUFLEN
3766 073036 012164 000000 MOV (R1)+,DESTIN(R4) ; MOVE FIRST TWO BYTES OF ADDRESS
3767 073042 012164 000002 MOV (R1)+,DESTIN+2(R4) ; MOVE BYTES THREE AND FOUR
3768 073046 011164 000004 MOV (R1),DESTIN+4(R4) ; MOVE BYTES FIVE AND SIX
3769 073052 005064 000006 CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
3770 073056 005064 000010 CLR SOURCC+2(R4) ; SIX BYTES WORTH
3771 073062 005064 000012 CLR SOURCC+4(R4)
3772 073066 013764 050544 000014 MOV PROT00,PROT0T(R4) ; MOVE PROCALL TYPE INTO HEADER
3773 073074 012702 050720 MOV @LOPDIR,R2 ; MOVE LOOPDIRECT FORMAT HEADER LOC. TO R2
3774 073100 012264 000016 MOV (R2)+,LDSKIP(R4) ; SKIP COUNT
3775 073104 011264 000020 MOV (R2),LDFCT1(R4) ; FUNCTION CODE (FORWARD)
3776 073110 013764 017762 000022 MOV PHYADR,LDADR1(R4) ; LOCAL NODE ADDRESS
3777 073116 013764 047764 000024 MOV PHYADR+2,LDADR1+2(R4) ; SIX BYTES
3778 073124 013764 047766 000026 MOV PHYADR+4,LDADR1+4(R4)
3779 073132 016264 000010 000030 MOV 10(R2),LDFCT2(R4) ; FUNCTION CODE (REPLY)
3780 073140 013764 017762 000032 MOV PHYADR,LDADR2(R4) ; LOCAL NODE ADDRESS
3781 073146 013764 047764 000034 MOV PHYADR+2,LDADR2+2(R4) ; SIX BYTES
3782 073154 013764 047766 000036 MOV PHYADR+4,LDADR2+4(R4)
3783 073162 062704 000040 ADD @LDADR2+6,R4 ; POINT R4 TO FIRST DATA BYTE
3784 073166 010437 050570 MOV R4,CMPBUF ; STORE BUFFER LOCATION FOR DATA COMPARE
3785 073172 CALL BLDBUF R4 ; BUILD DATA BUFFER
3786 073202 CALL GETXNX @XRGNXT ; UPDATE POINTER TO NEXT RING ENTRY
3787 073214 000411 BR 60$ ; EXIT
3788 073216 40$: ERRDF 13,MSG10,ERR1 ; TRANSMIT RING BOOKKEEPING ERROR

```

L9

073216 104455
073220 000015
073222 064656
073224 067460
3789 073226 000404
3790 073230
073230 104455
073232 000016
073234 064755
073236 067460
3791 073240
3792

BR 60:
ERRDF 14,EMSG14,ERR1
45:
60:
RETURN

; EXIT
; MESSAGE SIZE TOO BIG

TRAP C\$ERDF
.WORD 13
.WORD EMSG10
.WORD ERR1
TRAP C\$ERDF
.WORD 14
.WORD EMSG14
.WORD ERR1

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

.SBTTL BLDFAS BUILD PACKET FOR FULL ASSIST TRANSMISSION.

```

:--+
: FUNCTIONAL DESCRIPTION:
:   THIS SUBROUTINE BUILDS FULL ASSIST PACKETS FOR TRANSMISSION
:   FROM THE UNA. SOURCE ADDRESS, DESTINATION ADDRESS, PROT. TYPE
:   AND FULL ASSIST HEADER INFO ARE ADDED TO THE MESSAGE BUFFER.
:   THE MESSAGE BUFFER IS BUILT BY A CALL TO BLDBUF.
:
: INPUTS -      P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
:               IMPLICIT - P#SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
:                   XRG NXT POINTS TO THE NEXT AVAILABLE RING ENTRY
:                   PHYADR HOLDS THE CURRENT LOCAL NODE ADDRESS
:
: OUTPUTS -     THE BUILT BUFFER
:
: CALLING PROCEDURE - CALL BLDFAS P1
:
: SIDE EFFECTS - XRG NXT SI UPDATED TO POINT TO THE NEXT RING ENTRY
:
: REGISTER USAGE - R1 HOLDS ADDRESS OF TARGET NODE ADDRESS
:                  R2 HOLDS ADDRESS OF ASSIST NODE ADDRESS
:                  R3 HOLDS THE PACKET LENGTH
:                  R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
:--+

```

3821 073242
3822 073242
3823
3824 073246 013704 003762
3825 073252 032764 100000 000004
3826 073260 001140
3827 073262 016404 000002
3828 073266 013703 002372
3829 073272 062703 000060
3830 073276 010337 050560
3831 073302 162737 000016 050560
3832 073310 022703 002756
3833 073314 002527
3834 073316 010337 050566
3835 073322 011264 000000
3836 073326 016264 000002 000002
3837 073334 016264 000004 000004
3838 073342 005064 000006
3839 073346 005064 000010
3840 073352 005064 000012
3841 073356 013764 050544 000014
3842 073364 012764 000000 000016
3843 073372 012764 000002 000020
3844 073400 011164 000022
3845 073404 016164 000002 000024
3846 073412 016164 000004 000026
3847 073420 012764 000002 000030
3848 073426 011264 000032
3849 073432 016264 000002 000034
3850 073440 016264 000004 000036

```

BLDFAS:
      P#POP      R1,R2
      MOV        XRG NXT,R4
      BIT        #OWN,4(R4)
      BNE        40$
      MOV        2(R4),R4
      MOV        P#SIZE,R3
      ADD        #60,R3
      MOV        R3,XFER
      SUB        #16,XFER
      CMP        #XP KLEN,R3
      BLT        45$
      MOV        R3,BUFLEN
      MOV        (R2),DESTIN(R4)
      MOV        2(R2),DESTIN+2(R4)
      MOV        4(R2),DESTIN+4(R4)
      CLR        SOURCC(R4)
      CLR        SOURCC+2(R4)
      CLR        SOURCC+4(R4)
      MOV        PROT00,PROTOT(R4)
      MOV        #0,FASKIP(R4)
      MOV        #2,FAFCT1(R4)
      MOV        (R1),FAADR1(R4)
      MOV        2(R1),FAADR1+2(R4)
      MOV        4(R1),FAADR1+4(R4)
      MOV        #2,FAFCT2(R4)
      MOV        (R2),FAADR2(R4)
      MOV        2(R2),FAADR2+2(R4)
      MOV        4(R2),FAADR2+4(R4)
      ; PUT ADDRESS OF TARGET ADDRESS INTO R1
      ; AND ADDRESS OF ASSIST ADDRESS INTO R2
      ; MOVE NEXT PACKET ADDRESS TO R4
      ; CHECK OWNERSHIP BIT
      ; IF DON'T OWN, BOOKKEEPING ERROR,
      ; POINT R4 TO DATA BLOCK
      ; PUT MESSAGE SIZE INTO R3
      ; ADD HEADER INFO TO LENGTH
      ; PUT 'BYTES TRANSFERED' INTO WORD
      ; AND CORRECT FOR HEADER
      ; SEE IF LONGER THAN ONE PACKET
      ; IF YES, ERROR
      ; PUT PACKET LENGTH IN BUFLN
      ; MOVE FIRST TWO BYTES OF ADDRESS
      ; MOVE BYTES THREE AND FOUR
      ; MOVE BYTES FIVE AND SIX
      ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
      ; SIX BYTES WORTH
      ; MOVE PROTOCOL TYPE INTO HEADER
      ; SKIP COUNT
      ; FUNCTION CODE (FORWARD)
      ; TARGET NODE ADDRESS
      ; SIX BYTES
      ; FUNCTION CODE (FORWARD)
      ; ASSIST NODE ADDRESS
      ; SIX BYTES

```

3851	073446	012764	000002	000040	MOV	#2,FAFCT3(R4)	;	FUNCTION CODE (FORWARD)
3852	073454	013764	047762	000042	MOV	PHYADR,FAADR3(R4)	;	LOCAL NODE ADDRESS
3853	073462	013764	047764	000044	MOV	PHYADR+2,FAADR3+2(R4)	;	SIX BYTES
3854	073470	013764	047766	000046	MOV	PHYADR+4,FAADR3+4(R4)	;	
3855	073476	012764	000001	000050	MOV	#1,FAFCT4(R4)	;	FUNCTION CODE (REPLY)
3856	073504	013764	047762	000052	MOV	PHYADR,FAADR4(R4)	;	LOCAL NODE ADDRESS
3857	073512	013764	047764	000054	MOV	PHYADR+2,FAADR4+2(R4)	;	SIX BYTES
3858	073520	013764	047766	000056	MOV	PHYADR+4,FAADR4+4(R4)	;	
3859	073526	062704	000060		ADD	#FAADR4+6,R4	;	POINT R4 TO FIRST DATA BYTE
3860	073532	010437	050570		MOV	R4,CMPBUF	;	STORE BUFFER LOCATION FOR DATA COMPARE
3861	073536				CALL	BLDBUF R4	;	BUILD DATA BUFFER
3862	073546				CALL	GETXNX #XRG NXT	;	UPDATE POINTER TO NEXT RING ENTRY
3863	073560	000411			BR	50\$;	EXIT
3864	073562			40\$:	ERRDF	28,MSG10,ERR1	;	TRANSMIT RING BOOKKEEPING ERROR
	073562	104455						TRAP C\$ERDF
	073564	000034						.WORD 28
	073566	064656						.WORD MSG10
	073570	067460						.WORD ERR1
3865	073572	000404			BR	50\$;	EXIT
3866	073574			45\$:	ERRDF	29,MSG14,ERR1	;	MESSAGE SIZE TOO BIG
	073574	104455						TRAP C\$ERDF
	073576	000035						.WORD 29
	073600	064755						.WORD EMSC14
	073602	067450						.WORD ERR1
3867	073604			50\$:	RETURN			
3868								

```

3870          .SBTTL BLDAST BUILD TRANSMIT AND RECEIVE ASSIST PACKETS
3871
3872
3873 073606          BLDAST:
3874 073606          P#POP      R1,R2          ; PUT DESTINATION ADDRESS INTO R1
3875                                     ; ASSIST ADDRESS INTO R2
3876 073612 013704 003762          MOV      XRGXNT,R4          ; MOVE NEXT PACKET ADDRESS TO R4
3877 073616 032764 100000 000004          BIT      @JWN,4(R4)          ; CHECK OWNERSHIP BIT
3878 073624 001125                                     BNE      40$          ; IF DON'T OWN, BOOKKEEPING ERROR
3879 073626 016404 000002          MOV      2(R4),R4          ; POINT R4 TO DATA BLOCK
3880 073632 013703 002372          MOV      P#SIZE,R3          ; PUT MESSAGE SIZE INTO R3
3881 073636 062703 000050          ADD      #50,R3          ; ADD HEADER INFO INTO LENGTH
3882 073642 010337 050560          MOV      R3,XFER          ; PUT 'BYTES TRANSFERED' INTO WORD
3883 073646 162737 000016 050560          SUB      #16,XFER          ; AND CORRECT FOR HEADER
3884 073654 022703 002756          CMP      @XPklen,R3          ; SEE IF LONGER THAN ONE PACKET
3885 073660 002514                                     BLT      45$          ; IF YES, ERROR
3886 073662 010337 050566          MOV      R3,BUFLEN          ; PUT PACKET LENGTH INTO BUFLen
3887 073666 011164 000000          MOV      (R1),DESTIN(R4)          ; MOVE DESTINATION ADDRESS INTO HEADER
3888 073672 016164 000002 000002          MOV      2(R1),DESTIN+2(R4)          ; SIX BYTES WORTH
3889 073700 016164 000004 000004          MOV      4(R1),DESTIN+4(R4)
3890 073706 005064 000006          CLR      SOURCC(R4)          ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
3891 073712 005064 C00010          CLR      SOURCC+2(R4)          ; SIX BYTES WORTH
3892 073716 005064 000012          CLR      SOURCC+4(R4)
3893 073722 013764 050544 000014          MOV      PROTOO,PROTOT(R4)          ; MOVE PROTOCOL TYPE INTO HEADER
3894 073730 012764 000000 000016          MOV      #0,FASKIP(R4)          ; SKIP COUNT
3895 073736 012764 000002 000020          MOV      #2,FAFCT1(R4)          ; FUNCTION CODE (FORWARD)
3896 073744 011264 000022          MOV      (R2),FAADR1(R4)          ; TARGET NODE ADDRESS
3897 073750 016264 000002 000024          MOV      2(R2),FAADR1+2(R4)          ; SIX BYTES
3898 073756 016264 000004 000026          MOV      4(R2),FAADR1+4(R4)
3899 073764 012764 000002 000030          MOV      #2,FAFCT2(R4)          ; FUNCTION CODE (FORWARD)
3900 073772 013764 047762 000032          MOV      PHYADR,FAADR2(R4)          ; LOCAL NODE ADDRESS
3901 074000 013764 047764 000034          MOV      PHYADR+2,FAADR2+2(R4)          ; SIX BYTES WORTH
3902 074006 013764 047766 000036          MOV      PHYADR+4,FAADR2+4(R4)
3903 074014 012764 000001 000040          MOV      #1,FAFCT3(R4)          ; FUNCTION CODE (REPLY)
3904 074022 013764 047762 000042          MOV      PHYADR,FAADR3(R4)          ; LOCAL NODE ADDRESS
3905 074030 013764 047764 000044          MOV      PHYADR+2,FAADR3+2(R4)
3906 074036 013764 047766 000046          MOV      PHYADR+4,FAADR3+4(R4)
3907 074044 062704 000050          ADD      @FAADR3+6,R4          ; POINT R4 TO FIRST DATA BYTE
3908 074050 010437 050570          MOV      R4,CMPBUF          ; STORE BUFFER LOCATION FOR DATA COMPARE
3909 074054          CALL      BLDBUF P4          ; BUILD DATA BUFFER
3910 074064          CALL      GETXNX @XRGXNT          ; UPDATE RING POINTER
3911 074076 000411          BR      50$
3912 074100          40$: ERRDF      35,MSG10,ERR1          ; TRANSMIT RING BOOKKEEPING ERROR
3913 074100 104455          TRAP      C$ERDF
3914 074102 000043          .WORD      35
3915 074104 064655          .WORD      MSG10
3916 074106 067460          .WORD      ERR1
3917 074110 000404          BR      50$
3918 074112          45$: ERRDF      36,MSG14,ERR1          ; MESSAGE SIZE TOO BIG ERROR
3919 074112 104455          TRAP      C$ERDF
3920 074114 000044          .WORD      36
3921 074116 064755          .WORD      MSG14
3922 074120 067460          .WORD      ERR1
3923 074122          50$: RETURN
3924
3925
3926
3927

```

3919
 3920
 3921
 3922
 3923
 3924
 3925
 3925
 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

.SBTTL BLDREQ BUILD REQUEST ID PACKETS FOR TRANSMIT.

```

---+
| FUNCTIONAL DESCRIPTION:
| THIS SUBROUTINE BUILDS REQUEST ID PACKETS FOR TRANSMISSION
| FROM THE UNA, SOURCE ADDRESS, DESTINATION ADDRESS,
| PROTOCOL TYPE, SEQUENCE NUMBER AND REQUEST ID
| HEADER INFO ARE BUILT IN THIS SUBROUTINE.
|
| INPUTS - IMPLICIT - THE DESTINATION ADDRESS IS CONTAINED IN ADRBUF.
|
| OUTPUTS - BUILT MESSAGE PACKET
|
| CALLING PROCEDURE - CALL BLDREQ
|
| SIDE EFFECTS - THE MESSAGE PACKET IS BUILT AND CONTAINED IN THE
| BUFFER POINTED TO BY XRGXNT WHEN THE ROUTINE WAS
| ENTERED. XRGXNT IS UPDATED TO POINT TO THE NEXT ENTRY.
|
| REGISTER USAGE - R1 HOLDS ADDRESS OF DESTINATION ADDRESS.
| R2 IS A POINTER FOR REQUEST ID HEADER INFO.
| R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER.
|
---+
  
```

```

BLDREQ:
MOV XRGXNT,R4 ; MOVE NEXT PACKET ADDRESS TO R4
BIT @OWN,4(R4) ; CHECK OWNERSHIP BIT
BNE 40: ; IF DON'T OWN, BOOKKEEPING ERROR
MOV 2(R4),R4 ; POINT R4 TO DATA BLOCK
MOV @100,BUFLEN ; MOVE BUFFER SIZE TO BUFLN
MOV @ADRBUF,R1 ; MOVE ADDRESS OF DEST. ADR. TO R1
MOV (R1),DESTIN(R4) ; MOVE FIRST TWO BYTE OF DEST. ADR.
MOV (R1),DESTIN+2(R4) ; AND BYTES THREE AND FOUR
MOV (R1),DESTIN+4(R4) ; AND LAST TWO BYTES
CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDR.
CLR SOURCC+2(R4) ; SIX BYTES WORTH
CLR SOURCC+4(R4)
MOV PRO102,PROTOT(R4) ; MOVE PROTOCOL TYPE INTO HEADER
MOV @REQID,R2 ; MOVE REQUEST ID HEADER LOC. TO R2
MOV (R2),HEADER(R4) ; BYTE COUNT
MOV (R2),HEADER+2(R4) ; FUNCTION CODE (REQUEST ID)
MOV (R2),HEADER+4(R4) ; RECEIPT NO.
CALL GETXNX @XRGXNT ; UPDATE POINTER TO NEXT RING ENTRY
BR 50: ; EXIT
40: ERDF 21,MSG10,ERR1 ; TRANSMIT RING BOOKKEEPING ERROR
TRAP C:ERDF
.WORD 21
.WORD MSG10
.WORD ERR1
50: RETURN
  
```

.SBTTL GETXNX GET NEXT TRANSMIT OR RECIEVE RING ENTRY

```

---+
| FUNCTIONAL DESCRIPTION
  
```

D10

```

3972          ;           THIS SUBROUTINE GETS THE NEXT TRANSMIT OR RECIEVE RING
3973          ;           ENTRY.  IT IS ENTERED AT SEPERATE POINTS DEPENDING ON
3974          ;           WHICH RING IS BEING USED.
3975          ;
3976          ; INPUTS -   P1 - THE ADDRESS OF THE RING POINTER TO BE UPDATED.
3977          ;
3978          ; OUTPUTS -  THE RING POINTER IS UPDATED TO POINT TO THE NEXT AVAILABLE
3979          ;           ENTRY.
3980          ;
3981          ; CALLING PROCEDURE - CALL GETXNX @P1           ; FOR TRANSMIT UPDATES
3982          ;                               CALL GETRXN @P1          ; FOR RECIEVE UPDATES
3983          ;
3984          ; SIDE EFFECTS - NONE
3985          ;
3986          ; REGISTER USAGE - R1 POINTS TO THE FIRST ENTRY IN THE RING
3987          ;                               R2 POINTS TO THE LAST ENTRY IN THE RING
3988          ;                               R3 IS THE ADDRESS OF THE RING POINTER TO BE UPDATED
3989          ;
3990          ; ---+
3991          ;
3992          ; GETRXN::
3993          ; MOV     RRGSR1,R1           ; MOVE FIRST RING ENTRY TO R1
3994          ; MOV     RRGLST,R2          ; MOVE LAST RING ENTRY TO R2
3995          ; BR      GETCOM             ; GO TO COMMON CODE
3996          ;
3997          ; GETXNX::
3998          ; MOV     XRGSRT,R1           ; MOVE FIRST RING ENTRY TO R1
3999          ; MOV     XRGLST,R2          ; MOVE LAST RING ENTRY TO R2
4000          ; P#POP  R3                 ; GET ADDRESS OF RING POINTER IN R3
4001          ; CMP     (R3),R2            ; SEE IF POINTER POINTS TO LAST RING
4002          ; BEQ     15$                ; IF YES, BRANCH
4003          ; ADD     @10.,(R3)         ; ELSE, ADD ENTRY LENGTH TO POINTER
4004          ; BR      25$                ; EXIT
4005          ; MOV     R1,(R3)           ; POINT POINTER TO FIRST ENTRY IN RING
4006          ; 15$:   MOV
4007          ; 25$:   RETURN
4008          ;
4009          ;
4010          ; .SBTTL  BLDBUF  BUILD MESSAGE BUFFERS
4011          ;
4012          ; ---+
4013          ; FUNCTIONAL DESCRIPTION
4014          ;           THIS SUBROUTINE CREATES A MESSAGE BUFFER TO BE USED
4015          ;           FOR TRANSMISSION.
4016          ;
4017          ; INPUTS -   P1 - BUFFER LOCATION TO PUT BUILT MESSAGE
4018          ;           IMPLICIT - P#SIZE CONTAINS THE SIZE THE BUFFER IS TO BE
4019          ;           P#TYPE CONTAINS THE MESSAGE TYPE
4020          ; OUTPUTS - IMPLICIT - BUFFER STARTING AT LOCATION P1 CONTAINS A
4021          ;           MESSAGE P#SIZE BYTES LONG USING THE MESSAGE
4022          ;           TYPE SPECIFIED BY P#TYPE.
4023          ;
4024          ; CALLING PROCEDURE - CALL BLDBUF P1
4025          ;
4026          ; SIDE EFFECTS - NONE
4027          ;
4028          ; REGISTER USAGE - R1 POINTS TO THE NEXT BYTE OF STORED MESSAGE TO BE BUILT

```


E10

```

4029      ;
4030      ;
4031      ;
4032      ;
4033      ;
4034      ;
4035      ;
4036 074324      ;
4037 074324      ;
4038 074326 013702 002370      ;
4039 074332 006302      ;
4040 074334 013704 002372      ;
4041 074340 060304      ;
4042 074342 016201 003316      ;
4043 074346 005037 050542      ;
4044 074352 005237 050542      ;
4045 074356 112123      ;
4046 074360 026237 003300 050542      ;
4047 074366 001004      ;
4048 074370 016201 003316      ;
4049 074374 005037 050542      ;
4050 074400 020304      ;
4051 074402 001363      ;
4052 074404      ;
4053      ;

      IN THE MESSAGE BUFFER.
      R2 = (MESSAGE TYPE X 2), USED AS OFFSET FOR POINTERS
      R3 POINTS TO THE NEXT BYTE OF THE BUFFER UNDER CONSTRUCTION
      R4 POINTS TO THE LAST BYTE OF THE BUFFER UNDER CONSTRUCTION

BLDBUF::
      P#POP      R3      ; PUT BUFFER ADDRESS INTO R3
      MOV      P#TYPE,R2      ; PUT MESSAGE TYPE INTO R2
      ASL      R2      ; MULTIPLY BY 2
      MOV      P#SIZE,R4      ; PUT SIZE INTO R4
      ADD      R3,R4      ; MAKE R4 = LAST BYTE OF BUFFER
      MOV      MSGAD(R2),R1      ; POINT R1 TO FIRST BYTE OF STORED MESSAGE
      CLR      COUNT      ; CLEAR BYTE COUNTER
10$:      INC      COUNT      ; COUNT NO. OF BYTES COPIED
      MOVB     (R1)+,(R3)+      ; PUT BYTE IN BUFFER
      CMP      MSGCNT(R2),COUNT      ; ARE WE AT END OF STORED MESSAGE
      BNE      20$      ; IF NO, CHECK IF DONE
      MOV      MSGAD(R2),R1      ; ELSE, POINT R1 TO BEGINING
      CLR      COUNT      ; AND CLEAR COUNTER
20$:      CMP      R3,R4      ; IS BUFFER FILLED?
      BNE      10$      ; IF NO, LOOP
      RETURN      ; ELSE, RETURN

```

```

4055
4056      .SBTTL  DATCMP  COMPARE DATA BUFFERS
4057
4058      ;---+
4059      ; FUNCTIONAL DESCRIPTION
4060      ;           THIS SUBROUTINE COMPARES TWO DATA BUFFERS BYTE BY BYTE.
4061      ;           IF COMPARISON ERRORS OCCURED, LOCATION, EXPECTED DATA
4062      ;           AND RECIEVED DATA ARE PRINTED OUT FOR THE FIRST FIVE
4063      ;           ERRORS.  THE TOTAL NUMBER OF ERRORS IS ALSO PRINTED.
4064      ;
4065      ; INPUTS -   P1 - THE SIZE (IN BYTES) OF THE BUFFER TO BE COMPARED.
4066      ;           P2 - THE ADDRESS OF BUFFER TO COMPARE OTHER BUFFER AGAINST.
4067      ;           P3 - THE ADDRESS OF THE SECOND BUFFER.
4068      ;
4069      ; OUTPUTS -  P4 - THE NUMBER OF COMPARISON ERRORS.
4070      ;
4071      ; CALLING PROCEDURE - CALL DATCMP P1,P2,P3
4072      ;                       P#POP P4
4073      ;
4074      ; SIDE EFFECTS - NONE.
4075      ;
4076      ; REGISTER USAGE - R1 CONTAINS THE COMPARE SIZE
4077      ;                   R2 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 1
4078      ;                   R3 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 2
4079      ;                   R4 CONTAINS THE BYTE OFFSET (BYTES FROM BEGINING OF BUFFER
4080      ;
4081      ;---+
4082
4083      DATCMP::
4084      P#POP  R1,R2,R3      ; PUT COMPARE SIZE IN R1
4085      ;                   ; BUFFER 1 ADDRESS IN R2 AND
4086      ;                   ; BUFFER 2 ADDRESS IN R3
4087      074414 005004      CLR  R4      ; INITIALIZE BYTE OFFSET
4088      074416 005037 050550 CLR  TEMP    ; AND ERROR COUNTER
4089
4090      10$:  074422 005204      INC  R4      ; INCREMENT OFFSET COUNTER
4091      074424 121213      CMPB (R2),(R3) ; COMPARE BUFFERS
4092      074426 001421      BEQ  20$    ; IF SAME, BRANCH
4093      074430 005237 050550      INC  TEMP    ; INCREMENT ERROR COUNTER
4094      074434 022737 000005 050550 CMP  #5,TEMP ; IF MORE THAN 5 ERRORS,
4095      074442 002413      BLT  20$    ; DON'T PRINT MESSAGE
4096      074444      PFINTX #CMPE1,R4,(R2),(R3) ; PRINT ERROR MESSAGE
4097      074444 011346      MOV  (R3),-(SP)
4098      074446 011246      MOV  (R2),-(SP)
4099      074450 010446      MOV  R4, -(SP)
4100      074452 012746 066754      MOV  #CMPE1, -(SP)
4101      074456 012746 000004      MOV  #4, -(SP)
4102      074462 010600      MOV  SP,R0
4103      074464 104415      TRAP C#PNTX
4104      074466 062706 000012      ADD  #12,SP
4105
4106      20$:  074472 005202      INC  R2      ; INCREMENT BUFFER 1 POINTER
4107      074474 005203      INC  R3      ; INCREMENT BUFFER 2 POINTER
4108      074476 005301      DEC  R1      ; DECREMENT COMPARE SIZE
4109      074500 001350      BNE  10$    ; IF NOT FINISHED, GO BACK FOR MORE
4110      074502 022737 000000 050550 CMP  #0,TEMP ; WERE THERE ANY ERRORS?
4111      074510 001412      BEQ  30$    ; IF NO, EXIT
4112      074512      PRINTX #CMPE2,TEMP
    
```

```

074512 013746 050550
074516 012746 067051
074522 012746 000002
074526 010600
074530 104415
074532 062706 000006
4104 074536 30$: RETURN TEMP ; RETURN WITH ERROR COUNT ON STACK
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 074544
4139 074550 023727 050550 000001
4140 074556 001002
4141 074560
4142 074562 000402
4143 074564
4144 074570 012701 002656
4145 074574 005711
4146 074576 001415
4147 074600 021127 177777
4148 074604 001454
4149 074606
4150 074620
4151 074622 001412
4152 074624 062701 000026
4153 074630 000761
4154 074632 011211

MOV TEMP, -(SP)
MOV #CMPEP2, -(SP)
MOV #2, -(SP)
MOV SP, R0
TRAP C#PNTX
ADD #6, SP

; SBTTL WRITES WRITE DATA ONTO SUMMARY TABLE
;---+
; FUNCTIONAL DESCRIPTION:
; THIS SUBROUTINE UPDATES THE SUMMARY TABLE DATA FOR
; THE NODES SPECIFIED IN THE CALL STATEMENT. EITHER ONE
; OR TWO NODES CAN BE UPDATED PER CALL. AFTER THE CALL,
; THE SUMMARY DATA COUNTERS ARE CLEARED. THE SUMMARY TABLE
; IS CHECKED FOR A MATCHING NODE ADDRESS AND UPDATES THE
; DATE FOR THAT NODE, OR ADDS THE NODE TO THE TABLE IF IT
; DOESN'T EXIST. AN ERROR IS REPORTED IF THE END OF THE TABLE
; IS REACHED.
;
; INPUTS - P1 - THE NUMBER OF NODES TO UPDATE (1 OR 2).
; P2 - THE ADDRESS OF THE FIRST NODE ADDRESS.
; P3 - THE ADDRESS OF THE SECOND NODE ADDRESS IF P1 = 2 OR
; BLANK IF P1 = 1.
;
; OUTPUTS - THE SUMMARY TABLE IS UPDATED.
;
; CALLING PROCEDURE - CALL WRITES P1,P2(,P3)
;
; SIDE EFFECTS - THE SUMMARY COUNTERS ARE CLEARED.
;
; REGISTER USAGE - R1 POINTS TO THE CURRENT LOCATION IN THE SUMMARY TABLE.
; R2 POINTS TO THE NODE TO BE UPDATED'S ADDRESS.
; R3 IS SCRATCH
; R4 HOLDS THE SECOND NODE TO BE UPDATED ADDRESS.
;---+
WRITES: P#POP TEMP ; SEE HOW MANY NODES TO WRITE
CMP TEMP, #1 ; IF ONLY ONE, GET ADDRESS
BNE 5$
P#POP R2
BR 6$
5$: P#POP R2,R4 ; IF TWO, GET BOTH ADDRESSES
6$: MOV #STATBL,R1 ; MOVE STATISTICAL TABLE ADDRESS INTO R1
7$: TST (R1) ; SEE IF SLOT IS EMPTY
BEQ 15$ ; IF YES, BR
CMP (R1), #-1 ; SEE IF TABLE FULL
BEQ 25$ ; IF YES, ERROR
CALL CMPADR R1,R2 ; LOOK FOR MATCHING ADDRESS
P#POP R3
BEQ 20$ ; IF YES, BR
ADD #26,R1 ; ELSE, POINT R1 TO NEXT ENTRY
BR 7$ ; AND CHECK AGAIN
15$: MOV (R2),(R1) ; ADD NEW ADDRESS TO TABLE
    
```

```

4155 074634 016261 000002 000002      MOV      2(R2),2(R1)      ; SIX BYTES WORTH
4156 074642 016261 000004 000004      MOV      4(R2),4(R1)      ;
4157 074650 062701 000006      20$:    ADD      #6,R1          ; POINT R1 TO DATA
4158 074654 063721 050500      ADD      S.NREC,(R1)+     ; UPDATE SUMMARY DATA, RECEIVES NOT COMPLETE
4159 074660 063721 050476      ADD      S.REC,(R1)+     ; RECEIVES COMPLETE
4160 074664 063721 050502      ADD      S.LEN,(R1)+     ; LENGTH ERRORS
4161 074670 063721 050504      ADD      S.COMP,(R1)+    ; COMPARE ERRORS
4162 074674 063721 050506      ADD      S.BYTE,(R1)+    ; BYTES COMPARED
4163 074700 103001                BCC      22$              ; IF OVERFLOW, INCREMENT NEXT WORD
4164 074702 005511                ADC      (R1)              ;
4165 074704 062701 000002      22$:    ADD      #2,R1          ; POINT R1 TO NEXT DATA
4166 074710 063721 050510      ADD      S.XFER,(R1)+    ; BYTES TRANSFERED
4167 074714 103001                BCC      23$              ; IF OVERFLOW, INCREMENT NEXT WORD
4168 074716 005511                ADC      (R1)              ;
4169 074720 062701 000002      23$:    ADD      #2,R1          ; POINT R1 TO NEXT DATA
4170 074724 005337 050550      DEC      TEMP              ; DECR NO OF NODES COUNTER
4171 074730 001414                BEQ      30$              ; IF NO MORE, EXIT
4172 074732 010402                MOV      R4,R2            ; POINT R2 TO NEXT NODE
4173 074734 000715                BR       6$                ; AND UPDATE SUMMARY DATA
4174 074736      25$:    PRINTF   #TABFUL,#SUMM ; PRINT TABLE FULL MESSAGE
      074736 012746 053706                MOV      #SUMM,-(SP)
      074742 012746 053560                MOV      #TABFUL,-(SP)
      074746 012746 000002                MOV      #2,-(SP)
      074752 010600                MOV      SP,R0
      074754 104417                TRAP    C$PNTF
      074756 062706 000006                ADD     #6,SP
4175 074762 005037 050500      30$:    CLR      S.NREC          ; CLEAR SUMMARY DATA COUNTERS
4176 074766 005037 050476      CLR      S.REC
4177 074772 005037 050502      CLR      S.LEN
4178 074776 005037 050504      CLR      S.COMP
4179 075002 005037 050506      CLR      S.BYTE
4180 075006 005037 050510      CLR      S.XFER
4181 075012                RETURN
4182

```

```

4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207 075014
4208 075014
4209 075016 010546
4210 075020 010102
4211 075022 062702 000002
4212 075026 012703 075200
4213 075032 012704 075130
4214 075036 012705 075132
4215 075042 012737 000012 075116
4216 075050 005037 075214
4217 075054 161411
4218 075056 005612
4219 075060 161512
4220 075062 002403
4221 075064 005237 075214
4222 075070 000771
4223 075072 062411
4224 075074 005512
4225 075076 062412
4226 075100 022525
4227 075102 052737 000060 075214
4228 075110 113723 075214
4229 075114 005327
4230 075116 000000
4231 075120 001353
4232 075122 105023
4233 075124 012605
4234 075126
4235
4236 075130 145000
4237 075132 035632
4238 075134 160400
4239 075136 002765
4240 075140 113200

.SBTTL BINDEC CONVERT A 32 BIT BINARY NUMBER TO DECIMAL
;---+
; FUNCTIONAL DESCRIPTION:
; THIS SUBROUTINE CONVERTS A 32 BIT BINARY NUMBER TO
; A DECIMAL NUMBER REPRESENTED AS AN ASCII STRING.
; INPUTS - P1 - THE ADDRESS OF THE FIRST WORD OF BINARY DATA
; BITS 0-15. THE SECOND WORD, BITS 16-31, IS
; EXPECTED TO IMMEDIATELY FOLLOW THE FIRST WORD.
; OUTPUTS - THE ASCII STRING WILL BE LOCATED STARTING AT DECSTR
; SIDE EFFECTS - NONE
; REGISTER USAGE - R1 POINTS TO BITS 0-15 OF BINARY DATA
; R2 POINTS TO BITS 16-31 OF BINARY DATA
; R3 POINTS TO THE OUTPUT STRING
; R4 POINTS TO THE POWERS OF 10 TABLE
;---+
BINDEC:
P#POP R1 ; PUT ADDRESS OF BINARY WORD INTO R1
MOV R5, -(SP)
MOV R1, R2 ; PUT ADDRESS OF SECOND WORD INTO R2
ADD #2, R2
MOV #DECSTR, R3 ; PUT ADDRESS OF OUTPUT STRING INTO R3
MOV #TENPWR, R4 ; ADDRESS OF TEN POWER TABLE
MOV #TENPWR+2, R5
MOV #10, R4
1$: CLR PART ; CLEAR PARTIAL COUNTER
2$: SUB (R4), (R1) ; SUBTRACT 10 POWER
SBC (R2)
SUB (R5), (R2)
BLT 3$ ; BRANCH IF 10 POWER TOO LARGE
INC PART ; ELSE ADD 1 TO PARTIAL
BR 2$ ; LOOP
3$: ADD (R4)+, (R1) ; RESTORE BINARY WORDS
ADC (R2) ; AND POINT R4 TO NEXT TABLE ENTRIES
ADD (R4)+, (R2)
CMP (R5)+, (R5)+
BIS #0, PART ; CHANGE PARTIAL TO ASCII
MOVB PART, (R3)+ ; AND PUT INTO OUTPUT STRING
DEC (PC)+ ; HAVE WE DONE ALL 10 DIGITS
4$: .WORD 0
BNE 1$ ; IF NO, BRANCH
CLRB (R3)+ ; IF YES, TERMINATE WITH ZERO
MOV (SP)+, R5
RETURN

TENPWR: 145000 ; 1.0 E09
35632
160400 ; 1.0 E08
2765
113200 ; 1.0 E07
    
```

J10

4241	075142	000230	230	
4242	075144	041100	041100	; 1.0 E06
4243	075146	000017	17	
4244	075150	103240	103240	; 1.0 E05
4245	075152	000001	1	
4246	075154	023420	23420	; 1.0 E04
4247	075156	000000	0	
4248	075160	001750	1750	; 1.0 E03
4249	075162	000000	0	
4250	075164	000144	144	; 1.0 E02
4251	075166	000000	0	
4252	075170	000012	12	; 1.0 E01
4253	075172	000000	0	
4254	075174	000001	1	; 1.0 E00
4255	075176	000000	0	
4256				
4257	075200		DECSTR:: .BLKB 12.	; 12 BYTES FOR ASCIZ OUTPUT STRING
4258	075214	000000	PART:: .WORD 0	; PARTIAL COUNTER
4259				

```

4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280 075216
4281 075216 013704 003144
4282 075222 013703 003146
4283 075226 121327 000003
4284 075232 003405
4285 075234 105714
4286 075236 001441
4287 075240 121327 000013
4288 075244 003023
4289 075246 111301
4290 075250 006301
4291 075252 016101 075266
4292 075256 062701 075266
4293 075262 004711
4294 075264 000760
4295
4296
4297 075266 000114
4298 075270 000134
4299 075272 000152
4300 075274 000162
4301 075276 000204
4302 075300 000270
4303 075302 000612
4304 075304 000656
4305 075306 000270
4306 075310 000256
4307 075312 000744
4308
4309
4310
4311 075314 121314
4312 075316 001403
4313 075320 004737 075364
4314 075324 000740
4315 075326 005204
4316 075330 004737 075344
4317 075334 062703 000004

.SBTTL COMMAND LINE TRAVERSE ROUTINES
;
; **
; P$TRV SUBROUTINE
;
; PARSE THE COMMAND LINE SUBROUTINE
; TAKE ACTIONS (VIA ACTION TREE) AS PARSING LINE
; PARSING DIRECTIONS FROM "CLI PARSING NODES"
; REGS USED:
;
; R1,R5=SCRATCH
; R2=ACTION CODE PARAMETER FROM TREE
; R3=PARSE TREE POINTER
; R4=INPUT STRING POINTER
; CALLING SEQUENCE:
; JSR PC,P$TRV
; --

P$TRV:
MOV P$BUFA,R4
MOV P$TREE,R3
P$TR5: CMPB (R3),#3 ;SEE IF ONE OF FIRST THREE SPECIAL CODES
BLE 5$ ;IF YES, DON'T CHECK INPUT STRING
TSTB (R4) ;SEE IF ANY CHARS LEFT IN INPUT STRING
BEQ P$EXIT ;BR IF NO
CMPB (R3),#11. ;SEE IF SPECIAL CLI CHAR CODE OR ASCII
BGT 20$ ;BR IF REGULAR ASCII CHAR.
5$: MOVB (R3),R1 ;GET SPECIAL CHAR CODE INTO R5
ASL R1
MOV 10$(R1),R1 ;BUILD TRAVERSE ROUTINE ADDRESS
ADD #10$,R1
JSR PC,(R1) ;JSR TO SPECIAL CLI TRAVERSE ROUTINE
BR P$TR5 ;GO SEE IF MORE OF STRING LEFT

10$: .WORD TRVERR-10$ ;TRAVERSE TABLE FOR "CLI FUNCTIONS"
.WORD TRVEXI-10$ ;1
.WORD TRVBR-10$ ;2
.WORD TRVBIF-10$ ;3
.WORD TRVSPA-10$ ;4
.WORD TRVNUM-10$ ;5
.WORD TRVALP-10$ ;6
.WORD TRVALN-10$ ;7
.WORD TRVOCT-10$ ;8
.WORD TRVDEC-10$ ;9
.WORD TRVSTR-10$ ;10

;NOT A SPECIAL CODE
20$: CMPB (R3),(R4) ;SEE IF FIRST CHAR OF STRING IS A MATCH
BEQ 22$ ;BR IF A MATCH
JSR PC,TRVBRC ;IF NOT A MATCH, GO TAKE MISS BRANCH
BR P$TR5 ;THEN GO BACK PT'G TO MISS NODE
22$: INC R4 ;IF A MATCH, INCR. CHAR POINTER
JSR PC,TRVACT ;GO DO ACTION DEFINED BY
ADD #4,R3 ;ACTION CODE IN CLI NODE, THEN

```

```

4318
4319 075340 00073?          BR      P$TR5          ; ADJUST PTR TO NEXT CLI NODE
4320
4321 075342 000207          P$EXIT: RTS      PC          ;RETURN FROM PARSER
4322
4323          ;-----
4324
4325          ;GOTO USER ACTION ROUTINE
4326 075344 116302 000001    TRVACT: MOV      1(R3),R2          ;GET ACTION CODE FROM CLI NODE
4327 075350 042702 177400          BIC      #177400,R2          ;CLEAR ANY SIGN EXTENSION
4328 075354 013701 003150          MOV      P$ACT,R1          ;GET ADDRESS OF CLI ACTION ROUTINE
4329 075360 004711          JSR      PC,(R1)          ;GO DO ACTION DEFINED BY CODE
4330 075362 000207          RTS      PC          ;RETURN TO CALLING CODE
4331
4332          ;TAKE BRANCH IN TREE
4333 075364 016301 000002    TRVBRC: MOV      2(R3),R1          ;GE. BRANCH DISPLACEMENT FROM TREE
4334 075370 060103          ADD      R1,R3          ; AND POINT R3 TO THE "MISS" NODE
4335 075372 000207          RTS      PC          ; RETURN TO P$TRV
4336
4337          ;NO BRANCH TAKEN
4338 075374 062703 000004    TRVNOB: ADD      #4,R3          ;THINGS OK, UPDATE R3 TO POINT TO NEXT
4339 075400 000207          RTS      PC          ; NODE AND RETURN TO P$TRV
4340
4341          ;-----
4342          ;ERROR HANDLING
4343 075402 004737 075344    TRVERR: JSR      PC,TRVACT          ;TAKE ERROR ACTION
4344 075406 112737 177777 003161    MOV      # -1,P$GDBD          ;SET ERROR RETURN FLAG
4345 075414 005726          TST      (SP)+          ;GET RID OF "JSR PUSH TO TRVERR"
4346 075416 000137 075342    JMP      P$EXIT          ;RETURN DIRECT TO EXIT OF P$TRV ROUTINE
4347
4348          ;EXIT ACTION CODE
4349 075422 004737 075344    TRVEXI: JSR      PC,TRVACT          ;TAKE EXIT ACTION
4350 075426 105037 003161    CLRB     P$GDBD          ;SET GOOD/BAD FLAG TO "SUCCESS (0)"
4351 075432 005726          TST      (SP)+          ;GET RID OF "JSR PUSH TO TRVEXI"
4352 075434 000137 075342    JMP      P$EXIT          ;RETURN DIRECT TO EXIT OF P$TRV ROUTINE
4353
4354          ;BRANCH ACTION CODE
4355 075440 004737 075344    TRVBR:  JSR      PC,TRVACT          ;GO TAKE BRANCH ACTION
4356 075444 000137 075364    JMP      TRVBRC
4357
4358          ;BRANCH-IF ACTION CODE
4359 075450 004737 075344    TRVBIF: JSR      PC,TRVACT
4360 075454 105737 003161          TSTB     P$GDBD          ;SEE IF P$GDBD SET OR CLEARED BY ACTION
4361 075460 001402          BEQ      1$          ;IF CLEAR FALL THRU TO NEXT NODE
4362 075462 000137 075364          JMP      TRVBRC          ;ELSE TAKE THE "MISS" BRANCH
4363 075466 000137 075374    1$:      JMP      TRVNOB          ;JUST UPDATE TO NEXT NODE IF THINGS OK
4364
4365          ;SPACE ACTION CODE
4366 075472 005001          TRVSPA: CLR      R1          ;CLEAR "SPACE OR TAB FOUND" FLAG
4367 075474 121427 000011    1$:      CMPB     (R4),#11          ;SEE IF CHAR. IN CMD LINE = TAB
4368 075500 001003          BNE      2$          ;BR IF NO, NOT A TAB
4369 075502 005204          INC      R4          ;INC INPUT STRING POINTER
4370 075504 005201          INC      R1          ;INDICATE A TAB FOUND
4371 075506 000772          BR       1$          ;GO CHECK NEXT CHAR
4372
4373 075510 121427 000040    2$:      CMPB     (R4),#40          ;SEE IF CHAR. IN CMD LINE = SPACE
4374 075514 001003          BNE      10$          ;BR IF NO, NON-SPACE OR NON-TAB CHAR.

```



```

4375 075516 005204          INC      R4          ;INC INPUT STRING POINTER
4376 075520 005201          INC      R1          ;INDICATE A SPACE FOUND
4377 075522 000764          BR       1$         ;GO CHECK NEXT CHAR
4378 075524 005701          10$:    TST      R1          ;SEE IF ANY SPACES OR TABS FOUND
4379 075526 001404          BEQ     15$         ;BR IF NO, TAKE NO ACTION
4380 075530 004737 075344    JSR     PC,TRVACT   ;GO TAKE ACTION IF ANY FOUND
4381 075534 000137 075374    JMP     TRVNOB      ;JUST GO UPDATE R3 TO NEXT NODE IF OK
4382 075540 000137 075364    15$:    JMP     TRVBRC      ;TAKE BRANCH (MISS) IF NONE FOUND
4383
4384
4385 075544 012737 000012 003156 TRVDEC: MOV     $10.,P$RADX      ;USE DECIMAL AS RADIX AND ASSUME +
4386 075552 000137 075564          JMP     TRVNMA
4387 075556          TRVOCT: ;(SAME AS TRVNUM SINCE DEFAULT RADIX IS OCTAL)
4388 075556 012737 000010 003156 TRVNUM: MOV     $8.,P$RADX      ;USE OCTAL AS RADIX AND ASSUME +
4389 075564          TRVNMA: PUSH    R5
4390 075566 005001          CLR     R1          ;CLEAR DIGIT COUNTER
4391 075570 121427 000053          CMPB   (R4),#'+'      ;SEE IF THERE'S A + SIGN THERE
4392 075574 001001          BNE    10$         ; BR IF NO
4393 075576 000406          BR     11$         ; ELSE P$RADX ALREADY SAYS +, JUST BR
4394 075600 121427 000055          10$:    CMPB   (R4),#'-'      ;SEE IF THERE'S A - SIGN THERE
4395 075604 001004          BNE    1$         ; BR IF NO
4396 075606 112737 177777 003157          MOVB   $-1,P$RADX+1  ;SET "MINUS FLAG" (HI BYTE OF P$RADX)
4397 075614 005204          11$:    INC     R4          ;BUMP R4 TO POINT TO FIRST CHAR
4398
4399 075616 121427 000060          1$:    CMPB   (R4),#60      ;SEE IF CHAR. LESS THAN A "0"
4400 075622 002434          BLT    2$         ;BR IF YES (NOT NUMERIC)
4401 075624 121427 000067          CMPB   (R4),#67      ;SEE IF CHAR. GREATER THAN A "7"
4402 075630 003426          BLE    13$        ; BR IF YES
4403 075632 123727 003156 000012          CMPB   P$RADX,#10.    ;SEE IF IN DECIMAL MODE
4404 075640 001417          BEQ    12$        ; BR IF YES (CAN USE HIGHER LIMIT)
4405 075642 121427 000071          CMPB   (R4),#71      ;SEE IF DIGIT WAS A 8 OR 9
4406 075646 003027          BGT    2$         ;BR IF NON-NUMERIC
4407 075650          PRINTF $CLIBRX      ;ELSE WAS A 8 OR 9 WHEN IN OCTAL RADIX
4408 075650 012746 052554          MOV     $CLIBRX,-(SP)
4409 075654 012746 000001          MOV     $1,-(SP)
4410 075660 010600          MOV     SP,RC
4411 075662 104417          TRAP   C$PNTF
4412 075664 062706 000004          ADD     $4,SP
4408 075670 112737 177777 003161          MOVB   $-1,P$GDBD    ;SET ERROR RETURN FLAG
4409 075676 000475          BR     5$         ; PRINT ERROR AND TAKE MISS
4410
4411 075700 121427 000071          12$:    CMPB   (R4),#71      ;SEE IF CHAR. GREATER THAN A "9"
4412 075704 003003          BGT    2$         ;BR IF YES (NOT NUMERIC)
4413 075706 005204          13$:    INC     R4          ;UPDATE CMD LINE PTR TO NEXT CHAR.
4414 075710 005201          INC     R1          ;INDICATE A NUMERIC FOUND
4415 075712 000741          BR     1$         ;GO LOOK AT NEXT CHAR.
4416
4417 075714 005701          2$:    TST     R1          ;SEE IF FOUND ANY NUMERICS
4418 075716 001465          BEQ    5$         ;BR IF NO, TAKE "MISS" BRANCH
4419 075720 010405          MOV     R4,R5      ;GET POINTER TO START OF NUMERIC STRING
4420 075722 160105          SUB     R1,R5
4421 075724 005037 003154          CLR     P$NUM
4422 075730 112502          3$:    MOVB   (R5)+,R2      ;CLEAR LOC. WHERE VALUE WILL BE STORED
4423 075732 162702 000060          SUB     $60,R2      ;GET ASCII CHAR AND CONVERT IT TO A #
4424 075736 006337 003154          ASL    P$NUM
4425 075742 103440          BCS    7$         ;SHIFT CURRENT VALUE TO MAKE ROOM
4426 075744 013737 003154 003152          MOV     P$NUM,P$CNT  ;ERROR IF NUMBER TOO BIG
                          ;SAVE FOR LATER IN CASE DECIMAL RADIX

```

```

4427 075752 006337 003154      ASL      P$NUM
4428 075756 103432      BCS      7$      ;ERROR IF NUMBER TOO BIG
4429 075760 006337 003154      ASL      P$NUM
4430 075764 103427      BCS      7$      ;ERROR IF NUMBER TOO BIG
4431 075766 123727 003156 000012    CMPB     P$RADX,#10. ;SEE IF DECIMAL RADTX
4432 075774 001004      BNE      4$      ;BR IF NOT EQUAL
4433 075776 063737 003152 003154    ADD      P$CNT,P$NUM
4434 076004 103417      BCS      7$      ;ERROR IF NUMBER TOO BIG
4435 076006 060237 003154      4$:     ADD      R2,P$NUM
4436 076012 103414      BCS      7$      ;ERROR IF NUMBER TOO BIG
4437 076014 005301      DEC      R1
4438 076016 001344      BNE      3$
4439 076020 105737 003157    TSTB     P$RADX+1    ;SEE IF NUM WAS PRECEDED BY A - SIGN
4440 075024 001402      DEQ      15$     ; BR IF NO
4441 076026 005437 003154      NEG      P$NUM     ; ELSE NEGATE THE NUMBER BEFORE LEAVING
4442 076032      POP      R5       ;RESTORE R5
4443 076034 001737 075344    JSR      PC,TRVACT ;SINCE NUMERIC FOUND, GO TAKE ACTION
4444 076040 000137 075374    JMP      TRVNOB   ;GO POINT R3 TO NEXT NODE
4445
4446 076044      7$:     PRINTF   @CLINBG   ;PRINT NUMBER TOO BIG ERROR
      076044 012746 052527      MOV      @CLINBG,-(SP)
      076050 012746 000001      MOV      @1,-(SP)
      076054 010600      MOV      SP,R0
      076056 104417      TRAP     C$PNTF
      076060 062706 000004      ADD      @4,SP
4447 076064 112737 177777 003161    MOVB     @-1,P$G0BD ;SET ERROR RETURN FLAG
4448 076072      5$:     POP      R5       ;RESTORE R5
4449 076074 000137 075364    JMP      TRVBRC   ;TAKE "MISS" BRANCH
4450
4451
4452 076100 005001      TRVALP: CLR      R1       ;CLEAR ALPHA FOUND FLAG
4453 076102 121427 000101    1$:     CMPB     (R4),@101 ;SEE IF CHAR. LESS THAN A "A"
4454 076106 002406      BLT      2$      ;BR IF YES (NOT ALPHA)
4455 076110 121427 000132    CMPB     (R4),#132 ;SEE IF CHAR. GREATER THAN A "Z"
4456 076114 003003      BGT      2$      ;BR IF YES (NOT ALPHA)
4457 076116 005204      INC      R4       ;UPDATE CMD LINE PTR TO NEXT CHAR
4458 076120 005201      TNC      R1       ;INDICATE AN ALPHA WAS FOUND
4459 076122 000767      BR       1$      ;GO LOOK AT NEXT CHAR.
4460 076124 005701      2$:     TST      R1       ;SEE IF ANY ALPHA'S WERE FOUND
4461 076126 001404      BEQ      3$      ;BR IF NO
4462 076130 074737 075344    JSR      PC,TRVACT ;IF ANY FOUND TAKE ACTION
4463 076134 000137 075374    JMP      TRVNOB   ;THEN UPDATE R3 TO NEXT NODE -NO BRANCH
4464 076140 000137 075364    3$:     JMP      TRVBRC   ;NONE FOUND, TAKE MISS BRANCH
4465
4466 076144 005001      TRVALN: CLR      R1       ;CLEAR ALPHANUM FOUND FLAG
4467 076146 121427 000060    10$:    CMPB     (R4),@60  ;SEE IF CHAR. LESS THAN A "0"
4468 076152 002417      BLT      2$      ;BR IF YES (NOT NUMERIC OR ALPHA)
4469 076154 121427 000072    CMPB     (R4),#72  ;SEE IF CHAR. GREATER THAN A "9"
4470 076160 003003      BGT      1$      ;BR IF YES (NOT NUMERIC)
4471 076162 005204      INC      R4       ;UPDATE CMD LINE PTR TO NEXT CHAR.
4472 076164 005201      INC      R1       ;INDICATE A NUMERIC FOUND
4473 076166 000767      BR       10$     ;GO LOOK AT NEXT CHAR.
4474 076170 121427 000101    1$:     CMPB     (R4),@101 ;SEE IF CHAR. LESS THAN A "A"
4475 076174 002406      BLT      2$      ;BR IF YES (NOT ALPHA)
4476 076176 121427 000132    CMPB     (R4),#132 ;SEE IF CHAR. GREATER THAN A "Z"
4477 076202 005103      BGT      2$      ;BR IF YES (NOT ALPHA)
4478 076204 005204      INC      R4       ;UPDATE CMD LINE PTR TO NEXT CHAR

```

```

4479 076206 005201          INC      R1          ;INDICATE AN ALPHA FOUND
4480 076210 000756          BR       10#         ;GO LOOK AT NEXT CHAR.
4481 076212 005701          2# :    TST      R1          ;SEE IF ANY ALPHANUM'S WERE FOUND
4482 076214 001404          BEQ      3#         ;BR IF NO
4483 076216 004737 075344   JSR      PC,TRVACT  ;IF ANY FOUND TAKE ACTION
4484 076222 000137 075374   JMP      TRVNOB     ;THEN UPDATE R3 TO NEXT NODE -NO BRANCH
4485 076226 000137 075364   3# :    JMP      TRVBRC     ;NONE FOUND, TAKE MISS BRANCH
4486
4487
4488
4489 076 32                TRVS#R: PUSH     R5          ;SAVE R5
4490 076 34 010401         MOV      R4,R1      ;POINT R1 TO CMD STRING
4491 076236 010305         MOV      R3,R5
4492 076240 062705 000006   ADD      @6,R5      ;POINT R5 TO MATCH STRING FROM CLI NODE
4493 076244 005037 003152   CLR      P#CNT     ;CLEAR CHAR MATCH COUNT
4494 076250 105715          2# :    CLR      P#CNT     ;SEE IF END OF MATCH STRING YET
4495 076252 001411         TSTB    (R5)
4496 076254 105711         BEQ      10#        ;BR IF YES
4497 076256 001407         TSTB    (R1)       ;SEE IF END OF CMD LINE YET
4498 076260 121115         BEQ      10#        ;BR IF YES
4499 076262 001005         CMPB    (R1),(R5)  ;SEE IF CHARACTERS MATCH
4500 076264 005237 003152   BNE     10#        ;BR IF NO
4501 076270 005201         INC      P#CNT     ;MATCH -INCREMENT MATCH COUNT
4502 076272 005205         INC      R1
4503 076274 000765         INC      R5
4504                                BR       2#         ;UPDATE STRING POINTERS
4505 076276 005737 003152   10# :   TST      P#CNT     ;BR TO CONTINUE CHECKING CHARS.
4506 076302 001407         BEQ      15#
4507 076304 010104         MOV      R1,R4
4508 076306                    POP      R5
4509 076310 004737 075344   JSR      PC,TRVACT  ;WHEN DONE SEE IF ANY MATCHES FOUND
4510 076314 066303 000004   ADD      4(R3),R3  ;BR IF NO, GO TAKE THE MISS BRANCH
4511 076320 000207         RTS      PC        ;POINT CMD POINTER TO END OF STRING &
4512                                ;RESTORE R5
4513 076322                    ;IF A MATCH FOUND, GO DO MATCH ACTION
4514 076324 000137 075364   15# :   POP      R5          ;UPDATE R3 TO NEXT NODE (NO BRANCH)
4515                                ; (NO RETURN THR) TRVNOB SINCE DIFFERENT
4516                                ; DISPLACEMENT DUE TO MATCH STRING)
4517                                ;RESTORE R5
4518                                ; GO TAKE BRANCH
4519                                ; (PARSED OK), -1 IF ILL CMD.....

```

```

4518 TRVADR TRAVRSE COMMAND LINE INPUT ADDRESS
4519
4520 THIS ROUTINE IS CALLED BY TWO DIFFERENT ACTION ROUTINES. THE
4521 NODE ACTION ROUTINE CALLS IT TO PARSE THROUGH THE NODE
4522 ADDRESS INPUT BY THE OPERATOR. THE OPRSEL ACTION ROUTINE
4523 CALLS TRVADR TO PARSE THROUGH THE "OPERATOR SELECTED" MESSAGE
4524 WHICH HAS BEEN INPUT IN THE COMMAND LINE. FOR A NODE ADDRESS,
4525 THE ROUTINE LOOKS FOR A '/' AS A DELIMITER FOR THE ADDRESS,
4526 AND REPLACES THE / WITH A NULL BYTE FOR USE BY THE ADDRESS
4527 PACKING ROUTINE. WHEN CALLED BY THE OPRSEL ROUTINE, A ''
4528 IS EXPECTED AS THE DELIMITER FOR THE OPERATOR SELECTED MESSAGE.
4529 IF A NULL STRING IS ENTERED, AN ERROR MESSAGE IS PRINTED.
4530
4531 INPUTS R4 POINTS TO THE BEGINING OF THE ADDRESS
4532 OR MESSAGE IN THE COMMAND LINE
4533 OUTPUTS SUMMARIZED IN TABLE BELOW
4534
4535 COMMAND LINE OUTPUTS

```

ADDRESS	INPUT CONDITION	P#GDBD	R4 POINTS TO	CFLAG CONTAINS	P#ERR
4536	ILLEGAL CHAR.	-1	ILL. CHAR.		N/A
4537	ADR./ASSIST	0	END OF LINE	CASIST	N/A
4538	ADR./TARGET				
4539	ADR./	0	END OF LINE	CTARGET	N/A
4540	ADR.				
4541	ADR./CHAR. OR				
4542	"OPR SEL/CHAR.				
4543	OTHER THAN "A"	-1	/	CTARGET	N/A
4544	"T" OR BLANK				
4545	" "	0	CHAR. AFTER "		-1
4546	"OPR SEL"	0	CHAR. AFTER "	OPRSEL	0
4547					
4548	CALLING PROCEDURE -	JSR	PC,TRVADR		
4549	REGISTER USAGE -				
4550					
4551					
4552					
4553					
4554					
4555					
4556					

R1 IS USED AS A COUNTER TO REPORT ERROR MESSAGES
IF NULL STRINGS ARE ENTERED.
R4 POINTS TO THE NEXT CHAR. IN THE COMMAND LINE

```

4557 076330 005001 TRVADR: CLR R1 ;CLEAR HEX DIGIT FOUND FLAG
4558 076332 121427 000000 1#: CMPB (R1),#0 ;SEE IF NULL CHAR.
4559 076336 001423 BEQ 20# ; IF YES, RETURN
4560 076340 121427 000040 CMPB (R4),#40 ;SEE IF ILLEGAL CHARACTER
4561 076344 002414 BLT 10# ;IF YES, BRANCH TO ERROR ROUTINE
4562 076346 121427 000042 CMPB (R4),#42 ;SEE IF CHAR. IS A "/"
4563 076352 001454 BEQ 40# ;IF YES, BRANCH TO 40#
4564 076354 121427 000057 CMPB (R4),#57 ;SEE IF CHAR. IS A " "
4565 076360 001420 BEQ 30# ;BRANCH IF YES
4566 076362 121427 000132 CMPB (R4),#132 ;SEE IF CHAR. GREATER THAN "F"
4567 076366 003003 BGT 10# ; IF YES, ILLEGAL CHAR.
4568 076370 005204 INC R4 ;UPDATE CMD LINE POINTER TO NEXT CHAR.
4569 076372 005201 INC R1 ;INDICATE A VALID CHAR. FOUND
4570 076374 000756 BR 1# ;LOOK AT NEXT CHAR.
4571 076376 112737 177777 003161 10#: MOVB #-1,P#GDBD ;SET ERROR FLAG
4572 076404 000464 BR 50# ;RETURN
4573 076406 005701 20#: TST R1 ;SEE IF VALID CHARACTERS FOUND
4574 076410 001772 BEQ 10# ; IF NO, ILLEGAL CHAR.
4575 076412 012737 000000 003672 25#: MOV #CTARGET,CFLAG ;SET TARGET FLAG
4576 076420 000456 BR 50# ;RETURN
4577 076422 005701 30#: TST R1 ;SEE IF VALID CHARACTERS FOUND
4578 076424 001764 BEQ 10# ; IF NO, ILLEGAL CHAR.
4579 076426 112714 000000 MOVB #0,(R4) ; IF YES, REPLACE "/" WITH NULL CHAR.
4580 076432 005204 INC R4 ;UPDATE CMD. LINE POINTER TO NEXT CHAR.
4581 076434 121427 000000 CMPB (R4),#0 ;IS NEXT CHAR. NULL
4582 076440 001764 BEQ 25# ; IF YES, TAKE DEFAULT OF TARGET
4583 076442 121427 000101 CMPB (R4),#A ;IS NEXT CHAR. "A"
4584 076446 001412 BEQ 35# ; IF YES, SR 35#
4585 076450 121427 000124 CMPB (R4),#T ;IS NEXT CHAR. "T"
4586 076454 001756 BEQ 25# ; IF YES, SET TARGET FLAG
4587 076456 112737 177777 003161 MOVB #-1,P#GDBD ; ELSE, SET ERROR FLAG.
4588 076464 005304 DEC R4 ; READJUST COMMAND LINE POINTER
4589 076466 112714 000057 MOVB #/, (R4) ; AND REPLACE / IN CMD LINE TO FIX ERROR
4590 076472 000747 BR 25# ; SET TARGET FLAG AND RETURN
4591 076474 012737 000001 003672 35#: MOV #CASIST,CFLAG ;SET ASSIST FLAG
4592 076502 000425 BR 50#
    
```

```

4533 076504 005701          40$:  TST      R1          ;SEE IF ANY CHARACTERS TYPED
4594 076506 001407          BEQ      45$          ;IF NO, BRANCH TO 45$
4595 076510 112714 000000    MOVB     00,(R4)      ;ELSE, REPLACE ' ' WITH NULL
4596 076514 012737 000006 003672  MOV     @PRSEL,CFLAG ;SET OPERATOR SELECTED FLAG
4597 076522 005204          INC      R4
4598 076524 000414          BR       50$          ;RETURN
4599 076526          45$:  PRINTF   @NULSTR      ;PRINT NULL STRING ERROR MESSAGE
      076526 012746 053362          MOV     @NULSTR,-(SP)
      076532 012746 000001          MOV     @1,-(SP)
      076536 010600          MOV     SP,R0
      076540 104417          TRAP   C#PNTF
      076542 062706 000004          ADD    @4,SP
4600 076546 112737 177777 003163  MOVB     @-1,P#MERR ;SET OPER. SELECTED MSG. ERROR FLAG
4601 076554 005204          INC      R4          ;MOVE CMD. LINE POINTER TO NEXT CHAR.
4602 076556 000207          50$:  RTS      PC          ;RETURN
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614 076560          ;-----
      076560          .SBYTL  REPORT CODING SECTION
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628 076560 004737 101710          ;**
      076564          ; THE REPORT CODING SECTION CONTAINS THE
      076564          ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
      076566          ;--
      076566          BGNRPT
4629
4630
4631
4632
4633
4634
4635
4636
      L#RPT::
      .WORD  J#JMP
      .WORD  L10006-2-
      ;*****
      ; INSERT LOCAL STORAGE THAT IS USED ONLY
      ; DURING THE REPORT SECTION.
      ;*****

```

E11

```
4637 ;*****  
4638 ;   INSERT MESSAGES THAT ARE USED ONLY  
4639 ;   DURING THE REPORT SECTION.  
4640 ;*****  
4642  
4643         .EVEN  
4644  
4645 076570         ENDRPT  
076570  
076570 104425  
L10006: TRAP C$RPT
```

4647
 4648
 4649
 4650
 4651
 4652
 4653
 4654 076572
 076572
 4655
 4656 076572 177777
 4657 076574 177777
 4658 076576 177777
 4659
 4660 076600
 4661
 4663
 4664
 4665
 4666
 4667
 4668
 4669
 4670
 4671
 4672
 4673

.SBTTL PROTECTION TABLE

```

; **
; THIS TABLE IS USED BY THE RUNTIME SERVICES
; TO PROTECT THE LOAD MEDIA.
; --

```

BGNPROT

L\$PROT::

```

-1 ; OFFSET INTO P-TABLE FOR CSR ADDRESS
-1 ; OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
-1 ; OFFSET INTO P-TABLE FOR DRIVE NUMBER

```

ENDPROT

```

; ****
; INSERT BYTE OFFSET FOR DATA NOTED IN COMMENTS ABOVE. (OFFSET
; REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A PTABLE
; ENTRY TO THE ITEM IN QUESTION.) IF THE PARTICULAR
; ITEM DOES NOT APPLY, LEAVE ENTRY AS -1. WHEN THE RUNTIME
; SERVICES EXECUTES A GPHARD, IT USES THESE OFFSETS (IF NOT
; SET TO -1) TO GET THE ITEMS AND COMPARE WITH THOSE SAVED
; IN THE XXDP MONITOR. IF THE UNIT BEING REQUESTED MATCHES THE
; LOAD DEVICE, THE RUNTIME SERVICES RETURN AN INCOMPLETE FLAG ON
; THE GPHARD.
; ****

```

```

4676 .SBYTL INITIALIZE SECTION
4677
4678 ;**
4679 ; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
4680 ; AT THE BEGINNING OF EACH PASS.
4681 ;**
4682
4683 076600          BGNINIT
4684 076600          L$INIT:
4685
4686 ;*****
4687 ; THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS. THERE
4688 ; ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
4689 ; DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
4690 ; PLACE. THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
4691 ; THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
4692 ; CORRESPONDING EVENT FLAGS ARE:
4693 ;
4694 ;          START COMMAND          EF.START
4695 ;          RESTART COMMAND       EF.RESTART
4696 ;          CONTINUE COMMAND      EF.CONTINUE
4697 ;          POWERDOWN/POWERUP    EF.PWR
4698 ;          NEW PASS              EF.NEW
4699 ;
4700 ; EXAMPLE OF EVENT FLAG USE:
4701 ;          READEF  *EF.START
4702 ;          BCOMPLETE          STARTCODE
4703 ;
4704 ; DURING THE INIT CODE, USE THE "GPHARD" MACRO TO OBTAIN P-TABLE
4705 ; INFORMATION FOR DEVICE TESTING. GET ONE UNIT'S INFORMATION IF
4706 ; THIS IS A SEQUENTIAL DIAGNOSTIC. GET INFORMATION ON ALL
4707 ; UNITS AVAILABLE FOR TESTING IF THIS IS AN EXERCISER. THE NUMBER
4708 ; OF UNITS AVAILABLE IS IN A HEADER LOCATION: "L$UNIT".
4709 ;*****
4709 076600 022737 000020 003672 INIT:  CMP      *CEXIT,CFLAG          ;SEE IF EXIT COMMAND TYPED
4710 076606 001004          BNE      INIT1              ; IF NO, DO INIT CODE
4711 076610 005037 003672          CLR      L$FLAG              ; ELSE, CLEAR EXIT FLAG
4712 076614 000137 100130          JMP      INICLN              ; AND DO CLEANUP
4713 076620          INIT1:  READEF  *EF.START          ;IF HERE BECAUSE OF "START", DO INIT
4714 076620 012700 000040          MOV      *EF.START,R0
4715 076624 104447          TRAP     C$REFG
4716 076626          BCOMPLETE          START
4717 076626 103424          READEF  *EF.RESTART          ;IF HERE BECAUSE OF "RESTART", DO SOME INIT
4718 076630          BCS      START
4719 076630 012700 000037          MOV      *EF.RESTART,R0
4720 076634 104447          TRAP     C$REFG
4721 076636          BNCOMPLETE          5$
4722 076636 103002          BCC      5$
4723 076640 000137 100046          JMP      RESTR
4724 076644          5$:  READEF  *EF.CONTINUE          ;IF HERE BECAUSE OF "CONTINUE", EXIT
4725 076644 012700 000036          MOV      *EF.CONTINUE,R0
4726 076650 104447          TRAP     C$REFG
4727 076652          BNCOMPLETE          10$
4728 076652 103002          BCC      10$
4729 076654 000137 100046          JMP      RESTR
4730 076660          10$: READEF  *EF.NEW              ;IF HERE ON NEW PASS, SKIP SOME INIT
4731 076660 012700 000035          MOV      *EF.NEW,R0
4732 076664 104447          TRAP     C$REFG
4733 076666          BNCOMPLETE          15$

```



```

076666 103002
4723 076670 000137 100112
4724 076674 000137 100132
4725 076700
4726 076706
076706 104431
076710 010037 047660
4727 076714 013737 047660 047662
4728 076722 062737 000002 047662
4729 076730 012702 003674
4730 076734
076734 012700 000114
076740 104462
076742 010001
4731 076744
076744 103006
4732 076746 004737 067656
4733 076752 012737 000100 003704
4734 076760 000436
4735 076762
076762 012700 000120
076766 104462
076770 010001
4736 076772
076772 103017
4737 076774 004737 067656
4738 077000 062737 000002 003674
4739 077006 012777 001600 104660
4740 077014 162737 000002 003674
4741 077022 012737 000111 003704
4742 077030 000412
4743 077032
077032 012746 062622
077036 012746 000001
077042 010600
077044 104417
077046 062706 000004
4744 077052 000137 100130
4745 077056
077056 012700 000000
077062 104442
077064 010001
4746 077066
077066 103402
4747 077070 000137 100130
4748 077074 012137 047652
4749 077100 012137 047654
4750 077104 012137 047656
4751 077110
077110 013746 047656
077114 012746 071144
077120 013746 047654
077124 012746 000003
077130 104437
077132 062706 000010
4752 077136 013737 047652 047632
4753 077144 013737 047632 047634

```

```

NEW
INIEXI
I$STACK $1000,SP
MEMORY FRESIZ
MOV FRESIZ,FREMEM
ADD $2,FREMEM
MOV $CLKCSR,R2
CLOCK L,R1
BNCOMPLETE 1$
JSR PC,CLKSET
MOV $LCLKEN,CLKEN
BR 3$
CLOCK P,R1
BNCOMPLETE 2$
JSR PC,CLKSET
ADD $2,CLKCSR
MOV $PCLKCT,$CLKCSR
SUB $2,CLKCSR
MOV $PCLKEN,CLKEN
BR 3$
PRINTF $NOCLK
JMP INICLN
GPHARD $0,R1
BCOMPLETE 4$
JMP INICLN
MOV (R1)+,UNACSR
MOV (R1)+,UNAVEC
MOV (R1)+,UNAPRI
SETVEC UNAVEC,$UNAIISR,UNAPRI
MOV UNACSR,PCSR0
MOV PCSRO,PCSR1

```

```

BCC 15$
;IF DON'T KNOW WHY WE'RE HERE, EXIT
;SET PARAMETER STACK POINTER
;GET FREE MEMORY INFO
TRAP C$MEM
MOV RO,FRESIZ
;SIZE OF FREE MEMORY IN FRESIZ
;START OF FREE MEMORY IN FREMEM
;SETUP R2 AS A PRT. TO CLOCK INFO. BLOCK
;GET LINE CLOCK INFO
MOV $'L,RO
TRAP C$CLK
MOV RO,R1
;IF NONE, SEE IF P CLOCK PRESENT
BCC 1$
;SET UP CLOCK INFO TABLE AND VECTOR
;SET UP THE ENABLE LINE CLOCK DATA
;GET P CLOCK INFO
MOV $'P,RO
TRAP C$CLK
MOV RO,R1
;IF NO CLOCK, ERROR
BCC 2$
; ELSE SET UP CLOCK INFO AND VECTOR
; POINT CLKCSP TO P-CLK COUNT SET REG.
;LOAD CLK SEI REG WITH COUNT VALUE
;POINT CLKCSR BACK TO P-CLK CSR
;SETUP TO ENABLE P-CLK DATA
MOV $NOCLK,-(SP)
MOV $1,-(SP)
MOV SP,RO
TRAP C$PNTF
ADD $4,SP
;CANNOT CONTINUE, DO CLEANUP
;GET P-TAB POINTER FOR THIS UNIT
MOV $0,RO
TRAP C$GPHRD
MOV RO,R1
;THIS ONE IS NOT AVAILABLE
BCS 4$
;SAVE CSR
;SAVE VECTOR
;SAVE PRIORITY
;SETUP UNA INTERRUPT VECTOR
MOV UNAPRI,-(SP)
MOV $UNAIISR,-(SP)
MOV UNAVEC,-(SP)
MOV $3,-(SP)
TRAP C$SVEC
ADD $10,SP
;PCSR0

```

```

4754 077152 062737 000002 047634      ADD     #2,PCSR1           ;PCSR1
4755 077160 013737 047634 047636      MOV     PCSR1,PCSR2
4756 077166 062737 000002 047636      ADD     #2,PCSR2           ;PCSR2
4757 077174 013737 047636 047640      MOV     PCSR2,PCSR3
4758 077202 062737 000002 047640      ADD     #2,PCSR3           ;PCSR3
4759 077210 012703 000050      MOV     #TBLLEN,R3
4760 077214 012702 002404      MOV     #NODTBL,R2
4761 077220 005022      5$:    CLR     (R2)+
4762 077222 005303      DEC     R3
4763 077224 001375      BNE     5$
4764 077226 012703 000132      MOV     #STBLLEN,R3       ;CLEAR SUMMARY TABLE
4765 077232 012702 002656      MOV     #STATBL,R2
4766 077236 005022      6$:    CLR     (R2)+
4767 077240 005303      DEC     R3
4768 077242 001375      BNE     6$
4769 077244 005037 050500      CLR     S.NREC           ; CLEAR SUMMARY DATA COUNTERS
4770 077250 005037 050476      CLR     S.REC
4771 077254 005037 050502      CLR     S.LEN
4772 077260 005037 050504      CLR     S.COMP
4773 077264 005037 050506      CLR     S.BYTE
4774 077270 005037 050510      CLR     S.XFER
4775 077274 005037 003706      CLR     TIMMIN           ;CLEAR TIME SINCE-START-LOCATIONS
4776 077300 005037 003710      CLR     TIMSEC
4777 077304 013737 003702 003712      MOV     CLKHZ,TIMTCK
4778 077312      SETVEC  CLKVEC,#CLKINT,CLKBR ;LOAD TICKS/SEC
4779 077312 013746 003676      MOV     CLKBR,-(SP)       ;SETUP CLOCK INTERRUPT VECTOR
4780 077316 012746 067702      MOV     #CLKINT,-(SP)
4781 077322 013746 003700      MOV     CLKVEC,-(SP)
4782 077326 012746 000003      MOV     #3,-(SP)
4783 077332 104437      TRAP    C$SVEC
4784 077334 062706 000010      ADD     #10,SP
4785 077340 013777 003704 104326      MOV     CLKEN,#CLKCSR    ;SET ENABLE BITS IN THE CLOCK TO START
4786 077346      SETPRI  #PRI00          ;SET PRIORITY=0 SO CLOCK CAN INTERRUPT
4787 077346 012700 000000      MOV     #PRI00,R0
4788 077352 104441      TRAP    C$SPRI
4789 077354      CALL    UNAINI          ;INITIALIZE THE UNA
4790 077362      CALL    FUNCT #RDDEFA  ;READ UNA DEFAULT PHYSICAL ADDRESS
4791 077374      P$POP  R2              ;CHECK FOR ERROR
4792 077376 001402      BEQ     8$
4793 077400 000137 100034      JMP     20$
4794 077404      8$:    CALL    BINHEX #PCBB2,#6,#STRBUF ;PUT ADDRESS INTO HEX FORMAT
4795 077426      PRINTS #HDMSG1,#STRBUF ;PRINT ADDRESS
4796 077426 012746 002322      MOV     #STRBUF,-(SP)
4797 077432 012746 055342      MOV     #HDMSG1,-(SP)
4798 077436 012746 000002      MOV     #2,-(SP)
4799 077442 010600      MOV     SP,R0
4800 077444 104416      TRAP    C$PNTS
4801 077446 062706 000006      ADD     #6,SP
4802 077452      CALL    FUNCT #RDSTA   ;READ STATUS TO GET ROM VERSION
4803 077464      P$POP  R2              ;CHECK FOR ERROR
4804 077466 001162      BNE     20$
4805 077470 113702 047670      MOV     PCBB2,R2
4806 077474 142702 030300      BIC     #300,R2
4807 077500      PRINTS #HDMSG2,R2     ;PRINT ROM VERSION
4808 077500 010246      MOV     R2,-(SP)
4809 077502 012746 055413      MOV     #HDMSG2,-(SP)
4810 077506 012746 000002      MOV     #2,-(SP)

```

```

077512 010600
077514 104416
077516 062706 000006
4794 077522 PRINTS #HDMMSG3 ;PRINT MORE HEADER INFO
077522 012746 055466
077526 012746 000001
077532 010600
077534 104416
077536 062706 000004
4795 077542 MOV #UCB20,R3 ;SET UP FUNCTION CONTROL BLOCK
4796 077546 MOV #2,(R3)+ ; MOVE 2 BYTES...
4797 077552 MOV #TEMP,(R3)+ ; INTO LOCATION TEMP...
4798 077556 MOV #0,(R3)+
4799 077562 MOV #21040,(R3) ; FROM LOCATION 21040
4800 077566 CALL FUNCT #DMPMEM ;DUMP INTERNAL MEMORY
4801 077600 P#POP R2 ;CHECK FOR ERROR
4802 077602 001114 BNE 20#
4803 077604 013703 050550 MOV TEMP,R3 ;PUT RESULT INTO R3
4804 077610 032703 002000 BIT #BIT10,R3 ;DETERMINE STATUS
4805 077614 001414 BEQ 11#
4806 077616 032703 004000 BIT #BIT11,R3
4807 077622 001425 BEQ 12#
4808 077624 PRINTS #HDMMSG4 ; BIT10!BIT11 = REMOTE AND POWER UP BOOT ENABLED
077624 012746 055522
077630 012746 000001
077634 010600
077636 104416
077640 062706 000004
4809 077644 000435 BR 15#
4810 077646 032703 004000 11#: BIT #BIT11,R3
4811 077652 001422 BEQ 13#
4812 077654 PRINTS #HDMMSG5 ; BIT11 = REMOTE BOOT ENABLED WITH ROM
077654 012746 055577
077660 012746 000001
077664 010600
077666 104416
077670 062706 000004
4813 077674 000421 BR 15#
4814 077676 12#: PRINTS #HDMMSG6 ; BIT10 = REMOTE BOOT ENABLED
077676 012746 055650
077702 012746 000001
077706 010600
077710 104416
077712 062706 000004
4815 077716 000410 BR 15#
4816 077720 13#: PRINTS #HDMMSG7 ; REMOTE BOOT NOT ENABLED
077720 012746 055710
077724 012746 000001
077730 010600
077732 104416
077734 062706 000004
4817 077740 032703 010000 15#: BIT #BIT12,R3
4818 077744 001411 BEQ 16#
4819 077746 PRINTS #HDMMSG8 ; BIT12 = SELF TEST ENABLED
077746 012746 055751
077752 012746 000001
077756 010600

```

```

077760 104416
077762 062706 000004
4820 077766 000410
4821 077770 16$: BR PRINTS 17$ ; SELF TEST DISABLED
077770 012746 056014
077774 012746 000001
100000 010600
100002 104416
100004 062706 000004
4822 100010 012737 000000 002370 17$: MOV $ALPHA,P$TYPE ;SET MESSAGE DEFAULT VALUES
4823 100016 012737 001000 002372
4824 100024 012737 000001 002374
4825 100032 000405
4826 100034 20$: ERDF 18,EMSG18,ERR1
100034 104455
100036 000022
100040 065127
100042 067460
4827 100044 000431
4828 100046 005037 003706
4829 100052 005037 003710
4830 100056 013737 003702 003712
4831 100064
100064 013746 003676
100070 012746 067702
100074 013746 003700
100100 012746 000003
100104 104437
100106 062706 000010
4832 100112 013777 003704 103554 NEW: MOV CLKEN,$CLKCSR ;SET ENABLE BITS IN THE CLOCK TO START
4833 100120 SETPRI $PRI00 ;SET PRIORITY=0 SO CLOCK CAN INTERRUPT
100120 012700 000000
100124 104441
4834 100126 000401
4835 100130 INIEXI: DOCLN ;EXIT
100130 104444 ;ABOUT PASS
4836 100132 INIEXI: EXIT INIT ;EXIT INIT SECTION
100132 104432
100134 000002
4837
4639
4840 ;*****
4841 ; INSERT LOCAL STORAGE THAT IS USED ONLY
4842 ; DURING THE INITIALIZE SECTION.
4843 ;*****
4844
4845 ;*****
4846 ; INSERT MESSAGES THAT ARE USED ONLY
4847 ; DURING THE INITIALIZE SECTION.
4848 ;*****
4849
4850 .EVEN
4851
4852 100136 ENDINIT
100136
100136 104411
L10010: TRAP C$INIT

```



```

4874      .SBTTL  CLEANUP CODING SECTION
4875
4876      ;++
4877      ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
4878      ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
4879      ;--
4880
4881 100142      BGNCLN
100142
4882
4884
4885      ;*****
4886      ; INSERT YOUR CLEANUP CODING. THIS CODING SHOULD
4887      ; RESTORE YOUR TEST-DEVICE TO A NEUTRAL STATE.
4888      ; THIS CODE WILL BE EXECUTED AFTER EACH PASS AND AFTER THE
4889      ; PROGRAM IS INTERRUPTED BY "+C".
4891      ;*****
4892 100142 012737 000000 050044      MOV    #0,$WDMC+4      ;CLEAR MULTICAST ADDRESS LIST
4893 100150      CALL   FUNCT  #WDMULA      ; WRITE 0 INTO LIST LENGTH
4894 100162 012737 000400 050044      MOV    #400,$WDMC+4   ; RESET FOR 1 ENTRY
4895 100170      P$POP  R2      ;CHECK FOR ERROR
4896 100172 001404      BEQ    25$      ; IF OK CONTINUE
4897 100174      ERRHRD 38,MSG25      ; ELSE, REPORT ERROR
100174 104456
100176 000046      TRAP   C$ERHRD
100200 065442      .WORD  38
100202 000000      .WORD  MSG25
100202 000000      .WORD  0
4898 100204 005737 050516      25$:  TST    NIRCNT      ;SEE IF RECEVE COUNT ZERO
4899 100210 001413      BEQ    30$      ; IF YES, EXIT
4900 100212      CALL   RECEVE      ; ELSE, CLEAR OUT UNWANTED ENTRIES
4901 100220      P$POP  R2
4902 100222 001770      BEQ    25$      ; IF ANY WERE FOUND, UPDATE NEXT POINTER
4903 100224      CALL   GETRNX  #RRGNXT
4904 100236 000762      BR     25$      ;SEE IF ANY MORE ENTRIES
4905 100240 005077 103430      30$:  CLR    @CLKCSR      ;DISABLE CLOCK
4906 100244      SETPRI #PRI07      ;SET PROCESSOR PRIORITY BACK TO 7
100244 012700 000340      MOV    #PRI07,R0
100250 104441      TRAP   C$SPRI
4907 100252      EXIT   CLN
100252 104432      TRAP   C$EXIT
100254 000002      .WORD  L10012-.
4908
4910      ;*****
4911      ; INSERT LOCAL STORAGE THAT IS USED ONLY
4912      ; DURING THE CLEANUP SECTION.
4913      ;*****
4914
4915      ;*****
4916      ; INSERT MESSAGES THAT ARE USED ONLY
4917      ; DURING THE CLEANUP SECTION.
4918      ;*****
4920
4921      .EVEN
4922
4923 100256      ENDCLN
100256
100256 104412      L10012:  TRAP   C$CLEAN

```

```

4925 .SBTTL DROP UNIT SECTION
4926
4927 ;++
4928 ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
4929 ; TO NO LONGER BE TESTED.
4930 ;--
4931
4932 100260 BGNDU
100260
4933 L$DU:
4935 ;*****
4936 ; INSERT DROP CODE HERE. THIS CODE WILL BE EXECUTED AFTER
4937 ; A "DROP" COMMAND OR A "DODU" MACRO EXECUTION. THE PURPOSE
4938 ; OF THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
4939 ; UNIT HAS BEEN DROPPED. THIS SECTION IS OPTIONAL.
4940 ;*****
4942
4943 100260 EXIT DU
100260 000167 .WORD J$JMP
100262 000000 .WORD L10013-2-
4944
4946 ;*****
4947 ; INSERT LOCAL STORAGE THAT IS USED ONLY
4948 ; DURING THE DROP-UNIT SECTION.
4949 ;*****
4950
4951 ;*****
4952 ; INSERT MESSAGES THAT ARE USED ONLY
4953 ; DURING THE DROP-UNIT SECTION.
4954 ;*****
4956
4957 .EVEN
4958
4959 100264 ENDDU
100264
100264 104453 L10013: TRAP C$DU

```

ADD UNIT SECTION

```

4961 .SBTTL ADD UNIT SECTION
4962
4963
4964
4965 ; THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
4966 ; TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
4967 ; TO THE TEST CYCLE.
4968 ; --
4969 100266 BGNAU
100266
4970 L#AU:
4971
4972 ; ~~~~~
4973 ; INSERT ADD CODE HERE. THIS CODE WILL BE EXECUTED AFTER
4974 ; AN "ADD" COMMAND. THE PURPOSE OF THIS CODE IS TO DO ANY
4975 ; HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
4976 ; THIS SECTION IS OPTIONAL.
4977 ; ~~~~~
4978
4979 100266 EXIT AU
100266 000167 .WORD J#JMP
100270 000000 .WORD L10014-2-.
4981
4982 ; ~~~~~
4983 ; INSERT LOCAL STORAGE THAT IS USED ONLY
4984 ; DURING THE ADD-UNIT SECTION.
4985 ; ~~~~~
4986
4987 ; ~~~~~
4988 ; INSERT MESSAGES THAT ARE USED ONLY
4989 ; DURING THE ADD-UNIT SECTION.
4990 ; ~~~~~
4991
4992 .EVEN
4993
4994 ENDAU
4995
4996 100272 L10014:
100272 104452 TRAP C#AU
100272
4997
4998
4999 .SBTTL TEST 1:
5000
5001 ; ~~~~~
5002 ; APPEND THE NAME OF THIS TEST TO THE .SBTTL, AS SHOWN IN THE
5003 ; FOLLOWING EXAMPLE. .SBTTL TEST 1: NAME OF TEST
5004 ; ~~~~~
5005
5006
5007
5008
5009 ; --
5010 ; TEST TO ...
5011 ; --
5012
5013 ; ~~~~~
5014 ; CHANGE THE PHRASE "TEST TO ..." TO BE A FUNCTIONAL
5015 ; DESCRIPTION OF THE HARDWARE TEST WHICH FOLLOWS.
5016 ; ~~~~~
5017
5018
5019
5020
5021

```



```

5022      ; INSERT PROGRAM EQUATES THAT ARE USED ONLY IN THIS TEST.
5023      ; *****
5025
5026 100274      BGNTST
      100274
5027
5029      ; *****
5030      ; INSERT THE CODING FOR THIS HARDWARE TEST.
5031      ; *****
5033      .SBTTL GETCL  COMMAND LINE FETCH & INTERPRETATION SECTION
5034
5035 100274 105037 003161      GETCL:  CLRB  P%GDBD      ;CLEAR CMD LINE PARSING ERROR FLAG
5036 100300 105037 003160      CLRB  P%INUF
5037 100304      GMANID  CLI$PM,CMDBUF,A,0,1,72.,NO      ;GET CMD LINE FROM OPERATOR
      100304 104443
      100306 000406
      100310 002200
      100312 000142
      100314 052436
      100316 000000
      100320 000001
      100322 000110
      100324
5038 100324 012737 002200 003144      MOV  #CMDBUF,P%BUFA
5039 100332 012737 051070 003146      MOV  #CLITRE,P%FREE
5040 100340 012737 100456 003150      MOV  #CLIACY,P%ACT
5041 100346 005037 003672      CLR  CFLAG      ;CLEAR QUALIFIER FLAG
5042 100352 004737 075216      JSR  PC,P%TRV    ;GO PARSE COMMAND TREE
5043 100356 105737 003161      TSTB P%GDBD     ;SEE IF PARSED OK, OR AN ERROR
5044 100362 001412
5045 100364      BEQ  1$
      100364 012746 052445      PRINTF #CLIERM      ;IF NOT PRINT ERROR MESSAGE
      100370 012746 000001
      100374 010600
      100376 104417
      100400 062706 000004
5046 100404 000137 100274      JMP  GETCL
5047 100410 105737 003160      1$:  TSTB  P%INUF
5048 100414 001412
5049 100416      BEQ  10$
      100416 012746 052476      PRINTF #CLINUF     ;IF NOT PRINT ERROR MESSAGE
      100422 012746 000001
      100426 010600
      100430 104417
      100432 062706 000004
5050 100436 000137 100274      JMP  GETCL
5051 100442 022737 000020 003672 10$:  CMP  #CEXIT,CFLAG
5052 100450 001311
5053 100452      BNE  GETCL
      100452 104432
      100454 012302      EXIT  TST
      TRAP  C%EXIT
      .WORD  L10015..
5054
5055      .SBTTL CLI ACTION TABLE AND ROUTINES
5056      ;
5057      ; USER MUST CLEAR/SET P%GDBD IF USE "CLIBIF" IN CONNECTION WITH ACTION
5058 100456      ; R2 WILL HOLD ACTION CODE FROM PARSING (CLI) NGOE
5059 100456 006302      CLIACT:  ASL  R2      ;MULTIPLY ACTION CODE BY 2

```

Address	Offset	Code	Label	Operation	Description
5060	100460	016202	100474	MOV	10*(R2),R2 ;OFFSET VALUE
5061	100464	062702	100474	ADD	#10*,R2 ;ADD BASE VALUE
5062	100470	004712		JSR	PC,(R2) ;GO DO ACTION
5063	100472	000207		RTS	PC ;RETURN TO TRVACT
5064					
5065					
5066	100474	000122	101:	.WORD	ACTNUL-10# ;BRIEF DESCRIPTION OF ACTION TAKEN
5067	100476	000124		.WORD	ACTHLP-10# ;0-NULL
5068	100500	000162		.WORD	ACTNOO-10# ;1-HELP
5069	100502	000472		.WORD	ACTBLD-10# ;2-NODE
5070	100504	003542		.WORD	ACTRUN-10# ;3-BUILD
5071	100506	006336		.WORD	ACTPAT-10# ;4-RUN SPECIFIED TEST
5072	100510	010174		.WORD	ACTSAV-10# ;5-SET 'MESSAGE PATTERN' TEST FLAG
5073	100512	001214		.WORD	ACTSUM-10# ;6-SAVE NODE TABLE
5074	100514	001570		.WORD	ACTIDT-10# ;7-PRINT SUMMARY TABLE
5075	100516	002654		.WORD	ACTEXT-10# ;10-REQUEST ID
5076	100520	000114		.WORD	ACTNUF-10# ;11-EXIT
5077	100522	002664		.WORD	ACTXAD-10# ;12-NOT ENOUGH INFO
5078	100524	002746		.WORD	ACTSR4-10# ;13-EXTRACT NI NODE ADDRESS FROM INPUT LINE
5079	100526	007700		.WORD	ACTSND-10# ;14-SAVE POINTER TO BEGINING OF ADDRESS STRING
5080	100530	002754		.WORD	ACTALP-10# ;15-SET 'NODE' FLAG FOR SHOW COMMAND
5081	100532	002764		.WORD	ACTONE-10# ;16-SET 'ALPHA' FLAG
5082	100534	002774		.WORD	ACTZRO-10# ;17-SET 'ONES' FLAG
5083	100536	003004		.WORD	ACTIAL-10# ;20-SET 'ZEROS' FLAG
5084	100540	003014		.WORD	ACTOAL-10# ;21-SET 'IALT' FLAG
5085	100542	003024		.WORD	ACTCTT-10# ;22-SET 'OALT' FLAG
5086	100544	003034		.WORD	ACTOPR-10# ;27-SET 'CCIIT' FLAG
5087	100546	003162		.WORD	ACTTYP-10# ;24-SET 'OPER SEL' FLAG
5088	100550	003170		.WORD	ACTSZE-10# ;25-DETERMINE MESSAGE TYPE
5089	100552	003246		.WORD	ACTCPY-10# ;26-DETERMINE MESSAGE SIZE
5090	100554	003324		.WORD	ACTNAD-10# ;27-DETERMINE MESSAGE COPIES
5091	100556	003462		.WORD	ACTNAL-10# ;30-SET 'NODE/ADDRESS' FLAG
5092	100560	003662		.WORD	ACTRNA-10# ;31-SET 'NODE/ALL' FLAG
5093	100562	004730		.WORD	ACTRNL-10# ;32-SET 'ALL' FLAG FOR RUN COMMAND
5094	100564	006420		.WORD	ACTSMS-10# ;33-SET 'LOPPAIR' FLAG FOR RUN CMD
5095	100566	006512		.WORD	ACTCMS-10# ;34-SHOW CURRENT MESSAGE PARAMETERS
5096	100570	006616		.WORD	ACTCNT-10# ;35-RESET MESSAGE PARAMETERS TO DEFAULT
5097	100572	010054		.WORD	ACTCNL-10# ;36-SET 'COUNTER' FLAG FOR SHOW COMMAND
5098	100574	010140		.WORD	ACTFCT-10# ;37-CLEAR LOGICAL NODE NAMED FROM TABLE
5099	100576	010240		.WORD	ACTUNS-10# ;40-INITIATE UNA PORT COMMAND FUNCTION
5100	100600	010342		.WORD	ACTCSU-10# ;41-UNSAVE NODE TABLE
5101	100602	004302		.WORD	ACTDIR-10# ;42-CLEAR SUMMARY TABLE
5102	100604	010416		.WORD	ACTDFT-10# ;43-SET 'LCOP DIRECT' FLAG FOR RUN COMMAND
5103	100606	010464		.WORD	ACTUSF-10# ;44-LOOK FOR PASS COUNT DEFAULT
5104					;45-UNSAVE NODE TABLE FROM A FILE

```

5106
5107
5108 ;ACTION ROUTINE TO INDICATE THAT NOT ENOUGH COMMAND
5109 ;INFORMATION HAS BEEN ENTERED
5110 ;
5111
5112 100610 112737 177777 003160 ACTNUF: MOVB    0-1,P#NUF          ;SET FLAG TO SAY NEED MORE OF COMMAND
5113
5114 ;
5115 ;ACTION ROUTINE TO DO NOTHING
5116 ;
5117
5118 100616 000207 ACTNUL: RTS      PC          ;RETURN TO PARSER
5119
5120
5121 ;
5122 ;ACTION ROUTINE TO PRINT OUT HELP FILE
5123 ;
5124
5125 100620 ACTHLP: P#PUSH R2          ;SAVE R2
5126 100622 012702 003164      MOV    0#HLP1AB,R2      ;MOVE POINTER TO BEGINING OF HELP FILE
5127 100626 10%: PRINTF (R2)+ ;PRINT LINE AND INCREMENT POINTER
                    MOV    (R2)+,-(SP)
                    MOV    01,-(SP)
                    MOV    SP,R0
                    TRAP   C#PNTF
                    ADD    04,SP
                    100626 012706          ;
                    100630 012746 000001      MOV    (R2)+,-(SP)
                    100634 010600          MOV    01,-(SP)
                    100636 104417          MOV    SP,R0
                    100640 062706 000004      TRAP   C#PNTF
                    5128 100644 020227 003260      CMP    R2,0#HLPEND      ;SEE IF ENTIRE FILE PRINTED
                    5129 100650 001366          BNE    10%              ;IF NOT, PRINT MORE
                    5130 100652          P#POP R2                ;ELSE, RESTORE R2 AND RETURN
                    5131 100654 000207          RTS      PC
5132
5133
5134 ;
5135 ;ACTION ROUTINE TO READ IN NODE PHY. ADDRESS. STORE IT IN ADRBUF
5136 ;AND ENTER IT INTO THE NODE TABLE
5137 ;
5138
5139 100656 105037 003160 ACTNOD: CLRB    P#NUF          ;CLEAR NOTNUF FLAG
5140 100662 004737 076330      JSR    PC,TRVADR        ;TRAVERSE ADDRESS. CHECK IF TARGET OR ASSIST
5141 100666 105737 003161      TSTB  P#GDBD          ;CHECK IF RESULTS OK
5142 100672 001134          BNE    50%              ;IF NOT, RETURN WITH -1 IN P#GDBD
5143 100674 10%: CALL    EDPACK CBOADR,0#ADRBUF,06 ;GET ADDRESS INTO BUFFER
5144 100716          P#POP R1                ;CHECK RESULTS FOR NUMBER OF CHAR.S
5145 100720 001411          BEQ    15%              ;IF OK, BRANCH TO 15%
5146 100722          PRINTF 0#CADRER      ;ELSE PRINT ERROR MESSAGE
                    MOV    0#CADRER,-(SP)
                    MOV    01,-(SP)
                    MOV    SP,R0
                    TRAP   C#PNTF
                    ADD    04,SP
                    100722 012746 053232          MOV    0#CADRER,-(SP)
                    100726 012746 000001          MOV    01,-(SP)
                    100732 010600          MOV    SP,R0
                    100734 104417          TRAP   C#PNTF
                    100736 062706 000004          ADD    04,SP
                    5147 100742 000510          BR     50%              ;AND RETURN
                    5148 100744          15%: CALL    0#ADRBUF,0#ILLADR ;SEE IF ILLEGAL ADDRESS
                    5149 100762          P#POP R1
                    5150 100764 001021          BNE    17%              ;IF YES, PRINT ERROR MESSAGE
                    5151 100766          PRINTF 0#ILADMS
                    MOV    0#ILADMS,-(SP)

```

```

100772 012746 000001
100776 010600
101000 104417
101002 062706 000004
5152 101006          PRINTF  #ILADM1
101006 012746 053135
101012 012746 000001
101016 010600
101020 104417
101022 062706 000004
5153 101026 000456
5154 101030          17$:  BR      50$
5155          CALL    BINHEX #ADRBUF,#6,#STRBUF      ;CONVERT BINARY ADDRESS
101052 022737 000001 003672          ;INTO ASCII STRING
101060 001407          BEQ     20$      ;SEE IF TARGET OR ASSIST
101062 012737 062613 002312          MOV     #ARGTY7,KEYWD2      ;MOVE 'TARGET' INTO KEYWD2
101070 012737 000000 002400          MOV     #CTARGT,NODTY      ;MOVE TARGET INTO NODE TYPE
101076 000406          BR      25$
101100 012737 062603 002312 20$:  MOV     #ARGTY6,KEYWD2      ;MOVE 'ASSIST' INTO KEYWD2
101106 012737 000001 002400          MOV     #CASIST,NODTY
101114 101114          25$:  CALL    ENTRND          ;CALL ROUTINE TO ENTER NODE ON TABLE
101122 101122          P#POP  R1          ;CHECK RESULTS
101124 001017          BNE     50$      ;IF NODE TABLE FULL, RETURN
101126 012737 062510 002310          MOV     #CDTY7,KEYWD1      ;ELSE, MOVE "NODE" INTO KEYWD1
5167 101134          PRINTS #MSG2,KEYWD2,#STRBUF ;INDICATE IF TARGET OR ASSIST
101134 012746 002322          MOV     #STRBUF,-(SP)
101140 013746 002312          MOV     KEYWD2,-(SP)
101144 012746 055177          MOV     #MSG2,-(SP)
101150 012746 000003          MOV     #3,-(SP)
101154 010600          MOV     SP,RO
101156 104416          TRAP   C#PNTS
101160 062706 000010          ADD    #10,SP
5168 101164 000207          50$:  RTS     PC
5169
5170
5171
5172          ;
5173          ;ACTION ROUTINE TO BUILD NODE TABLE USING THE PERIODIC ADDRESS
5174          ;IDENTIFICATION MESSAGES SENT OUT BY NODE ON THE NI
5175          ;
5176 101166          ACTBLD: PRINTS #MSG1          ; PRINT 'BUILD' COMMAND MESSAGE
101166 012746 054711          MOV     #MSG1,-(SP)
101172 012746 000001          MOV     #1,-(SP)
101176 010600          MOV     SP,RO
101200 104416          TRAP   C#PNTS
101202 062706 000004          ADD    #4,SP
5177 101206          PRINTS #MSG11
101206 012746 055024          MOV     #MSG11,-(SP)
101212 012746 000001          MOV     #1,-(SP)
101216 010600          MOV     SP,RO
101220 104416          TRAP   C#PNTS
101222 062706 000004          ADD    #4,SP
5178 101226          PRINTS #MSG12
101226 012746 055137          MOV     #MSG12,-(SP)
101232 012746 000001          MOV     #1,-(SP)
101236 010600          MOV     SP,RO
101240 104416          TRAP   C#PNTS

```

Line	Address	OpCode	OpData	Comment	Macro
5179	101242	062706	000004		
5180	101246			P#PUSH R2,R3,R4	
5181	101254			CALL FUNCT #WDMULA	
5182	101266			P#POP R2	
5183	101270	001404		BEQ 10#	
5184	101272			ERRHRD 25,EMSG25	
	101272	104456			
	101274	000031			
	101276	065442			
	101300	000000			
5184	101302	005037	050550	10#:	CLR TEMP
5185	101306	005037	050552		CLR TEMP1
5186	101312	005037	050554		CLR TEMP2
5187	101316	012737	000012	050556	MOV #12,TEMP3
5188	101324	012737	002404	002402	MOV #NODTBL,SLOT
5189	101332	012704	003720		MOV #TIMERS,R4
5190	101336	012714	000074	19#:	MOV #60,(R4)
5191	101342			20#:	BREAK
	101342	104422			
5192	101344	005714			TST (R4)
5193	101346	001501			BEQ 40#
5194	101350				CALL RECEIVE
5195	101356				P#POP R2
5196	101360	001770			BEQ 20#
5197	101362	013703	003764		MOV RRGXNT,R3
5198	101366				CALL GETRNX #RRGXNT
5199	101400	016303	000002		MOV 2(R3),R3
5200	101404	062703	000006		ADD #SOURCC,R3
5201	101410	012702	002404		MOV #NODTBL,R2
5202	101414			21#:	CALL CMPADR R2,R3
5203	101426				P#POP R1
5204	101430	001744			BEQ 20#
5205	101432	062702	000010	22#:	ADD #10,R2
5206	101436	022712	177777		CMP #-1,(R2)
5207	101442	001364			BNE 21#
5208	101444				CALL FINDSL
5209	101452				P#POP R2
5210	101454	001023			BNE 35#
5211	101456	013702	002402		MOV SLOT,R2
5212	101462	012322			MOV (R3)+,(R2)+
5213	101464	012322			MOV (R3)+,(R2)+
5214	101466	011312			MOV (R3),(R2)
5215	101470	113762	050552	000003	MOVB TEMP1,3(R2)
5216	101476	105137	050552		COMB TEMP1
5217	101502	142737	000376	050552	BICB #376,TEMP1
5218	101510	005237	050550		INC TEMP
5219	101514	012737	000013	050556	MOV #13,TEMP3
5220	101522	000707			BR 20#
5221	101524			35#:	PRINTB #TABFUL,#NOD
	101524	012746	053701		
	101530	012746	053560		
	101534	012746	000002		
	101540	010600			
	101542	104414			
	101544	062706	000006		
5222	101550	000430			BR 50#
5223	101552	005337	050556	40#:	DEC TEMP3

```

; SAVE REGISTERS
; WRITE MULTICAST ADDRESS LIST
; CHECK FOR ERROR
; IF OK, CONTINUE
; ELSE REPORT ERROR

TRAP C#ERHRD
.WORD 25
.WORD EMSG25
.WORD 0

; CLEAR 'NO. NODES IN LAST MIN.' COUNTER
; CLEAR NODE TYPE ARGUMENT (SET TO TARGET)
; SET INTERVAL COUNTER
; SET 'MINS. SINCE LAST NEW NODE' COUNTER
; SET SLOT TO BEGINING OF NODE TABLE
; SET UP FOR 1 MINUTE LOOP

; ALLOW FOR CONTROL C INTERRUPTION
TRAP C#BRK

; SEE IF INTERVAL IS UP
; IF YES, BRANCH
; ELSE, CHECK FOR RECEPTION OF ID MESSAGE
; R2 HOLDS NO OF MESSAGES RECEIVED
; IF NONE, KEEP LOOKING
; IF ONE, GET RECEIVE RING POINTER
; UPDATE POINTER
; POINT R3 TO MESSAGE BUFFER
; POINT R3 TO NODE ADDRESS
; POINT R2 TO NODE TABLE
; SEE IF NODE ALREADY ON TABLE

; IF SAME, DON'T ADD TO TABLE
; ELSE CHECK NEXT TABLE ENTRY
; SEE IF AT END OF TABLE
; IF NO, COMPARE NEXT ENTRY
; IF YES, GET NEXT TABLE SLOT
; SEE IF TABLE FULL
; IF YES, BRANCH
; IF NO, ADD NODE TO TABLE

; SIX BYTES WORTH

; SET NODE TYPE (TARGET OR ASSIST)
; SWITCH TYPE FOR NEXT TIME

; INCREMENT 'NODES IN LAST MIN.' COUNTER
; RESET 'MINS. SINCE LAST NODE' COUNTER
; CHECK FOR MORE INPUT
; PRINT "TABLE FULL" MESSAGE
MOV #NOD,-(SP)
MOV #TABFUL,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C#PNTB
ADD #6,SP

; SEE IF 10 MINS SINCE LAST NODE

```

```

5224 101556 001425          BEQ      50$          ; IF YES, EXIT
5225 101560 005237 050554  INC      TEMP2       ; SEE IF TIME IS UP
5226 101564 023727 050554 000050  CMP      TEMP2, #40.
5227 101572 001417          BEQ      50$          ; IF YES, EXIT
5228 101574          PRINTS  #BLDMSG, TEMP, TEMP2 ; ELSE, PRINT "STILL WORKING" MESSAGE
      101574 013746 050554          MOV      TEMP2, -(SP)
      101600 013746 050550          MOV      TEMP, -(SP)
      101604 012746 052757          MOV      #BLDMSG, -(SP)
      101610 012746 000003          MOV      #3, -(SP)
      101614 010600          MOV      SP, R0
      101616 104416          TRAP    C#PNTS
      101620 062706 000010          ADD      #10, SP
5229 101624 005037 050550          CLR      TEMP
5230 101630 000642          BR       19$
5231 101632 012737 000000 050044 50$:  MOV      #0, #WDMC+4 ; CLEAR MULTICAST ADDRESS LIST
5232 101640          CALL    FUNCT #WDMULA ; WRITE 0 INTO LIST LENGTH
5233 101652          P#POP   R2           ; CHECK FOR ERROR
5234 101654 001404          BEQ      54$          ; IF OK CONTINUE
5235 101656          ERRHRD 34, EMSG25 ; ELSE, REPORT ERROR
      101656 104456          TRAP    C#ERHRD
      101660 000042          .WORD  34
      101662 055442          .WORD  EMSG25
      101664 000000          .WORD  0
5236 101666 004737 110374          JSR      PC, ACTSND ; PRINT NODE TABLE
5237 101672 012737 000400 050044 54$:  MOV      #400, #WDMC+4 ; RESET MULTICAST LIST FOR 1 ENTRY
5238 101700          P#POP   R2, R3, R4 ; RESTORE REGISTERS
5239 101706 000207          RTS      PC
5240
5241
5242
5243 ; ACTION ROUTINE TO PRINT OUT THE SUMMARY DATA
5244 ;
5245
5246 101710 105037 003160  ACTSUM: CLRB   P#NNUF ; CLEAR NOTNUF FLAG
5247 101714          P#PUSH R2, R3, R4
5248 101722 012701 002656  MOV      #STATBL, R1 ; MOVE ADDRESS OF TABLE TO R1
5249 101726 005711          TST     (R1)         ; SEE IF TABLE EMPTY
5250 101730 001013          BNE    5$           ; IF NOT, CONT.
5251 101732          PRINTF #TABEMT, #SUMM ; ELSE PRINT 'TABLE EMPTY' MESSAGE
      101732 012746 053706          MOV      #SUMM, -(SP)
      101736 012746 053632          MOV      #TABEMT, -(SP)
      101742 012746 000002          MOV      #2, -(SP)
      101746 010600          MOV      SP, R0
      101750 104417          TRAP    C#PNTF
      101752 062706 000006          ADD      #6, SP
5252 101756 000536          BR       30$          ; EXIT
5253 101760 021127 177777 5$:  CMP      (R1), #-1   ; SEE IF AT END OF TABLE
5254 101764 001533          BEQ     30$          ; IF YES, EXIT
5255 101756 005711          TST     (R1)         ; SEE IF REST OF TABLE EMPTY
5256 101770 001531          BEQ     30$          ; IF YES, EXIT
5257 101772          CALL   BINHEX R1, #6, #STRBUF ; PRINT SUMMARY DATA
5258 102012          PRINTF #SUMMS1, #STRBUF ; NODE ADDRESS
      102012 012746 002322          MOV      #STRBUF, -(SP)
      102016 012746 067215          MOV      #SUMMS1, -(SP)
      102022 012746 000002          MOV      #2, -(SP)
      102026 010600          MOV      SP, R0
      102030 104417          TRAP    C#PNTF
  
```

```

5259 102032 062706 000005
102036 PRINTF  #SUMMS2 ; FIRST HEADER ADD #6,SP
102036 012746 067236 MOV #SUMMS2,-(SP)
102042 012746 000001 MOV #1,-(SP)
102046 010600 MOV SP,R0
102050 104417 TRAP C#PNTF
102052 062706 000004 ADD #4,SP
5260 102056 016102 000006 MOV 6(R1),R2 ; RX NOT COMPLETE
5261 102062 016103 000010 MOV 10(R1),R3 ; RX COMPLETE
5262 102066 016104 000012 MOV 12(R1),R4 ; LENGTH ERRORS
5263 102072 PRINTF  #SUMMS3,R2,R3,R4 ; PRINT THEM OUT
102072 010446 MOV R4,-(SP)
102074 010346 MOV R3,-(SP)
102075 010246 MOV R2,-(SP)
102100 012746 067322 MOV #SUMMS3,-(SP)
102104 012746 000004 MOV #4,-(SP)
102110 010600 MOV SP,R0
102112 104417 TRAP C#PNTF
102114 062706 000012 ADD #12,SP
5264 102120 PRINTF  #SUMMS4 ; SECOND HEADER
102120 012746 067351 MOV #SUMMS4,-(SP)
102124 012746 000001 MOV #1,-(SP)
102130 010600 MOV SP,R0
102132 104417 TRAP C#PNTF
102134 062706 000004 ADD #4,SP
5265 102140 016102 000014 MOV 14(R1),R2 ; COMPARE ERRORS
5266 102144 062701 000016 ADD #16,R1 ; BYTES COMPARED
5267 102150 CALL BINDEC R1 ; PUT I' TO ASCII STRING
5268 102160 PRINTF  #SUMMS5,R2,#DECSTR ; PRINT THEM OUT
102160 012746 075200 MOV #DECSTR,-(SP)
102164 010246 MOV R2,-(SP)
102166 012746 067434 MOV #SUMMS5,-(SP)
102172 012746 000003 MOV #3,-(SP)
102176 010600 MOV SP,R0
102200 104417 TRAP C#PNTF
102202 062706 000010 ADD #10,SP
5269 102206 062701 000004 ADD #4,R1 ; BYTES TRANSFERED
5270 102212 CALL BINDEC R1 ; PUT INTO ASCII STRING
5271 102222 PRINTF  #SUMMS6,#DECSTR ; PRINT
102222 012746 075200 MOV #DECSTR,-(SP)
102226 012746 067452 MOV #SUMMS6,-(SP)
102232 012746 000002 MOV #2,-(SP)
102236 010600 MOV SP,R0
102240 104417 TRAP C#PNTF
102242 062706 000006 ADD #6,SP
5272 102246 062701 000004 ADD #4,R1 ; POINT R1 TO NEXT TABLE ENTRY
5273 102252 000642 BR 5# ; DO IT ALL AGAIN
5274 102254 30$: P#POP R2,R3,R4
5275 102262 000207 RTS PC
5276
5277
5278
5279 ; ACTION ROUTINE TO INITIATE THE REQUEST ID TEST TO THE SPECIFIED NODE
5280 ;
5281
5282
5283 ; FUNCTIONAL DESCRIPTION

```

```

5284 ; THIS SUBROUTINE BUILDS AND TRANSMITS REQUEST ID PACKETS
5285 ; TO THE NODE SPECIFIED BY THE OPERATOR IN THE COMMAND LINE.
5286 ; THE SYSTEM ID INFO OF THE SPECIFIED NODE IS THEN DISPLAYED.
5287 ; IF THE NODE DOES NOT RESPOND BEFORE 60 SECONDS HAVE PASSED
5288 ; AN ERROR IS REPORTED TO THE OPERATOR.
5289 ;
5290 ; INPUTS - IMPLICIT - THE SPECIFIED NODE ADDRESS IS LOCATED IN ADRBUF.
5291 ;
5292 ; OUTPUTS - SYSTEM ID INFO OR ERROR MESSAGE PRINTED TO OPERATOR.
5293 ;
5294 ; CALLING PROCEDURE - JSR PC, ACTIDT
5295 ;
5296 ; SIDE EFFECTS - XRGXNT POINTER IS UPDATED BY A CALL TO BLDREQ SUB.
5297 ;
5298 ; REGISTER USAGE - R1 POINTS TO $WDMO FOR WRITE MODE OPERATIONS.
5299 ; R2 IS SCRATCH.
5300 ; R3 POINTS TO THE RECEIVED MESSAGE BUFFER.
5301 ; R4 POINTS TO TIMEOUT TIMER
5302 ;
5303 ;---+
5304
5305 102264 105737 003162 ACTIDT: TSTB P$AERR ; SEE IF ADDRESS ENTERED WAS VALID
5306 102270 001402 BEQ 1$ ; IF NOT, EXIT ACTION ROUTINE
5307 102272 000137 103240 JMP 55$
5308 102276 105037 003160 1$: CLRB P$NNUF ; CLEAR NOTNUF FLAG
5309 102302 CALL CMPADR $ADRBUF,$ILLADR ; SEE IF ILLEGAL ADDRESS
5310 102320 P$POP R1
5311 102322 001012 BNE 2$ ; IF NO, CONTINUE
5312 102324 PRINTF $ILADMS ; ELSE PRINT ILLEGAL ADDRESS MESSAGE
102324 012746 053051 MOV $ILADMS,-(SP)
102330 012746 000001 MOV $1,-(SP)
102334 010600 MOV SP,R0
102336 104417 TRAP C$PNIF
102340 062706 000004 ADD $4,SP
5313 102344 000137 103240 JMP 55$
5314 102350 2$: P$PUSH R1,R2,R3,R4 ; SAVE REGISTERS
5315 102360 CALL CMPADR $ADRBUF,$PHYADR ; SEE IF ADDRESS IS OWN (HOST NODE)
5316 102376 P$POP R1
5317 102400 001535 BEQ 27$
5318 102402 012737 177776 050554 MOV $-2,$TEMP2 ; SET COUNTER FOR NO. OF TIMES TRICD
5319 102410 012701 050300 MOV $WDMO,R1 ; SET UP TO WRITE MODE
5320 102414 012761 010000 000002 MOV $10000,2(R1) ; 10000: TPAD =1 (PAD TRANSMIT BUFFERS)
5321 102422 CALL FUNCT $WDMODE ; WRITE MODE
5322 102434 P$POP R2 ; CHECK FOR ERROR
5323 102436 001402 BEQ 3$ ; BR IF ERROR
5324 102440 000137 103146 JMP 40$
5325 102444 3$: CALL BLDREQ ; BUILD REQUEST ID MESSAGE PACKET
5326 102452 CALL XMIT ; TRANSMIT REQUEST
5327 102460 P$POP R2 ; GET RESULTS, R2 = SUCCESS/FAILURE
5328 102462 001402 BEQ 4$ ; IF OK BRANCH
5329 102464 000137 103160 JMP 45$ ; ELSE JUMP TO 45$
5330 102470 005737 050534 4$: TST RETRYS ; SEE IF FAILED DUE TO EXCESSIVE COLLISIONS
5331 102474 001412 BEQ 5$ ; IF NO, CONT.
5332 102476 PRINTF $RTRYER ; YES, PRINT 'EXCESSIVE COLLISIONS' MESSAGE
102476 012746 052675 MOV $RTRYER,-(SP)
102502 012746 000001 MOV $1,-(SP)
102506 010600 MOV SP,R0

```


102510	104417							TRAP	C\$PNTF
5333 102512	062706	000004						ADD	#4,SP
5334 102516	000137	103206		5\$:	JMP	52\$:	EXIT
5335 102522	012704	003720			MOV	#TIMERS,R4		:	SET UP FOR 10 SECOND TIMEOUT
5336 102526	012714	000012			MOV	#10.,(R4)			
5337 102532	104422			15\$:	BREAK			TRAP	C\$BRK
5338 102534	005714				TST	(R4)		:	SEE IF TIME HAS EXPIRED
5339 102536	001425				BEQ	20\$:	IF YES, BRANCH
5340 102540					CALL	RECEVE		:	CHECK FOR ANSWER
5341 102546					P\$POP	R2		:	R2 HOLDS NO. OF BUFFERS RECEIVED
5342 102550	001770				BEQ	15\$:	IF NO BUFFERS RECIEVED, LOOP
5343 102552	013703	003764			MOV	RRGNXT,R3		:	GET RECEIVE RING POINTER
5344 102556					CALL	GETRNX #RRGNXT		:	UPDATE POINTER
5345 102570	016303	000002			MOV	2(R3),R3		:	POINT R3 TO MESSAGE BUFFER
5346 102574	026327	000022	051115		CMP	SIRCPT(R3),#MR		:	SEE IF MESSAGE RECIEVED IS IN REPLY TO ONE SENT
5347 102602	001424				BEQ	25\$:	IF YES, BRANCH TO 25\$
5348 102604	005237	050554			INC	TEMP2		:	INCREMENT RETRY COUNTER
5349 102610	001350				BNE	15\$:	IF NO, LOOK FOR CORRECT REPLY MESSAGE
5350 102612				20\$:	PRINTF	#EMSG22		:	ELSE, REPORT ERROR
102612	012746	065266						MOV	#EMSG22,-(SP)
102616	012746	000001						MOV	#1,-(SP)
102622	010600							MOV	SP,R0
102624	104417							TRAP	C\$PNTF
102626	062706	000004						ADD	#4,SP
5351 102632	005237	050500			INC	S.NFIEC		:	UPDATE SUMMARY DATA
5352 102636	000137	103170			JMP	51\$:	AND EXIT
5353									
5354 102642	012703	050413		22\$:	MOV	#UCB22+13,R3		:	POINT R3 TO SAME PLACE IN MESSAGE AS
5355 102646	012702	050454			MOV	#UCB22+54,R2		:	IF IT WERE RECEIVED SYS ID, R2 TO
5356 102652	000410				BR	27\$:	NODE ADDRESS
5357									
5358 102654	005237	050476		25\$:	INC	S.REC		:	INCREMENT 'RECEIVED MESSAGES' COUNTER
5359 102660	062737	000056	050510		ADD	#46.,S.XFER		:	UPDATE 'BYTES TRANSFERED' COUNTER
5360 102666	010302				MOV	R3,R2		:	PUT POINTER INTO R2
5361 102670	062702	000042			ADD	#SIADDR,R2		:	POINT R2 TO ADDRESS
5362 102674				27\$:	CALL	BINHEX R2,#6,#STRBUF		:	PUT ADDRESS INTO STRBUF
5363 102714					PRINTF	#SIMSG1,#STRBUF		:	PRINT REMOTE NODE ADDRESS
102714	012746	002322						MOV	#STRBUF,-(SP)
102720	012746	066440						MOV	#SIMSG1,-(SP)
102724	012746	000002						MOV	#2,-(SP)
102730	010600							MOV	SP,R0
102732	104417							TRAP	C\$PNTF
102734	062706	000006						ADD	#6,SP
5364 102740	016302	000022			MOV	SIRCPT(R3),R2		:	GET RECEIPT NUMBER
5365 102744					PRINTF	#SIMSG2,R2		:	PRINT RECEIPT NUMBER
102744	010246							MOV	R2,-(SP)
102746	012746	066477						MOV	#SIMSG2,-(SP)
102752	012746	000002						MOV	#2,-(SP)
102756	010600							MOV	SP,R0
102760	104417							TRAP	C\$PNTF
102762	062706	000006						ADD	#6,SP
5366 102766	116302	000027			MOVB	SIVERS(R3),R2		:	GET VERSION NO. AND PRINT
5367 102772					PRINTF	#SIMSG3,R2			
102772	010246							MOV	R2,-(SP)
102774	012746	066532						MOV	#SIMSG3,-(SP)

	103000	012746	000002						
	103004	010600							MOV #2,-(SP)
	103006	104417							MOV SP,R0
	103010	062706	000006						TRAP C\$PNTF
5368	103014	116302	000030						ADD #6,SP
5369	103020			MOVB	SIECO(R3),R2		; GET ECO NO. AND PRINT		
	103020	010246		PRINTF	#SIMSG4,R2				
	103022	012746	066572						MOV R2,-(SP)
	103026	012746	000002						MOV #SIMSG4,-(SP)
	103032	010600							MOV #2,-(SP)
	103034	104417							MOV SP,R0
	103036	062706	000006						TRAP C\$PNTF
5370	103042	116302	000031						ADD #6,SP
5371	103046			MOVB	SIUECO(R3),R2		; GET USER ECO NO. AND PRINT		
	103046	010246		PRINTF	#SIMSG5,R2				
	103050	012746	066613						MOV R2,-(SP)
	103054	012746	000002						MOV #SIMSG5,-(SP)
	103060	010600							MOV #2,-(SP)
	103062	104417							MOV SP,R0
	103064	062706	000006						TRAP C\$PNTF
5372	103070	116302	000035						ADD #6,SP
5373	103074			MOVB	SIFNCT(R3),R2		; GET FUNCTON CODE AND PRINT		
	103074	010246		PRINTF	#SIMSG6,R2				
	103076	012746	066641						MOV R2,-(SP)
	103102	012746	000002						MOV #SIMSG6,-(SP)
	103106	010600							MOV #2,-(SP)
	103110	104417							MOV SP,R0
	103112	062706	000006						TRAP C\$PNTF
5374	103116	116302	000053						ADD #6,SP
5375	103122			MOVB	SIDEV(R3),R2		; GET DEVICE TYPE AND PRINT		
	103122	010246		PRINTF	#SIMSG7,R2				
	103124	012746	066667						MOV R2,-(SP)
	103130	012746	000002						MOV #SIMSG7,-(SP)
	103134	010600							MOV #2,-(SP)
	103136	104417							MOV SP,R0
	103140	062706	000006						TRAP C\$PNTF
5376	103144	000411							ADD #6,SP
5377				BR	51\$; EXIT		
5378	103146								
	103146	104455		40\$:	ERRDF	23,EMSG23,ERR1	; ERROR -- CAN'T WRITE MODE		
	103150	000027							TRAP C\$ERDF
	103152	065321							.WORD 23
	103154	067460							.WORD EMSG23
5379	103156	000424							.WORD ERR1
5380				BR	54\$				
5381	103160								
	103160	104455		45\$:	ERRDF	24,EMSG24,ERR1	; ERROR -- CAN'T TRANSMIT PACKETS		
	103162	000030							TRAP C\$ERDF
	103164	065366							.WORD 24
	103166	067460							.WORD EMSG24
5382									.WORD ERR1
5383	103170			51\$:	CALL	WRITES #1,#ADRBUF	; UPDATE SUMMARY TABLE		
5384	103206	005061	000002	52\$:	CLR	2(R1)	; DISABLE TRANSMIT PADDING		
5385	103212				CALL	FUNCT #WDMODE			
5386	103224				P\$POP	R2	; CHECK FOR ERROR		
5387	103226	001347			BNE	40\$; IF ONE, REPORT IT		
5388	103230			54\$:	P\$POP	R1,R2,R3,R4	; RESTORE REGISTERS		

```

5389 103240 000207      55$:   RTS      PC
5390
5391
5392
5393      ; ACTION ROUTINE TO CHECK FOR ADDITION PARAMETER CHANGE INPUTS
5394      ; AND PRINT OUT NEW PARAMETER INFO WHEN ALL INPUT ARE PROCESSED
5395      ;
5396
5397 103242 105714      ACTMSG: TSTB      (R4)      ;CHECK FOR ADDITIONAL INPUT
5398 103244 001401      BEQ      12$      ;BR IF NO
5399 103246 000437      BR      50$      ;IF YES RETURN
5400 103250 012737 062500 002310 12$:   MOV      #CMDTY6,KEYWD1
5401 103256 013701 002370      MOV      P$TYPE,R1      ;GET MESSAGE TYPE ASCII STRING ADDRESS
5402 103262 006301      ASL      R1      ;INTO R1
5403 103264 062701 003262      ADD      #MSGTAB,R1
5404 103270      PRINTF  #MSGPRM      ;PRINT 'MESSAGE' COMMAND MESSAGE
      MOV      #MSGPRM,-(SP)
      MOV      #1,-(SP)
      MOV      SP,RO
      TRAP    C$PNTF
      ADD      #4,SP
      MOV      P$CPYS,-(SP)
      MOV      P$SIZE,-(SP)
      MOV      (R1),-(SP)
      MOV      #MSG4,-(SP)
      MOV      #4,-(SP)
      MOV      SP,RO
      TRAP    C$PNTF
      ADD      #12,SP
      PRINTF  #MSG4,(R1),P$SIZE,P$CPYS      ;PRINT MSG PARAMETERS
5405 103310      PRINTF  #MSG4,(R1),P$SIZE,P$CPYS      ;PRINT MSG PARAMETERS
      MOV      P$CPYS,-(SP)
      MOV      P$SIZE,-(SP)
      MOV      (R1),-(SP)
      MOV      #MSG4,-(SP)
      MOV      #4,-(SP)
      MOV      SP,RO
      TRAP    C$PNTF
      ADD      #12,SP
      MOV      #12,SP
5406 103342 105037 003160      50$:   CLRB     P$NNUF      ;CLEAR NOTNUF FLAG
5407 103346 000207      RTS      PC
5408
5409
5410
5411      ; ACTION ROUTINE TO RETURN CONTROL TO THE SUPERVISOR
5412      ;
5413
5414 103350 012737 000020 003672 ACTEXT: MOV      #CEXIT,CFLAG      ;SET EXIT FLAG
5415 103356 000207      RTS      PC
5416
5417
5418
5419      ; ACTION ROUTINE TO TAKE NI NODE ADDRESS FROM INPUT STRING BUFFER
5420      ; AND STORE IT IN THE BUFFER CALLED ADRBUF
5421      ;
5422
5423 103360      ACTXAD: CALL    EDPACK CBOADR,#ADRBUF,#6      ;PUT NODE ADDRESS INTO ADRBUF
5424 103402      P$POP    P$AERR      ;SET ADDRESS=12 CHAR. GOOD/BAD FLAG
5425 103406 105737 003162      TSTB     P$AERR      ;IF GOOD, RETURN
5426 103412 001412      BEQ      10$
5427 103414      PRINTF  #CADRER      ;ELSE, PRINT ERROR MESSAGE
      MOV      #CADRER,-(SP)
      MOV      #1,-(SP)
      MOV      SP,RO
      TRAP    C$PNTF
      ADD      #4,SP
103414 012746 053232
103420 012746 000001
103424 010600
103426 104417
103430 062706 000004
  
```

```

5428 103434 105037 003160
5429 103440 000207          10$: CLRB   P$NNUF      ; AND CLEAR 'NOT ENOUGH' FLAG
5430          RTS   PC
5431
5432          ; ACTION ROUTINE TO STORE POINTER TO BEGINING OF OPERATOR INPUT ADDRESS
5433          ; IN COMMAND INPUT BUFFER
5434          ;
5435
5436 103442 010437 002366 ACTSR4: MOV   R4,CBOADR      ;SAVE STRING POINTER
5437 103446 000207          10$:  RTS   PC
5438
5439
5440          ;
5441          ; ACTION ROUTINE TO SET MESSAGE TYPE = ALPHA FLAG
5442          ;
5443
5444 103450 012737 000000 002370 ACTALP: MOV   $ALPHA,P$TYPE  ;SET MESSAGE TYPE
5445 103456 000207          RTS   PC
5446
5447
5448          ;
5449          ; ACTION ROUTINE TO SET MESSAGE TYPE = ALL ONES FLAG
5450          ;
5451
5452 103460 012737 000001 002370 ACTONE: MOV   $ONES,P$TYPE   ;SET MESSAGE TYPE
5453 103466 000207          RTS   PC
5454
5455
5456          ;
5457          ; ACTION ROUTINE TO SET MESSAGE TYPE = ALL ZEROS FLAG
5458          ;
5459
5460
5461 103470 012737 000002 002370 ACTZRO: MOV   $ZEROS,P$TYPE  ;SET MESSAGE TYPE
5462 103476 000207          RTS   PC
5463
5464
5465          ;
5466          ; ACTION ROUTINE TO SET MESSAGE TYPE = ALTERNATING ONES FLAG
5467          ;
5468
5469 103500 012737 000003 002370 ACT1AL: MOV   $ONEALT,P$TYPE ;SET MESSAGE TYPE
5470 103506 000207          RTS   PC
5471
5472
5473          ;
5474          ; ACTION ROUTINE TO SET MESSAGE TYPE = ALTERNATING ZEROS FLAG
5475          ;
5476
5477 103510 012737 000004 002370 ACTOAL: MOV   $ZROALT,P$TYPE ;SET MESSAGE TYPE
5478 103516 000207          RTS   PC
5479
5480
5481          ;
5482          ; ACTION ROUTINE TO SET MESSAGE TYPE = CCITT FLAG
5483          ;
5484

```

```

5485 103520 012737 000005 002370 ACTCTT: MOV      #CCITT,P#TYPE      ;SET MESSAGE TYPE
5486 103526 000207          R'S      PC
5487
5488
5489
5490          ; ACTION ROUTINE TO SET MESSAGE TYPE * OPERATOR SELECTED INPUT
5491          ;
5492
5493 103530 105037 003163 ACTOPR: CLRB    P#MERR          ;CLEAR MESSAGE ERROR FLAG
5494 103534 004737 076330      JSR      PC,TRVADR        ;PARSE THROUGH INPUT STRING
5495 103540 105737 003161      TSTB    P#GDBD          ;TEST GOOD/BAD FLAG
5496 103544 001403          BEQ      10#             ;IF GOOD, BR 10#
5497 103546 105037 003161      CLRB    P#GDBD          ;CLEAR FLAG
5498 103552 000415          BR       20#            ;SET CTARGET FLAG AND RETURN
5499 103554 022737 000006 003672 10#:  CMP     #OPRSEL,CFLAG    ;CHECK TO SEE IF STRING VALID
5500 103562 001011          BNE     20#             ;IF NOT OK, RETURN WITH ERROR FLAG SET
5501 103564 012737 000006 002370      MOV     #OPRSEL,P#TYPE  ;SET MESSAGE TYPE
5502 103572          CALL    SELMSG CBOADR   ;PUT OPERATOR SELECTED STRING INTO BUFFER
5503 103604 000423          BR       50#            ;RETURN
5504 103606 022737 000000 003672 20#:  CMP     #CTARGET,CFLAG  ;SEE IF CTARGET FLAG SET. IF YES ERROR
5505 103614 001011          BNE     30#             ;IF NOT SET, BR 30#
5506 103616          PRINTF #UNBOND        ;ELSE PRINT UNBOUNDED INPUT STRING ERR. MSG.
          103616 012746 054252          MOV     #UNBOND,-(SP)
          103622 012746 000001          MOV     #1,-(SP)
          103626 010600          MOV     SP,RO
          103630 104417          TRAP   C#PRINTF
          103632 062706 000004          ADD     #4,SP
5507 103636 000406          BR       50#            ;RETURN
5508 103640 105737 003163 30#:  TSTB    P#MERR          ;IF P#MERR FLAG SET, UNBOUNDED STRING
5509 103644 001003          BNE     50#             ;WAS ENTERED, ERROR ALREADY HANDLED
5510 103646 112737 177777 003161      MOVB   #1,P#GDBD       ;SET ERROR FLAG AND RETURN
5511 103654 000207          RTS      PC             ;RETURN
5512
5513
5514
5515          ; ACTION ROUTINE TO CHECK FOR MORE INPUT AFTER MESSAGE TYPE HAS BEEN
5516          ;ALTERED
5517          ;
5518
5519 103656 004737 103242 ACTTYP: JSR      PC,ACTMSG    ;CHECK FOR ADDITIONAL COMMANDS
5520 103662 000207          RTS      PC
5521
5522
5523
5524          ; ACTION ROUTINE TO INPUT MESSAGE SIZE PARAMETER, CHECK TO SEE IF
5525          ;IT IS WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO
5526          ;SEE IF MORE INPUT EXISTS
5527          ;
5528
5529 103664 023727 003154 000037 ACTSIZE: CMP     P#NUM,#31     ;CHECK FOR VALID SIZE RANGE
5530 103672 003410          BLE     10#
5531 103674 022737 002673 003154      CMP     #1467.,P#NUM
5532 103702 003404          BLE     10#             ;IF VALID CONTINUE
5533 103704 013737 003154 002372      MOV     P#NUM,P#SIZE   ;SET MESSAGE SIZE
5534 103712 000410          BR       20#
5535 103714          PRINTF #SIZLMT      ;PRINT SIZE LIMITS EXCEEDED MESSAGE
          103714 012746 054061          MOV     #SIZLMT,-(SP)

```

```

103720 012746 000001
103724 010600
103726 104417
103730 062706 000004
5536 103734 004737 103242      201: JSR PC,ACTMSG
5537 103740 000207              RTS PC
5538
5539
5540
5541
5542
5543
5544
5545
5546 103742 023727 003154 000000 ACTCPY: CMP P#NUM,#0
5547 103750 003410              BLE 101
5548 103752 022737 000400 003154      CMP #256.,P#NUM
5549 103760 003404              BLE 101
5550 103762 013737 003154 002374      MOV P#NUM,P#CPYS
5551 103770 000410              BR 201
5552 103772
103772 012746 053775      101: PRINTF #CPYLMT
103776 012746 000001
104002 010600
104004 104417
104006 062706 000004
5553 104012 004737 103242      201: JSR PC,ACTMSG
5554 104016 000207              RTS PC
5555
5556
5557
5558
5559
5560
5561 104020 105037 003160
5562 104024 105737 003162
5563 104030 001051
5564 104032
5565 104036 012702 002314
5566 104042 012703 002404
5567 104046
5568 104060
5569 104062 001416
5570 104064 062703 000010
5571 104070 022713 177777
5572 104074 001364
5573 104076
104076 012746 054160
104102 012746 000001
104106 010600
104110 104417
104112 062706 000004
5574 104116 000414
5575 104120 005023
5576 104122 005023
5577 104124 005023
5578 104126 005013

;ACTION ROUTINE TO INPUT COPIES PARAMETER, CHECK TO SEE IF IT IS
;WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO SEE IF
;MORE INPUT PARAMETERS EXIST
;
;ACTION ROUTINE TO CLEAR NODE SPECIFIED BY PHYSICAL ADDRESS FROM NODE TABLE
;
;CHECK FOR VALID COPIES RANGE
;IF VALID, CONTINUE
;SET MESSAGE COPIES
;PRINT COPY LIMIT EXCEEDED MESSAGE
MOV #CPYLMT,-(SP)
MOV #1,-(SP)
MOV SP,RO
TRAP C#PNTF
ADD #4,SP
;CHECK FOR ADDITIONAL COMMANDS
;
;CLEAR NOTNUF FLAG
;SEE IF ADDRESS ENTERED WAS VALID
;IF NOT, EXIT ACTION ROUTINE
;SAVE R2 AND R3
;MOVE ADDRESS OF ADDRESS INTO R2
;MOVE ADDRESS OF NODE TABLE INTO R3
;SEE IF ADDRESSES MATCH
;IF YES, BR 251
;ELSE POINT R3 TO NEXT ENTRY
;SEE IF END OF TABLE
;IF NOT, COMPARE NEXT ENTRY
;ELSE, PRINT ADDRESS DOESN'T COMPARE MSG.
MOV #NOCHPR,-(SP)
MOV #1,-(SP)
MOV SP,RO
TRAP C#PNTF
ADD #4,SP
;RETURN
;ELSE, CLEAR NODE FROM TABLE
251: BR 301
CLR (R3)
CLR (R3)
CLR (R3)
CLR (R3)

```

```

5579 104130          PRINTF  #ADRDEL          ;PRINT NODE DELETED FROM TABLE MESSAGE
      104130 012746 054340          MOV      #ADRDEL,-(SP)
      104134 012746 000001          MOV      #1,-(SP)
      104140 010600          MOV      SP,R0
      104142 104417          TRAP    C#PNTF
      104144 062706 000004          ADD     #4,SP
5580 104150          30#: P#POP  R2,R3          ;RESTORE R2 AND R3
5581 104154 000207          35#: RTS    PC          ;RETURN
5582
5583
5584
5585          ; ACTION ROUTINE TO CLEAR NODE TABLE
5586          ;
5587
5588 104156          ACTNAL: P#PUSH R2,R3          ;SAVE R2,R3
5589 104162 012703 000050          MOV     #TBLEN,R3          ;SET INCR. COUNTER TO 40
5590 104166 012702 002404          MOV     #NODTBL,R2        ;MOVE NODE TABLE ADDRESS INTO R2
5591 104172 005022          10#: CLR   R2)          ;CLEAR BYTE IN NODE TABLE
5592 104174 005303          DEC     3                ;DECREMENT COUNTER
5593 104176 001375          BNE    0                 ;CONTINUE UNTIL DONE
5594 104200          PRINTF  #TABCLR,#NOD          ;PRINT NODE TABLE CLEARED MESSAGE
      104200 012746 053701          MOV     #NOD,-(SP)
      104204 012746 054514          MOV     #TABCLR,-(SP)
      104210 012746 000002          MOV     #2,-(SP)
      104214 010600          MOV     SP,R0
      104216 104417          TRAP    C#PNTF
      104220 062706 000006          ADD     #6,SP
5595 104224 105037 003160          CLR    P#NUF             ;CLEAR NOTNUF FLAG
5596 104230          P#POP  R2,R3          ;RESTORE R2 AND R3
5597 104234 000207          RTS    PC
5598
5599
5600
5601          ; ACTION ROUTINE TO RUN SPECIFIED TEST
5602          ;
5603
5604 104236 105037 003160          ACTRUN: CLR    P#NUF             ; CLEAR 'NOT ENOUGH' FLAG
5605 104242 013737 003154 002376          MOV     P#NUM,P#PASS
5606 104250 022737 000032 002310          5#: CMP     #CRNALL,KEYWD1          ; SEE IF 'ALL' TEST
5607 104256 001004          BNE    10#              ; IF NO, CONTINUE
5608 104260          CALL   RUNALL          ; IF YES, DO ALLNODE
5609 104266 000423          BR     30#
5610 104270 022737 000033 002310          10#: CMP     #CLUPPR,KEYWD1          ; IS IT 'LOOPPAIR' TEST
5611 104276 001004          BNE    15#              ; IF NO, CONTINUE
5612 104300          CALL   RUNLUP          ; IF YES, DO LOOPPAIR
5613 104306 000413          BR     30#
5614 104310 022737 000043 002310          15#: CMP     #CDIR,KEYWD1          ; IS IT 'DIRECT' TEST
5615 104316 001004          BNE    20#              ; IF NO, CONTINUE
5616 104320          CALL   RUNDIR          ; IF YES, DO DIRECT
5617 104326 000403          BR     30#
5618 104330          CALL   RUNPAT          ; ELSE, ITS 'PATTERN' TEST
5619 104336 023727 002376 17777          30#: CMP     P#PASS,#-1          ; SEE IF PASS SET FOR INDEFINATE
5620 104344 001741          BEQ    5#                ; IF YES, LOOP
5621 104346 005337 002376          DEC     P#PASS          ; HAVE WE DONE ALL PASSES?
5622 104352 001336          BNE    5#                ; IF NO, LOOP
5623 104354 000207          RTS    PC
5624

```

```

5625
5626 ; ACTION ROUTINE TO SET 'RUN ALL' FLAG
5627 ;
5628
5629 104356 012737 000032 002310 ACTRNA: MOV #CRNALL,KEYWD1 ; SET FLAG
5630 104364 000207 RTS PC
5631
5632 104366 RUNALL: CALL DIRCOM ; RUN LOOP/DIRECT TEST
5633 104374 P#POP R1 ; CHECK RESULTS
5634 104376 001415 BEQ 5# ; IF OK, BRANCH
5635 104400 022701 000001 CMP #1,R1 ; ELSE, WAS TABLE EMPTY?
5636 104404 001410 BEQ 3# ; IF YES, DON'T PRINT ABORT MESSAGE
5637 104406 PRINTS #PASABT ; ELSE ABORT TEST AND PRINT MESSAGE
104406 012746 061526 MOV #PASABT,-(SP)
104412 012746 000001 MOV #1,-(SP)
104416 010600 MOV SP,RO
104420 104416 TRAP C#PNTS
104422 062706 000004 ADD #4,SP
5638 104426 000137 104774 3#: JMP 32#
5639 104432 012737 002404 002402 5#: MOV #NODTBL,SLOT ; MOVE NODE TABLE ADDRESS TO SLOT
5640 104440 CALL FULSLT ; FIND FIRST ENTRY
5641 104446 013701 002402 MOV SLOT,R1 ; AND PUT TARGET ADDRESS INTO R1
5642 104452 013737 002374 050562 10#: MOV P#CPYS,CPYCN1 ; SET UP LOOP FOR NO. OF COPIES
5643 104460 062737 000010 002402 ADD #10,SLOT ; UPDATE SLOT
5644 104466 CALL FULSLT ; GET NEXT ASSIST NODE FROM TABLE
5645 104474 013702 002402 MOV SLOT,R2
5646 104500 022737 177777 002402 CMP #-1,SLOT ; SEE IF AT END OF TABLE
5647 104506 001515 BEQ 25# ; IF YES, BR
5648 104510 15#: CALL BLDFAS R1,SLOT ; BUILD FULL ASSIST MESSAGE
5649 104524 CALL XMIT ; TRANSMIT MESSAGE
5650 104532 P#POP R3 ; CHECK RESULTS
5651 104534 001404 BEQ 17# ; IF OK, CONTINUE
5652 104536 ERRHRD 37,MSG24 ; PRINT ERROR MESSAGE
104536 104456 TRAP C#ERRHRD
104540 000045 .WORD 37
104542 065366 .WORD MSG24
104544 000000 .WORD 0
5653 104546 17#: CALL BINHEX R1,#6,#STRBUF ; PRINT ERROR MESSAGE
5654 104566 CALL BINHEX R2,#6,#STRBU1 ;
5655 104606 PRINTB #TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1 ; ASSIST NODE
104606 012746 002344 MOV #STRBU1,-(SP)
104612 012746 062603 MOV #ARGTY6,-(SP)
104616 012746 002322 MOV #STRBUF,-(SP)
104622 012746 062613 MOV #ARGTY7,-(SP)
104626 012746 061624 MOV #TSTMS4,-(SP)
104632 012746 000005 MOV #5,-(SP)
104636 010600 MOV SP,RO
104640 104414 TRAP C#PNTB
104642 062706 000014 ADD #14,SP
5656 104646 CALL RUNCOM ; DO RECEIVE LOOP
5657 104654 P#POP R4 ; CHECK RESULTS
5658 104656 001405 BEQ 21# ; IF OK, LOOP SOME MORE
5659 104660 20#: ERRHRD 28,MSG42,ERR3 ; ELSE PRINT ERROR MESSAGE
104660 104456 TRAP C#ERRHRD
104662 000034 .WORD 28
104664 066353 .WORD MSG42
104666 067604 .WORD ERR3

```



```

5660 104670 000410
5661 104672 21$: BR 101$
      104672 012746 062013 PRINTB #OKFU
      104676 012746 000001
      104702 010600
      104704 104414
      104706 062706 000004
5662 104712 005337 050562 101$: DEC CPYCNT ; DECREMENT 'COPIES' COUNTER
5663 104716 001274 BNE 15$ ; IF MORE TO DO, LOOP
5664 104720 CALL WRITES #2,R1,SLOT ; ELSE, UPDATE SUMMARY TABLE
5665 104740 000644 BR 10$
5666 104742 062701 000010 25$: ADD #10,R1 ; POINT R1 TO NEXT TARGET NODE
5667 104746 010137 002402 MOV R1,SLOT ; UPDATE SLOT
5668 104752 CALL FULSLT ; GET ADDRESS FROM TABLE
5669 104760 013701 002402 MOV SLOT,R1
5670 104764 022737 177777 002402 CMP #-1,SLOT ; SEE IF END OF TABLE
5671 104772 001227 BNE 10$ ; IF NO, CONTINUE ELSE, FINISHED
5672 104774 32$: RETURN
5673
5674 ;
5675 ;ACTION ROUTINE TO SET 'RUN LOOP DIRECT' FLAG
5676 ;
5677
5678 104776 012737 000043 002310 ACTDIR: MOV #CDIR,KEYWD1 ; SET FLAG
5679 105004 000207 RTS PC
5680
5681 105006 RUNDIR: CALL DIRCOM ; CALL COMMON CODE
5682 105014 P#POP R1
5683 105016 022701 000001 CMP #1,R1 ; WAS TABLE EMPTY?
5684 105022 001400 BEQ 10$ ; IF YES, DON'T PRINT
5685 105024 10$: RETURN
5686
5687 105026 005001 DIRCOM: CLR R1 ; CLEAR RESULTS REGISTER
5688 105030 012737 002404 002402 MOV #NODTBL,SLOT ; MOVE NODE TABLE ADDRESS TO SLOT
5689 105036 CALL FULSLT ; SEE IF TABLE EMPTY
5690 105044 022737 177777 002402 CMP #-1,SLOT
5691 105052 001015 BNE 9$ ; IF NO CONTINUE
5692 105054 PRINTF #TABEMT,#NCD ; ELSE, PRINT "TABLE EMPTY" MESSAGE
      105054 012746 053701
      105060 012746 053632
      105064 012746 000002
      105070 010600
      105072 104417
      105074 062706 000006
5693 105100 012701 000001 MOV #1,R1 ; PUT 'TABLE EMPTY' INDICATOR IN R1
5694 105104 000545 BR 32$
5695 105106 012737 002404 002402 9$: MOV #NODTBL,SLOT
5696 105114 013737 002574 050562 10$: MOV #CPYS,CPYCNT ; SET UP FOR NO. OF COPIES
5697 105122 CALL FULSLT ; GET NEXT NODE IN TABLE
5698 105130 022737 177777 002402 CMP #-1,SLOT ; SEE IF AT END OF TABLE
5699 105136 001530 BEQ 32$ ; IF YES, EXIT
5700 105140 CALL BINEX SLOT,#6,#STRBUF ; PRINT ADDRESS BEING TESTED
5701 105162 15$: PRINTB #TSTMS2,#DIRECT,#STRBUF ; NODE ADDRESS
      105162 012746 002322
      105166 012746 062173
      105172 012746 061571
      105176 012746 000003
      MOV #STRBUF,-(SP)
      MOV #DIRECT,-(SP)
      MOV #TSTMS2,-(SP)
      MOV #3,-(SP)
  
```

```

105202 010600
105204 104414
105206 062706 000010
5702 105212 022737 000005 002310      CMP      #CPATRN,KEYWD1
5703 105220 001016                      BNE      16$
5704 105222 013701 002370      MOV      P$TYPE,R1
5705 105226 006301                      ASL      R1
5706 105230 062701 003262      ADD      #MSGTAB,R1
5707 105234                      PRINTB   #MESPAL,(R1)
105234 011146
105236 012746 062125
105242 012746 000002
105246 010600
105250 104414
105252 062706 000006
5708 105256          16$:      CALL     BLDLD  SLOT      ; CALL BUILD LOOPDIRECT SUBROUTINE
5709 105270          CALL     XMIT                    ; TRANSMIT LOOPDIRECT MESSAGES
5710 105276          P$POP   R2                    ; GET RESULTS, R2 = SUCCESS/FAILURE
5711 105300 001405          BEQ     26$                    ; IF OK, EXIT
5712 105302          25$:      ERRHRD  26,EMSG24      ; ELSE PRINT ERROR MESSAGE
105302 104456
105304 000032
105306 065366
105310 000000
5713 105312 000700
5714 105314          26$:      BR       10$
5715 105322          CALL     RUNCOM      ; DO RECIEVE LOOP
5716 105324 001407          P$POP   R4                    ; GET RESULTS
5717 105326          BEQ     29$                    ; IF NO ERRORS, CONTINUE
105326 104456
105330 000033
105332 065732
105334 067516
5718 105336 012701 177777          MOV     4-1,R1      ; PUT ERROR INDICATOR INTO R1
5719 105342 000410          BR      101$
5720 105344          29$:      PRINTB   #OK                    ; RESPONSE OK
105344 012746 061061
105350 012746 000001
105354 010600
105356 104414
105360 062706 000004
5721 105364 005337 050562          101$:   DEC     CPYCNT      ; DECREMENT 'COPIES' COUNTER
5722 105370 001274          BNE     15$                    ; IF MORE TO DO, LOOP
5723 105372          CALL     WRITES #1,SLOT      ; ELSE,UPDATE SUMMARY TABLE
5724 105410 062737 000010 002402  30$:     ADD     #10,SLOT      ; INCREMENT TO NEXT NODE TABLE ENTRY
5725 105416 000636
5726 105420          32$:     BR      10$
5727
5728
5729
5730          ; ACTION ROUTINE TO SET 'RUN LOOPPAIR' FLAG
5731
5732
5733 105424 012737 000033 002310  ACTRNL: MOV     #CLUPPR,KEYWD1      ; SET FLAG
5734 105432 000207          RTS     PC
5735
5736 105434 005037 050552          RUNLUP: CLR     TEMP1          ; CLEAR 'HEADER PRINTED' FLAG

```


105774	012746	062603				MOV	ARGTY6, -(SP)
106000	012746	002322				MOV	STRBUF, -(SP)
106004	012746	062613				MOV	ARGTY7, -(SP)
106010	012746	061624				MOV	TSTMS4, -(SP)
106014	012746	000005				MOV	5, -(SP)
106020	010600					MOV	SP, R0
106022	104414					TRAP	C\$PNTB
106024	062706	000014				ADD	14, SP
5773	106030		CALL	RUNCOM			; DO RECIEVE LOOP
5774	106036		P\$POP	R3			; CHECK RESULTS
5775	106040	001405	BEQ	23\$; IF OK, CONT.
5776	106042		ERRHRD	28,EMSG40,ERR3			
	106042	104456				TRAP	C\$ERHRD
	106044	000034				.WORD	28
	106046	066174				.WORD	EMSG40
	106050	067604				.WORD	ERR3
5777	106052	000410	BR	101\$			
5778	106054		PRINTB	OKRE			
	106054	012746				MOV	OKRE, -(SP)
	106060	012746				MOV	1, -(SP)
	106064	010600				MOV	SP, R0
	106066	104414				TRAP	C\$PNTB
	106070	062706				ADD	4, SP
5779	106074		101\$:	CALL	BLDAST R1,R2		; BUILD RECEIVE ASSIST MESSAGE
5780	106106			CALL	XMIT		; TRANSMIT MESSAGE
5781	106114			P\$POP	R4		; CHECK RESULTS
5782	106116	001276		BNE	21\$; IF OK CONTINUE, ELSE REPORT ERROR
5783	106120			CALL	BINHEX R1,06,STRBUF		; PRINT ERROR MESSAGE
5784	106140			CALL	BINHEX R2,06,STRBU1		
5785	106160		PRINTB	TSTMS4, ARGTY7, STRBUF, ARGTY6, STRBU1			ASSIST NODE =
	106160	012746				MOV	STRBU1, -(SP)
	106164	012746				MOV	ARGTY6, -(SP)
	106170	012746				MOV	STRBUF, -(SP)
	106174	012746				MOV	ARGTY7, -(SP)
	106200	012746				MOV	TSTMS4, -(SP)
	106204	012746				MOV	5, -(SP)
	106210	010600				MOV	SP, R0
	106212	104414				TRAP	C\$PNTB
	106214	062706				ADD	14, SP
5786	106220		CALL	RUNCOM			; DO RECEIVE LOOP
5787	106226		P\$POP	R3			; GET RESULTS
5788	106230	001405	BEQ	25\$; IF OK, CONT.
5789	106232		ERRHRD	28,EMSG41,ERR3			
	106232	104456				TRAP	C\$ERHRD
	106234	000034				.WORD	28
	106236	066263				.WORD	EMSG41
	106240	067604				.WORD	ERR3
5790	106242	000410	BR	102\$			
5791	106244		PRINTB	OKTR			
	106244	012746				MOV	OKTR, -(SP)
	106250	012746				MOV	1, -(SP)
	106254	010600				MOV	SP, R0
	106256	104414				TRAP	C\$PNTB
	106260	062706				ADD	4, SP
5792	106264		102\$:	CALL	BLDFAS R1,R2		; BUILD FULL ASSIST MESSAGE
5793	106276			CALL	XMIT		; TRANSMIT MESSAGE
5794	106304			P\$POP	R4		; CHECK RESULTS

```

5795 106306 001402          BEQ      26$          ; IF OK CONTINUE, ELSE REPORT ERROR
5796 106310 000137 105714  JMP      21$
5797 106314          26$: CALL    BINHEX R1,#6,#STRBUF ; PRINT ERROR MESSAGE
5798 106334          CALL    BINHEX R2,#6,#STRBU1 ;
5799 106354          PRINTB #TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1 ; ASSIST NODE =
      106354 012746 002344          MOV     #STRBU1,-(SP)
      106360 012746 062603          MOV     #ARGTY6,-(SP)
      106364 012746 002322          MOV     #STRBUF,-(SP)
      106370 012746 062613          MOV     #ARGTY7,-(SP)
      106374 012746 061624          MOV     #TSTMS4,-(SP)
      106400 012746 000005          MOV     #5,-(SP)
      106404 010600          MOV     SP,RO
      106406 104414          TRAP   C$PNTB
      106410 062706 000014          ADD     #14,SP
5800 106414          CALL    RUNCOM          ; DO RECEIVE LOOP
5801 106422          P$POP  R3              ; CHECK RESULTS
5802 106424 001405          BEQ      28$          ; IF NO ERRORS, CONT
5803 106426          ERRHRD 28,MSG42,ERR3
      106426 104456          TRAP   C$ERHRD
      106430 000034          .WORD  28
      106432 066353          .WORD  MSG42
      106434 067604          .WORD  ERR3
5804 106436 000410          BR       103$
5805 106440          28$: PRINTB #OKFU
      106440 012745 062013          MOV     #OKFU,-(SP)
      106444 012746 000001          MOV     #1,-(SP)
      106450 010600          MOV     SP,RO
      106452 104414          TRAP   C$PNTB
      106454 062706 000004          ADD     #4,SP
5806 106460 005337 050562  103$: DEC     CPYCNT          ; DECREMENT 'COPIES' COUNTER
5807 106464 001402          BEQ      29$          ; IF MORE TO DO, LOOP
5808 106466 000137 105670  JMP      20$
5809 106472          29$: CALL    WRITES #2,R1,R2 ; ELSE,UPDATE SUMMARY TABLE
5810 106510 062702 000010  ADD     #10,R2
5811 106514 010237 002402  MOV     R2,SLOT
5812 106520 000137 105522  JMP      10$
5813 106524          30$: RETURN
5814
5815 106526 005737 050534  RUNCOM: TST     RETRYJ          ; SEE IF FAILED DUE TO EXCESSIVE COLLISIONS
5816 106532 001120          BNE     38$          ; IF YES, BR, ELSE CONT.
5817 106534 012704 003720  MOV     #TIMERS,R4 ; SET UP FOR 10 SECOND TIMEOUT
5818 106540 012714 000012  MOV     #10.,(R4)
5819 106544 005002          CLR     R2          ; CLEAR RESULTS REGISTER
5820 106546          35$: BREAK
      106546 104422          TRAP   C$BRK
5821 106550 005714          TST     (R4)
5822 106552 001516          BEQ     40$          ; SEE IF TIME HAS EXPIRED
5823 106554          CALL    RECEVE          ; CHECK FOR ANSWER
5824 106562          P$POP  R1          ; R2 HOLDS NO. OF BUFFERS RECEIVED
5825 106564 001770          BEQ     35$          ; IF NO BUFFERS RECIEVED, LOOP
5826 106566 063737 050560 050510  ADD     XFER,S,XFER ; UPDATE BYTES TRANSFERED SUM. COUNTER
5827 106574 005237 050476  INC     S.REC        ; UPDATE PACKETS RECEIVED SUM. COUNTER
5828 106600 013703 003764  MOV     ARGNXT,R3   ; GET RECEIVE RING POINTER
5829 106604          CALL    GETRNX #RRGNXT ; UPDATE POINTER
5830 106616 016301 000006  MOV     6(R3),R1   ; GET PACKET LENGTH FROM DISCRIPTOR
5831 106622 042701 170000  BIC     #170000,R1 ; ZERO OUT EXCESS INFOR
5832 106626 162701 000004  SUB     #4,R1      ; SUBTRACT CRC BYTES

```

```

5833 106630 020137 050566          CMP      R1, BUFLN          ; CHECK FOR LENGTH ERROR
5834 106636 001423                    BEQ      37$                ; IF OK, BR
5835 106640 005237 050502          INC      S.LEN             ; ELSE, UPDATE LENGTH ERRORS COUNTER
5836 106644 012737 062342 002312  MOV      @LENGTH, KEYWD2    ; MOVE 'LENGTH' TO ERROR INDICATOR
5837 106652 012702 177777          MOV      @-1, R2           ; INDICATE ERROR TO R2
5838 106656                    PRINTX   @LGERMS, BUFLN, R1 ; PRINT LENGTH ERROR MESSAGE
                    MOV      R1, -(SP)
                    MOV      BUFLN, -(SP)
                    MOV      @LGERMS, -(SP)
                    MOV      @3, -(SP)
                    MOV      SP, R0
                    TRAP     C$PNTX
                    ADD      @10, SP
5839 106704 000450                    BR       50$                ; AND EXIT
5840 106706 016303 000002          37$:   MOV      2(R3), R3         ; POINT R3 TO MESSAGE BUFFER
5841 106712 066303 000016          ADD      16(R3), R3        ; POINT R3 TO DATA AFTER SKIP COUNT
5842 106716 062703 000030          ADD      @30, R3          ; POINT R3 TO FIRST DATA BYTE
5843 106722 063737 002372 050506  ADD      P$SIZE, S.BYTE    ; UPDATE BYTES COMPARED SUMMARY COUNTER
5844 106730                    CALL     DATCMP P$SIZE, CMPRUF, R3 ; CHECK FOR DATA COMPARE ERRORS
5845 106750                    P$POP   R3                  ; CHECK RESULTS
5846 106752 001425                    BEQ      50$                ; IF ERRORS,
5847 106754 060337 050504          ADD      R3, S.COMP        ; UPDATE COMPARE ERRORS SUMMARY COUNTER
5848 106760 012737 062351 002312  MOV      @COMPAR, KEYWD2    ; MOVE 'COMPARE' TO ERROR INDICATOR
5849 106766 012702 177777          MOV      @-1, R2           ; INDICATE ERROR TO R2
5850 106772 000415                    BR       50$
5851
5852 106774 012737 062516 002312  38$:   MOV      @RETRY, KEYWD2     ; MOVE 'EXCESSIVE COLLISIONS' TO ERROR INCICATOR
5853 107002 012702 177777          MOV      @-1, R2           ; INDICATE ERROR IN R2
5854 107006 000407                    BR       50$
5855
5856 107010 005237 050500          40$:   INC      S.NREC          ; UPDATE MESSAGES NOT RECEIVED COUNTER
5857 107014 012737 062302 002312  MOV      @NORESP, KEYWD2    ; MOVE 'NO RESPONCE' TO ERROR INDICATOR
5858 107022 012702 177777          MOV      @-1, R2           ; INDICATE ERROR TO R2
5859
5860 107026          50$:   RETURN   R2              ; RETURN
5861
5862
5863
5864          ; ACTION ROUTINE TO SET 'RUN PATTERN' FLAG
5865          ;
5866
5867 107032 012737 000005 002310  ACTPAT: MOV      @CPATRN, KEYWD1 ; SET FLAG
5868 107040 000207                    RTS      PC
5869
5870
5871 107042          RUNPAT: P$PUSH P$TYPE ; SAVE TYPE PARAMETER
5872 107046 005037 002370          CLR      P$TYPE          ; SET TYPE TO FIRST TYPE
5873 107052          5$:   CALL     DIRCOM          ; SEND MESSAGES
5874 107060          P$POP   R1            ; GET RESULTS TO KEEP STACK IN ORDER
5875 107062 001403                    BEQ      10$                ; IF OK, CONT
5876 107064 022701 000001          CMP      @1, R1           ; ELSE, WAS TABLE EMPTY
5877 107070 001406                    BEQ      15$                ; IF YES, RETURN
5878 107072 005237 002370          10$:   INC      P$TYPE        ; SET TO NEXT TYPE
5879 107076 022737 000005 002370  CMP      @5, P$TYPE        ; SEE IF DONE ALL OF THEM
5880 107104 002362                    BGE     5$                  ; IF NOT, DO MORE
5881 107106          15$:   P$POP   P$TYPE        ; RESTORE MESSAGE TYPE
5882 107112          RETURN

```

```

5883
5884
5885 ; ACTION ROUTINE TO SHOW THE CURRENT MESSAGE PARAMETERS
5886 ;
5887
5888 107114 013701 002370 ACTSMS: MOV P$TYPE,R1 ;GET MESSAGE TYPE INTO R1
5889 107120 006301 ASL R1 ;MULTIPLY BY 2
5890 107122 062701 003262 ADD #MSGTAB,R1 ;ADD MESSAGE TABLE OFFSET
5891 107126 PRINTF #MSGPRM ;PRINT MESSAGE PARAMETER MESSAGE
107126 012746 054641 MOV #MSGPRM,-(SP)
107132 012746 000001 MOV #1,-(SP)
107136 010600 MOV SP,RO
107140 104417 TRAP C$PNTF
107142 062706 000004 ADD #4,SP
5892 107146 PRINTF #MSG4,(R1),P$SIZE,P$CPYS ;PRINT PARAMETERS
107146 013746 002374 MOV P$CPYS,-(SP)
107152 013746 002372 MOV P$SIZE,-(SP)
107156 011146 MOV (R1),-(SP)
107160 012746 055276 MOV #MSG4,-(SP)
107164 012746 000004 MOV #4,-(SP)
107170 010600 MOV SP,RO
107172 104417 TRAP C$PNTF
107174 062706 000012 ADD #12,SP
5893 107200 105037 003160 CLRB P$NNUF
5894 107204 000207 RTS PC
5895
5896
5897
5898 ; ACTION ROUTINE TO CLEAR THE CURRENT MESSAGE PARAMETERS AND
5899 ; RESET THEM TO THE DEFAULT VALUE
5900 ;
5901
5902 107206 012737 000000 002370 ACTCMS: MOV #ALPHA,P$TYPE ;RESET TYPE
5903 107214 012737 001000 002372 MOV #512,P$SIZE ;RESET SIZE
5904 107222 012737 000001 002374 MOV #1,P$CPYS ;RESET COPIES
5905 107230 PRINTF #CLRMSG ;PRINT MESSAGE PARAMETERS RESET MESSAGE
107230 012746 053716 MOV #CLRMSG,-(SP)
107234 012746 000001 MOV #1,-(SP)
107240 010600 MOV SP,RO
107242 104417 TRAP C$PNTF
107244 062706 000004 ADD #4,SP
5906 107250 PRINTF #MSG4,MSGTAB,P$SIZE,P$CPYS ;PRINT PARAMETERS
107250 013746 002374 MOV P$CPYS,-(SP)
107254 013746 002372 MOV P$SIZE,-(SP)
107260 013746 003262 MOV MSGTAB,-(SP)
107264 012746 055276 MOV #MSG4,-(SP)
107270 012746 000004 MOV #4,-(SP)
107274 010600 MOV SP,RO
107276 104417 TRAP C$PNTF
107300 062706 000012 ADD #12,SP
5907 107304 105037 003160 CLRB P$NNUF ;CLEAR NOTNUF FLAG
5908 107310 000207 RTS PC
5909
5910
5911 ; ACTION ROUTINE TO SET SHOW COUNTERS FLAG
5912 ;
5913

```

5914							
5915	107312			ACTCNT: CALL	FUNCT #RDCNTS		;READ COUNTERS
5916	107324			P\$POP	R1		;CHECK RESULT
5917	107326	001402		BEQ	21\$;BRANCH IF ERROR
5918	107330	000137	110356	JMP	40\$		
5919							;PRINT COUNTER INFO
5920							
5921	107334			21\$: CALL	BINHEX #FHYADR,#6,#STRBUF		;GET ADDRESS INTO ASCII
5922	107356			PRINTF	#CNTRO0,#STRBUF		
	107356	012746	002322			MOV	#STRBUF,-(SP)
	107362	012746	062703			MOV	#CNTRO0,-(SP)
	107366	012746	000002			MOV	#2,-(SP)
	107372	010600				MOV	SP,RO
	107374	104417				TRAP	C\$PNTF
	107376	062706	000006			ADD	#6,SP
5923	107402			PRINTF	#CNTRO1,UCB12+2		
	107402	013746	050162			MOV	UCB12+2,-(SP)
	107406	012746	062763			MOV	#CNTRO1,-(SP)
	107412	012746	000002			MOV	#2,-(SP)
	107416	010600				MOV	SP,RO
	107420	104417				TRAP	C\$PNTF
	107422	062706	000006			ADD	#6,SP
5924	107426			CALL	BINDEC #UCB12+4		
5925	107440			PRINTF	#CNTRO2,#DECSTR		
	107440	012746	075200			MOV	#DECSTR,-(SP)
	107444	012746	063032			MOV	#CNTRO2,-(SP)
	107450	012746	000002			MOV	#2,-(SP)
	107454	010600				MOV	SP,RO
	107456	104417				TRAP	C\$PNTF
	107460	062706	000006			ADD	#6,SP
5926	107464			CALL	BINDEC #UCB12+10		
5927	107476			PRINTF	#CNTRO3,#DECSTR		
	107476	012746	075200			MOV	#DECSTR,-(SP)
	107502	012746	063066			MOV	#CNTRO3,-(SP)
	107506	012746	000002			MOV	#2,-(SP)
	107512	010600				MOV	SP,RO
	107514	104417				TRAP	C\$PNTF
	107516	062706	000006			ADD	#6,SP
5928	107522			PRINTF	#CNTRO4,UCB12+14		
	107522	013746	050174			MOV	UCB12+14,-(SP)
	107526	012746	063133			MOV	#CNTRO4,-(SP)
	107532	012746	000002			MOV	#2,-(SP)
	107536	010600				MOV	SP,RO
	107540	104417				TRAP	C\$PNTF
	107542	062706	000006			ADD	#6,SP
5929	107546			PRINTF	#CNTRO5,UCB12+16		
	107546	013746	050176			MOV	UCB12+16,-(SP)
	107552	012746	063210			MOV	#CNTRO5,-(SP)
	107556	012746	000002			MOV	#2,-(SP)
	107562	010600				MOV	SP,RO
	107564	104417				TRAP	C\$PNTF
	107566	062706	000006			ADD	#6,SP
5930	107572			CALL	BINDEC #UCB12+20		
5931	107604			PRINTF	#CNTRO6,#DECSTR		
	107604	012746	075200			MOV	#DECSTR,-(SP)
	107610	012746	063260			MOV	#CNTRO6,-(SP)
	107614	012746	000002			MOV	#2,-(SP)

	107620	010600					MOV	SP,R0
	107622	104417					TRAP	C\$PNTF
	107624	062706	000006				ADD	#6,SP
5932	107630			CALL	BINDEC	#UCB12+24		
5933	107642			PRINTF	#CNTR07,	#DECSTR		
	107642	012746	075200				MOV	#DECSTR,-(SP)
	107646	012746	063317				MOV	#CNTR07,-(SP)
	107652	012746	000002				MOV	#2,-(SP)
	107656	010600					MOV	SP,R0
	107660	104417					TRAP	C\$PNTF
	107662	062706	000006				ADD	#6,SP
5934	107666			PRINTF	#CNTR08,	UCB12+30		
	107666	013746	050210				MOV	UCB12+30,-(SP)
	107672	012746	063367				MOV	#CNTR08,-(SP)
	107676	012746	000002				MOV	#2,-(SP)
	107702	010600					MOV	SP,R0
	107704	104417					TRAP	C\$PNTF
	107706	062706	000006				ADD	#6,SP
5935	107712			PRINTF	#CNTR09,	UCB12+32		
	107712	013746	050212				MOV	UCB12+32,-(SP)
	107716	012746	063442				MOV	#CNTR09,-(SP)
	107722	012746	000002				MOV	#2,-(SP)
	107726	010600					MOV	SP,R0
	107730	104417					TRAP	C\$PNTF
	107732	062706	000006				ADD	#6,SP
5936	107736			CALL	BINDEC	#UCB12+34		
5937	107750			PRINTF	#CNTR10,	#DECSTR		
	107750	012746	075200				MOV	#DECSTR,-(SP)
	107754	012746	063513				MOV	#CNTR10,-(SP)
	107760	012746	000002				MOV	#2,-(SP)
	107764	010600					MOV	SP,R0
	107766	104417					TRAP	C\$PNTF
	107770	062706	000006				ADD	#6,SP
5938	107774			CALL	BINDEC	#UCB12+40		
5939	110006			PRINTF	#CNTR11,	#DECSTR		
	110006	012746	075200				MOV	#DECSTR,-(SP)
	110012	012746	063552				MOV	#CNTR11,-(SP)
	110016	012746	000002				MOV	#2,-(SP)
	110022	010600					MOV	SP,R0
	110024	104417					TRAP	C\$PNTF
	110026	062706	000006				ADD	#6,SP
5940	110032			CALL	BINDEC	#UCB12+44		
5941	110044			PRINTF	#CNTR12,	#DECSTR		
	110044	012746	075200				MOV	#DECSTR,-(SP)
	110050	012746	063622				MOV	#CNTR12,-(SP)
	110054	012746	000002				MOV	#2,-(SP)
	110060	010600					MOV	SP,R0
	110062	104417					TRAP	C\$PNTF
	110064	062706	000006				ADD	#6,SP
5942	110070			CALL	BINDEC	#UCB12+50		
5943	110102			PRINTF	#CNTR13,	#DECSTR		
	110102	012746	075200				MOV	#DECSTR,-(SP)
	110106	012746	063670				MOV	#CNTR13,-(SP)
	110112	012746	000002				MOV	#2,-(SP)
	110116	010600					MOV	SP,R0
	110120	104417					TRAP	C\$PNTF
	110122	062706	000006				ADD	#6,SP


```

5961
5962      ; ACTION ROUTINE TO PRINT OUT THE NODE TABLE
5963      ;
5964
5965 110374 105037 003160      ACTSND: CLRB      P#NNUF
5966 110400 012737 002404 002402      MOV      #NODTBL,SLOT      ; MOVE NODE TABLE ADDRESS INTO SLOT
5967 110406                                CALL     FULSLT      ; SEE IF TABLE EMPTY
5968 110414 022737 177777 002402      CMP      #-1,SLOT      ; IF YES, DON'T PRINT HEADER
5969 110422 001437                                BEQ      15#
5970 110424                                PRINTF   #NTBHDR      ; PRINT NODE TABLE HEADER
      110424 012746 053462                                MOV      #NTBHDR,-(SP)
      110430 012746 000001                                MOV      #1,-(SP)
      110434 010600                                MOV      SP,RO
      110436 104417                                TRAP     C#PNTF
      110440 062706 000004                                ADD      #4,SP
5971 110444                                10#:    CALL     FULSLT      ; FIND LOCATION IN TABLE WITH AN ADDRESS
5972 110452 022737 177777 002402      CMP      #-1,SLOT      ; CHECK IF AT END OF TABLE
5973 110460 001432                                BEQ      20#
5974 110462                                CALL     BINEX SLOT,#6,#STRBUF ; IF YES, RETURN
5975 110504                                CALL     PRNOD      ; ELSE, PUT ASCII ADDRESS INTO BUFFER
5976 110512 062737 000010 002402      ADD      #6,SLOT      ; PRINT NODE TABLE ENTRY
5977 110520 000751                                BR       10#
5978 110522                                15#:    PRINTF   #TABEMT,#NOD ; INCR. SLOT TO POINT TO NEXT TABLE ENTRY
      110522 012746 053701                                MOV      #NOD,-(SP)
      110526 012746 053632                                MOV      #TABEMT,-(SP)
      110532 012746 000002                                MOV      #2,-(SP)
      110536 010600                                MOV      SP,RO
      110540 104417                                TRAP     C#PNTF
      110542 062706 000006                                ADD      #6,SP
5979 110546 000207                                20#:    RTS      PC      ; RETURN
5980
5981
5982
5983      ; ACTION ROUTINE TO CLEAR A NODE SPECIFIED BY NODE LOGICAL NAME
5984      ; FROM THE NODE TABLE
5985      ;
5986
5987 110550      ACTCNL: P#PUSH R2      ; SAVE R2
5988 110552 013702 003154      MOV      P#NUM,R2      ; PUT NODE LOGICAL NUMBER INTO R2
5989 110556 006302      ASL     R2      ; MULTIPLY BY 8
5990 110560 006302      ASL     R2      ; NODE TABLE ADDRESS *
5991 110562 006302      ASL     R2      ; (LOG. NO. X 8) * #NODTBL
5992 110564 062702 002404      ADD     #NODTBL,R2      ; ADD OFFSET
5993 110570 005022      CLR     (R2).      ; CLEAR ENTRY (8 BYTES)
5994 110572 005022      CLR     (R2).
5995 110574 005022      CLR     (R2).
5996 110576 005012      CLR     (R2)
5997 110600      P#POP  R2      ; RESTORE R2
5998 110602 105037 003160      CLRB   P#NNUF      ; CLEAR NOTNUF FLAG
5999 110606      PRINTF #LOGDEL,P#NUM ; PRINT MESSAGE INDICATING DELETION
      110606 013746 003154                                MOV      P#NUM,-(SP)
      110612 012746 054426                                MOV      #LOGDEL,-(SP)
      110616 012746 000002                                MOV      #2,-(SP)
      110622 010600                                MOV      SP,RO
      110624 104417                                TRAP     C#PNTF
      110626 062706 000006                                ADD      #6,SP
6000 110632 000207      RTS      PC      ; RETURN

```

```

6001
6002
6003      ; ACTION ROUTINE TO INITIATE A UNA PORT COMMAND
6004      ;
6005
6006 110634 105037 003160 ACTFCT: CLRB      P#NNUF      ; CLEAR NOTNUF FLAG
6007 110640      CALL      FUNCT P#NUM  ; CALL FUNCTION ROUTINE WITH FUNCTION CODE
6008 110652      P#POP      R1      ; CHECK RESULTS
6009 110654 001404      BEQ        1#      ; IF OK EXIT
6010 110656      ERRHRD    30,MSG30 ; ELSE REPORT ERROR
        110656 104456
        110660 000036      TRAP      C#ERHRD
        110662 065531      .WORD     30
        110664 000000      .WORD     MSG30
6011 110666 000207      .WORD     0
6012
6013      ; ACTION ROUTINE TO SAVE NODE TABLE
6014      ;
6015
6016
6017 110670 ACTSAV: P#PUSH   R2,R3      ; SAVE R2 AND R3
6018 110674 012702 002404      MOV        #NODTBL,R2 ; SET REGISTERS FOR COPYING
6019 110700 012703 002530      MOV        #SAVTBL,R3 ; R2 = FROM, R3 = TO
6020 110704      PRINTF   #UNSMMSG,#SAVED ; PRINT 'TABLE SAVED' MESSAGE
        110704 012746 054620      MOV        #SAVED,-(SP)
        110710 012746 054561      MOV        #UNSMMSG,-(SP)
        110714 012746 000002      MOV        #2,-(SP)
        110720 010600      MOV        SP,R0
        110722 104417      TRAP      C#PNTF
        110724 062706 000006      ADD        #6,SP
6021 110730 000137 111012      JMP        SAVCOM
6022
6023      ; ACTION ROUTINE TO UNSAVE NODE TABLE
6024      ;
6025
6026
6027 110734 105037 003160 ACTUNS: CLRB      P#NNUF      ; CLEAR 'NOT ENOUGH' FLAG
6028 110740      P#PUSH   R2,R3      ; SAVE R2 AND R3
6029 110744 121427 000057      CMPB      (R4),#57
6030 110750 001002      BNE        5#
6031 110752 000137 111024      JMP        QUIT
6032 110756 012703 002404      5#:      MOV        #NODTBL,R3 ; SET REGISTERS FOR COPYING
6033 110762 012702 002530      MOV        #SAVTBL,R2 ; R2 = FROM, R3 = TO
6034 110766      PRINTF   #UNSMMSG,#RESTOR ; PRINT 'TABLE RESTORED' MESSAGE
        110766 012746 054627      MOV        #RESTOR,-(SP)
        110772 012746 054561      MOV        #UNSMMSG,-(SP)
        110776 012746 000002      MOV        #2,-(SP)
        111002 010600      MOV        SP,R0
        111004 104417      TRAP      C#PNTF
        111006 062706 000006      ADD        #6,SP
6035 111012 012701 000050 SAVCOM: MOV        #TBLLEN,R1 ; MOVE TABLE LENGTH TO R1
6036 111016 012223      10#:      MOV        (R2)+,(R3)+ ; MOVE WORD
6037 111020 005301      DEC        R1      ; DECREMENT COUNTER
6038 111022 001375      BNE        10#      ; IF MORE, LOOP
6039 111024      QUIT:      P#POP      R2,R3      ; ELSE, RESTORE COUNTERS
6040 111030 105037 003160      CLRB      P#NNUF      ; CLEAR 'NOT ENOUGH' FLAG
6041 111034 000207      RTS        PC

```

```

6042
6043
6044      ; ACTION ROUTINE TO CLEAR SUMMARY TABLE
6045      ;
6046
6047 111036 105037 003160 ACTCSU: CLRB P#NUMF ; CLEAR 'NOT ENOUGH' COUNTER
6048 111042 P#PUSH R2 ; SAVE R2
6049 111044 012701 000132 MOV #STLEN,R1 ; MOVE TABLE LENGTH TO R1
6050 111050 012702 002656 MOV #STATBL,R2 ; MOVE SUMMARY TABLE ADDRESS TO R2
6051 111054 005022 S#: CLR (R2)- ; CLEAR FIRST WORD
6052 111056 005301 DEC R1 ; SEE IF FINISHED
6053 111060 001375 BNE S# ; IF NO, DO MORE
6054 111062 PRINTF #TABCLR,#SUMM ; ELSE, PRINT 'TABLE CLEARED' MESSAGE
        MOV #SUMM,-(SP)
        MOV #TABCLR,-(SP)
        MOV #2,-(SP)
        MOV SP,R0
        TRAP C#PNTF
        ADD #6,SP
111062 012746 053706
111066 012746 054514
111072 012746 000002
111076 010600
111100 104417
111102 062706 000006
6055 111106 P#POP R2 ; AND RESTORE R2
6056 111110 RTS PC
6057
6058
6059      ; ACTION ROUTINE TO CHECK FOR PASS DEFAULT VALUE
6060      ;
6061
6062 111112 ACIDFT:
6063 111112 121427 000040 1#: CMPB (R4),#40 ; SEE IF SPACES
6064 111116 001002 BNE 2# ; IF NO, CONT.
6065 111120 005204 INC R4 ; ELSE, POINT TO NEXT CHAR
6066 111122 000773 BR 1# ; AND CHECK AGAIN
6067 111124 121427 000000 2#: CMPB (R4),#0 ; SEE IF DEFAULT VALUE
6068 111130 001007 BNE 10# ; IF NO, BR
6069 111132 012763 000030 000002 MOV #30,2(R3) ; IF YES, POINT R3 TO SKIP CHECK PASS COUNT
6070 111140 012737 000001 003154 MOV #1,P#NUM ; SET DEFAULT TO 1
6071 111146 000403 BR 15# ; RETURN
6072 111150 012763 000004 000002 10#: MOV #4,2(R3) ; POINT R3 TO CHECK FOR PASS COUNT
6073 111156 000207 15#: RTS PC
6074
6075
6076      ; ACTION ROUTINE TO READ A FILE FROM EXTERNAL MEDIA ONTO THE NODE TABLE
6077      ;
6078
6079 111160 ACTUSF:
6080 111160 P#PUSH R2 ; SAVE R2
6081 111162 005002 CLR R2 ; INITIALIZE R2 TO NODE TYPE 'TARGET'
6082 111164 OPEN CBOADR ; OPEN FILE, NAME=ASCIZ STRING
        MOV CBOADR,R0
        TRAP C#OPEN
111164 013700 002366 BCOMPLETE 1# ; RETURN IF SUCCESSFUL
111170 104434 BCS 1# ; ELSE PRINT "OPEN ERROR"
6083 111172 PRINTF #OPNERR,CBOADR ; ELSE PRINT "OPEN ERROR"
        MOV CBOADR,-(SP)
        MOV #OPNERR,-(SP)
        MOV #2,-(SP)
        MOV SP,R0
        TRAP C#PNTF
111174 013746 002366
111200 012746 112120
111204 012746 000002
111210 010600
111212 104417

```

```

6085 111214 062706 000006
      111220
      111220 104435
6086 111222
6087 111230 005737 112114
6088 111234 001064
6089 111236 005737 112116
6090 111242 001411
6091 111244
      111244 012746 112233
      111250 012746 000001
      111254 010600
      111256 104417
      111260 062706 000004
6092 111264 000450
6093 111266
      111266 012746 111570
      111272 012746 112202
      111276 012746 000002
      111302 010600
      111304 104417
      111306 062706 000006
6094 111312
6095 111334
6096 111336 001411
6097 111340
      111340 012746 053306
      111344 012746 000001
      111350 010600
      111352 104417
      111354 062706 000004
6098 111360 000412
6099 111362 110237 002400
6100 111366 105102
6101 111370 142702 000376
6102 111374
6103 111402
6104 111404 000706
6105 111406
6106 111410 105037 003160
6107 111414
      111414 012746 054627
      111420 012746 054561
      111424 012746 000002
      111430 010600
      111432 104417
      111434 062706 000006
6108 111440
6109
6110
6111
6112
6113
6114
6115
6116
6117

      CLOSE
      ; CLOSE FILE
      ADD #6,SP
      TRAP C1CLOS
1$: CALL RDLIN ; READ A LINE AT A TIME
     TST BAD ; SEE IF AN ERROR DURING READ
     BNE 25$ ; BR ON ERROR TO LEAVE
     TST EOFF ; SEE IF EOF BEFORE PROCESS
     BEQ 10$ ; IF VALID, PROCESS
     PRINTF #EOFFND ; ELSE SAY 'END OF FILE' AND LEAVE
      MOV #EOFFND, -(SP)
      MOV #1, -(SP)
      MOV SP,RO
      TRAP C1PNTF
      ADD #4,SP
10$: BR 25$
     PRINTF #PLINE, #FILLIN ; PRINT LINE READ FROM FILE
      MOV #FILLIN, -(SP)
      MOV #PLINE, -(SP)
      MOV #2, -(SP)
      MOV SP,RO
      TRAP C1PNTF
      ADD #6,SP
     CALL EDPACK #FILLIN, #ADRBUF, #6 ; PUT ADDRESS INTO BINARY
     P1POP R1 ; CHECK RESULTS
     BEQ 12$ ; IF OK, BR
     PRINTF #CADERR ; ELSE PRINT ERROR MESSAGE
      MOV #CADERR, -(SP)
      MOV #1, -(SP)
      MOV SP,RO
      TRAP C1PNTF
      ADD #4,SP
12$: BR 25$
     MOVB R2, NODTY ; AND EXIT
     COMB R2 ; SET UP NODE TYPE
     BICB #376, R2 ; SWITCH TYPE FOR NEXT TIME
     CALL ENTRND ; ENTER IN NODE TABLE
     P1POP R1 ; GET RESULTS
     BR 1$ ; READ MORE ADDRESS
25$: P1POP R2 ; RESTORE R2
     CLRB #UNMSG ; CLEAR 'NOT ENOUGH' FLAG
     PRINTF #UNMSG, #RESTOR ; PRINT 'TALBE RESTORED' MESSAGE
      MOV #RESTOR, -(SP)
      MOV #UNMSG, -(SP)
      MOV #2, -(SP)
      MOV SP,RO
      TRAP C1PNTF
      ADD #6,SP
RETURN
.SBTTL READ LINE OF OPENED FILE
;
; THIS ROUTINE GETS BYTES FROM AN OPENED FILE UNTIL A CR IS ENCOUNTERED
; "EOF" AND "BAD" FLAGS ARE SET IF END-OF-FILE OR ERRORS ARE ENCOUNTERED
;
; NOTE: ASSUMING A ASCII TEXT FILE IS BEING READ, FOR EXAMPLE:

```

```

6118      |      AA-00-03-00-01-AB<CR><LF>
6119      |      "
6120      |      AA-00-03-00-01-AB<CR><LF>
6121      |
6122      |      WHAT YOU SEE READ BYTE-BY-BYTE IS:
6123      |      "A..-AB<CR><LF>A..-AB<CR><LF>..<0><0><0>.....???"
6124      |      SO I MADE ASSUMPTION THAT SINCE SEE "0-PADDING" AFTER LAST CHAR TO
6125      |      END-OF-FILEBLOCK, ANY CHARACTER THAT IS NOT "SPACE OR GREATER" OR A
6126      |      <CR> OR <LF> THEN I'LL TAKE THAT AS END-OF-FILE(TEXT), SET EOF-FLAG
6127      |      AND LEAVE.
6128      |
6129      |      INPUTS:
6130      |      FILLIN  BUFFER TO HOLD LINE OF BYTES READ FROM OPENED FILE
6131      |      (CR NOT INCLUDED, 0-BYTE TERMINATED)
6132      |
6133      |      OUTPUTS:
6134      |      BAD     IF NON-ZERO, ERROR IN READING A BYTE FROM FILE
6135      |      EOFF    IF NON-ZERO, END OF FILE WAS ENCOUNTERED
6136      |      FILLIN  ASCIZ STRING THAT WAS READ AS CHAR-CR-LF STRING
6137      |      (CR-LF REMOVED)
6138 111442 012702 111570  RDLIN:  MOV     @FILLIN,R2      ;POINT R2 TO A LINE BUFFER
6139 111446 005037 112114  CLR     BAD              ;CLEAR FLAGS
6140 111452 005037 112116  CLR     EOFF
6141 111456      |      1$:  GETBYT  (R2)          ;GO GET A BYTE FROM INPUT FILE
6142 111462      |      TRAP  C$GETB
6143 111464      |      MOV   RO,(R2)
6144 111466 104426 110012  BCOMPLETE 2$          ;BR IF READ-BYTE SUCESSFUL
6145 111470 103414      |      BCS   2$
6146 111474 012746 112154  4$:  PRINTF @RDERR      ; ELSE PRINT "READ-ERROR"
6147 111476 012746 000001      |      MOV   @RDERR,-(SP)
6148 111478 010600      |      MOV   @1,-(SP)
6149 111480 104417      |      MOV   SP,RO
6150 111500 062706 000004      |      TRAP  C$PNTF
6151 111504 012737 177777 112114  MOV     @-1,BAD          ; SET BAD-TRY FLAG AND LEAVE
6152 111512 000416      |      BR    5$
6153 111514 122712 000015  2$:  CMPB   @15,(R2)      ; IS THE CHARACTER A <CR>
6154 111520 001756      |      BEQ   1$           ; BR IF YES (GO BACK TO GET <LF>)
6155 111522 122712 000012  CMPB   @12,(R2)      ; IS THE CHARACTER A <LF>
6156 111526 001410      |      BEQ   5$           ; BR IF YES (TERMINATE AND LEAVE)
6157 111530 122712 000040  CMPB   @40,(R2)      ; IS IT A "EOF" (END-OF-FILE(TEXT))
6158 111534 101002      |      ; (EOF=ANY NON-CHAR>37 EXCEPT CR,LF)
6159 111536 005202      |      BR   IF YES
6160 111540 000746      |      IF NO, LEAVE CHAR IN BUFFER
6161 111542 012737 177777 112116  6$:  MOV     @-1,EOFF      ; AND GO GET MORE CHARS
6162 111546      |      ; IF YES, TERMINATE INPUT BUFF
6163 111550 105012      |      ; AND SET EOF-FLAG
6164 111552      |      5$:  CLRB   (R2)
6165 111554      |      RETURN
6166 111554      |      FILENM: .BLKB 12.  ;BUFFER FOR FILE NAME
6167 111570      |      FILLIN: .BLKB 132. ;BUFFER FOR SINGLE LINE READ FROM FILE
6168 111774      |      MATCH: .BLKB 80.  ;BUFFER FOR WORD TO MATCH FROM FILE
6169 112114 000000      |      BAD:   .WORD 0     ;ERROR/NOT-FOUND FLAG WORD
6170 112116 000000      |      EOFF:  .WORD 0     ;END-OF-FILE FLAG (<>0 = EOF)

```

```

6167
6168 112120    045    116    045  OPNERR: .ASCIZ  /%NMA?UNABLE TO OPEN "%TMA"?/
      112123    101    077    125
      112126    116    101    102
      112131    114    105    040
      112134    124    117    040
      112137    117    120    105
      112142    116    040    042
      112145    045    124    045
      112150    101    042    / 77
      112153    000
6169 112154    045    116    045  RDERR: .ASCIZ  /%NMA?FILE READ ERROR?/
      112157    101    077    106
      112162    111    114    105
      112165    040    122    105
      112170    101    104    040
      112173    105    122    122
      112176    117    122    077
      112201    000
6170 112202    045    116    045  PLINE: .ASCIZ  /%NMAFILE LINE WAS:%NMT%N/
      112205    101    106    111
      112210    114    105    040
      112213    114    111    116
      112216    105    040    127
      112221    101    123    072
      112224    045    116    045
      112227    124    045    116
      112232    000
6171 112233    045    116    045  EOFEND: .ASCIZ  /%NMAEND-OF-FILE FOUND, FILE READ/
      112236    101    105    116
      112241    104    055    117
      112244    106    055    106
      112247    111    114    105
      112252    040    106    117
      112255    125    116    104
      112260    054    040    106
      112263    111    114    105
      112266    040    122    105
      112271    101    104    000
    
```

```

6172
6173
6174      ;---+
6175      ;          SELMSG                      OPERATOR SELECTED MESSAGE STORAGE
6176      ;
6177      ;          THIS ROUTINE WILL TAKE THE OPERATOR SELECTED MESSAGE FROM THE COMMAND
6178      ;          LINE INPUT STRING BUFFER AND PUT IT INTO A BUFFER AT LOCATION OPSLBF.
6179      ;
6180      ;          INPUTS -                      P1 - ADDRESS OF OPERATOR SELECTED MESSAGE IN
6181      ;                                     INPUT STRING
6182      ;          EXPLICIT OUTPUTS -           NONE
6183      ;          IMPLICIT OUTPUTS -          THE BUFFER AT OPSLBF WILL CONTAIN THE ASCII
6184      ;                                     OPERATOR SELECTED INPUT STRING FOLLOWED BY
6185      ;                                     A NULL CHARACTER
6186      ;          SUBORDINATE ROUTINES -      NONE
6187      ;          CALLING PROCEEDURE -        CALL SELMSG P1 ;INPUT ADDRESS OF ASCII STRING
6188      ;          REGISTER USAGE -           R1 CONTAINS ADDRESS OF INPUT STRING
6189      ;                                     R2 CONTAINS ADDRESS OF OUTPUT STRING
    
```



```

6190
6191
6192 112274
6193 112276 012702 003570
6194 112302 005003
6195 112304 105711
6196 112306 001404
6197 112310 112122
6198 112312 005203
6199 112314 000137 112304
6200 112320 112712 0000C0
6201 112324 010337 003314
6202 112330
6203
6204
6205
6206
6207
6208
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6220
6221 112332
6222 112340
6223 112342 001403
6224 112344
6225 112350 C00416
6226 112352 012703 000006
6227 112356 013701 002402
6228 112362 012702 002314
6229 112366 112221
6230 112370 005303
6231 112372 001375
6232 112374 005201
6233 112376 113711 002400
6234 112402
6235 112406
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246

```

```

      |---+
SELMSG: P#POP  R1          ;PUT ADDRESS OF OPR, SEL ASCII STRING INTO R1
        MOV     #OPSLBF,R2 ;PUT ADDRESS OF OUTPUT BUFFER INTO R2
        CLR     R3         ;CLEAR CHARACTER COUNTER
5$:     TSTB    (R1)        ;CHECK FOR END OF STRING
        BEQ     10$        ;GO TO 10$ IF END
        MOVB   (R1)+,(R2)+ ;ELSE, MOVE BYTE TO OUTPUT BUFFER
        INC     R3         ;COUNT NUMBER OF CHARACTERS IN INPUT BUFFER
        JMP     5$         ;GO DO MORE CHARACTERS
10$:    MOVB   #0,(R2)     ;PUT ZERO AT END OF OUTPUT BUFFER
        MOV     R3,MSG6C   ;STORE NUMBER OF CHARACTERS FOR USE IN BUF. BUILDING
        RETURN

```

```

      |---+
      |
      | ENTRND                      ENTER NODE IN TABLE
      |
      | THIS ROUTINE ENTERS A NODE INTO THE NODE TABLE
      |
      | INPUTS                      NONE
      | EXPLICIT OUTPUTS           P1 - ZERO IF SUCCESSFUL, -1 IF TABLE FULL
      | IMPLICIT OUTPUTS          THE ADDRESS CONTAINED IN ADRBUF WILL BE
      |                            ADDED TO THE NODE TABLE IN THE FIRST
      |                            AVAILABLE SLOT WITH THE NODE TYPE CONTAINED
      |                            IN NODTY (TARGET OR ASSIST)
      | SUBORDINATE ROUTINES       FINDSL - FIND EMPTY SLOT IN TABLE
      | CALLING PROCEEDURE         CALL ENTRND
      |                            P#POP P1          ;OUTPUT GOOD/BAD RESULT
      |---+

```

```

ENTRND: CALL FINDSL          ;FIND AVAILABLE SLOT IN TABLE
        P#POP  R1           ;CHECK IF TABLE FULL
        BEQ   5$           ;IF NOT FULL BR TO 5$
        P#PUSH # -1        ;ELSE PUT FULL INDICATION ON STACK
        BR   20$          ;RETURN
5$:     MOV     #6,R3       ;SET INCR. COUNTER TO 6 (BYTES)
        MOV     SLOT,R1     ;MOV ADDRESS OF AVAILABLE SLOT TO R1
        MOV     #ADRBUF,R2 ;MOV ADDRESS OF NODE ADDRESS TO R2
10$:    MOVB   (R2)+,(R1)+ ;MOV BYTE OF ADDRESS
        DEC     R3         ;DECR. COUNTER
        BNE    10$        ;CONTINUE UNTIL 6 BYTES TRANSFERED
        INC     R1         ;SET POINTER TO NODE TYPE LOCATION
        MOVB   NODTY,(R1) ;MOVE NODE TYPE INTO TABLE
        P#PUSH #0          ;PUT ADDRESS ADDED INDICATION ON STACK
20$:    RETURN

```

```

      |---+
      |
      | FINDSL                      FIND EMPTY SLOT IN NODE TABLE
      |
      | INPUTS                      NONE
      | EXPLICIT OUTPUTS           NONE
      | IMPLICIT OUTPUTS          THE ADDRESS OF THE FIRST AVAILABLE SLOT IN THE
      |                            NODE TABLE WILL BE LOCATED IN SLOT. THE
      |                            PARAMETER STACK WILL CONTAIN -1 IF THE NODE
      |                            TABLE IS FULL AND 0 IF AN EMPTY SLOT WAS FOUND
      | SUBORDINATE ROUTINES       NONE

```

```

6247      ;          CALLING PROCEDURE          CALL FINDSL
6248      ;                                     P$POP P1          ; -1 IF FULL/ 0 IF SLOT AVAILABLE
6249      ;
6250      ;---+
6251
6252 112410 012702 002404      FINDSL: MOV      #NODTBL,R2          ;MOVE ADDRESS OF NODE TABLE TO R2
6253 112414 022712 000000      10$:  CMP      #0,(R2)          ;SEE IF SLOT EMPTY
6254 112420 001422              BEQ      20$          ;IF YES, BR 20$
6255 112422 062702 000010      ADD      #0.,R2          ;ELSE MOVE POINTER TO NEXT ENTRY LOC.
6256 112426 022712 177777      CMP      #-1,(R2)       ;SEE IF AT END OF TABLE
6257 112432 001370              BNE     10$          ;IF NOT, CONTINUE LOOKING
6258 112434              PRINTF #TABFUL,#NOD      ;ELSE, PRINT TABLE FULL MESSAGE
        112434 012746 053701              MOV      #NOD,-(SP)
        112440 012746 053560              MOV      #TABFUL,-(SP)
        112444 012746 000002              MOV      #2,-(SP)
        112450 010600              MOV      SP,R0
        112452 104417              TRAP    C$PNTF
        112454 052706 000006              ADD     #6,SP
6259 112460              P$PUSH #-1          ;PUT TABLE FULL INDICATION ON STACK
6260 112464 000404              BR      30$          ;RETURN
6261 112466 010237 002402      20$:  MOV      R2,SLOT      ;MOVE ADDRESS OF EMPTY LOC. INTO SLOT
6262 112472              P$PUSH #0          ;PUT LOC. FOUND INDICATION ON STACK
6263 112476      30$:  RETURN          ;RETURN
6264
6265      ;---+
6266      ;          FULSLT          FULL SLOT ROUTINE
6267      ;
6268      ;          THIS ROUTINE FINDS A LOCATION IN THE TABLE WHERE A NODE PHYSICAL
6269      ;          ADDRESS EXISTS. IT IS USED WHEN PRINTING OUT THE NODE TABLE.
6270      ;
6271      ;          INPUTS          NONE
6272      ;          EXPLICIT OUTPUTS  NONE
6273      ;          IMPLICIT OUTPUTS  THE LOCATION SLOT WILL CONTAIN THE PHYSICAL
6274      ;          ADDRESS OF A NODE TABLE ENTRY. SLOT WILL
6275      ;          CONTAIN -1 WHEN POINTING TO THE END OF THE
6276      ;          NODE TABLE
6277      ;          SUBORDINATE ROUTINES  NONE
6278      ;          CALLING PROCEDURE  CALL FULSLT
6279      ;
6280      ;---+
6281
6282 112500 013701 0 2402      FULSLT: MOV      SLOT,R1          ;MOVE SLOT LOCATION TO R1
6283 112504 022711 000000      10$:  CMP      #0,(R1)          ;CHECK IF EMPTY
6284 112510 001412              BEQ     20$          ;IF YES, BR 20$
6285 112512 022711 .77777      CMP      #-1,(R1)       ;SEE IF END OF NODE TABLE
6286 112516 001403              BEQ     15$          ;IF YES, BR 15$
6287 112520 010137 002402      MOV      R1,SLOT      ;ELSE PUT EMPTY LOC. ADDRESS INTO SLOT
6288 112524 000407              BR      30$          ;RETURN
6289 112526 012737 177777 002402 15$:  MOV      #-1,SLOT      ;PUT -1 INTO SLOT TO SHOW END OF TABLE
6290 112534 000403              BR      30$          ;RETURN
6291 112536 062701 000010      20$:  ADD      #8.,R1          ;INCR. POINTER TO NEXT LOCATION
6292 112542 000760              BR      10$          ;CHECK NEXT LOC.
6293 112544      30$:  RETURN          ;RETURN
6294
6295      ;---+
6296      ;          CMPADR          COMPAIR TWO ADDRESSES
6297      ;
  
```

```

6298      ;      THIS ROUTINE COMPAIRS TWO SIX BYTE STRINGS
6299      ;
6300      ;      INPUTS      P1 - ADDRESS OF FIRST STRING
6301      ;      P2 - ADDRESS OF SECOND STRING
6302      ;      OUTPUTS    P3 - 0 = COMPARISON/-1 = NO COMPARISON
6303      ;
6304      ;      CALLING PROCEDURE  CALL  CMPADR P1,P2
6305      ;      P$POP      P3
6306      ;
6307      ;---+
6308
6309      CMPADR: P$POP      R2,R3      ;PUT ADDRESS OF STRING TO BE COMPARED IN R2 AND R3
6310      112552 022223      CMP      (R2)+,(R3)+      ;DO FIRST TWO BYTES COMPARE
6311      112554 001006      BNE      10$      ; IF NO, EXIT
6312      112556 022223      CMP      (R2)+,(R3)+      ;DO SECOND TWO BYTES COMPARE
6313      112560 001004      BNE      10$      ; IF NO, EXIT
6314      112562 021213      CMP      (R2),(R3)      ;DO LAST TWO BYTES COMPARE
6315      112564 001002      BNE      10$      ; IF NO, EXIT
6316      112566 005001      CLR      R1      ;PUT COMPARISON OK INDICATOR IN R1
6317      112570 000402      BR      15$
6318      112572 012701 177777 10$:      MOV      0-1,R1      ;PUT NO COMPARISON INDICATOR IN R1
6319      112576      15$:      RETURN  R1
6320
6321      ;---+
6322      ;      PRTNOD      PRINT NODE TABLE
6323      ;
6324      ;      INPUTS      NONE
6325      ;      EXPLICIT OUTPUTS  NONE
6326      ;      IMPLICIT OUTPUTS  ONE ENTRY IN THE NODE TABLE WILL BE PRINTED
6327      ;      SUBORDINATE ROUTINES  NONE
6328      ;      CALLING SEQUENCE  CALL PRTNOD
6329      ;
6330      PRTNOD: PRINTF  0NODADR,0STRBUF      ;PRINT NODE ADDRESS
6331      112602 012746 002322      MOV      0STRBUF,-(SP)
6332      112606 012746 053430      MOV      0NODADR,-(SP)
6333      112612 012746 000002      MOV      02,-(SP)
6334      112616 010600      MOV      SP,R0
6335      112620 104417      TRAP    C$PNTF
6336      112622 062706 000006      ADD      06,SP
6337      112626 013702 002402      MOV      SLOT,R2      ;MOVE SLOT ADDRESS INTO R2
6338      112632 162702 002404      SUB      0NODTBL,R2      ;CALCULATE NODE LOGICAL NAME
6339      112636 006202      ASR      R2      ;USING: LOG. NO. *
6340      112640 006202      ASR      R2      ;(SLOT-0NODTAB)/8
6341      112642 006202      ASR      R2
6342      112644 010246 053440      PRINTF  0LOGNAM,R2      ;PRINT LOGICAL NAME
6343      112646 012746 000002      MOV      R2,-(SP)
6344      112652 012746 000002      MOV      0LOGNAM,-(SP)
6345      112656 010600      MOV      02,-(SP)
6346      112660 104417      MOV      SP,R0
6347      112662 062706 000006      TRAP    C$PNTF
6348      112666 013702 002402      ADD      06,SP
6349      112672 062702 000007      MOV      SLOT,R2      ;SEE IF TARGET OR ASSIST NODE
6350      112676 111203      ADD      07,R2      ; INFO CONTAINED IN 7TH BYTE OF ENTRY
6351      112700 020327 000001      MOVB    (R2),R3      ;MOVE INTO R3
6352      112704 001404      CMP      R3,0CASIST      ;SEE IF ASSIST NODE
6353      112706 012737 062613 002312  BFQ      10$      ; IF YES, BR 10$
6354      112706 012737 062613 002312  MOV      0ARGTY7,KEYWD2 ; ELSE MOVE 'TARGET' INTO KEYWD2
  
```

```

6343 112714 000403          BR      20$          ; CONTINUE
6344 112716 012737 062603 002312 10$:  MOV     $ARGTY6,KEYWD2 ; MOVE 'ASSIST' INTO KEYWD2
6345 112724          013746 002312          PRINTF $NODTYP,KEYWD2 ;PRINT NODE TYPE
      112724          012746 053453          MOV     KEYWD2,-(SP)
      112730          012746 000002          MOV     $NODTYP,-(SP)
      112734          010600          MOV     $2,-(SP)
      112740          104417          MOV     SP,R0
      112742          062706 000006          TRAP   C$PNTF
      112744          RETURN          ;RETURN
6346 112750          EXIT      TST
6347 112752          104432          TRAP   C$EXIT
      112754          000002          .WORD  L10015-
6348
6350          ;*****
6351          ;   INSERT LOCAL STORAGE THAT IS USED ONLY
6352          ;   DURING THIS TEST.
6353          ;*****
6354
6355          ;*****
6356          ;   INSERT MESSAGES THAT ARE USED ONLY
6357          ;   DURING THIS TEST.
6358          ;*****
6360
6361          .EVEN
6362
6363 112756          ENDTST
      112756          L10015:
      112756 104401          TRAP   C$ETST
6364
6366          ;*****
6367          ;   BEGIN THE REMAINING TESTS ON NEW PAGES.
6368          ;*****

```

```

6371
6372          .SBTTL  HARDWARE PARAMETER CODING SECTION
6373
6374          ;++
6375          ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
6376          ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
6377          ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
6378          ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
6379          ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
6380          ; WITH THE OPERATOR.
6381          ;--
6382
6383          BGNHRD
6384          112760          000015
6385          112760          000015
6386          112762          L$HARD: .WORD L10016-L$HARD/2
6387
6388          ;*****
6389          ; INSERT HARDWARE PARAMETER INTERPRETIVE CODE HERE.  THIS CODE
6390          ; IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR
6391          ; DEVICE INFORMATION TO PUT IN THE P-TABLE.  THIS CODE IS USED
6392          ; IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
6393          ; USED IN THIS SECTION ARE "GPRMD", "GPRMA" AND "GPRML".
6394          ;*****
6395          GPRMA  ASKCSR,0,0,160000,177776,YES          ; GET CSR ADDRESS
6396          112762          000031          .WORD  T$CODE
6397          112764          113014          .WORD  ASKCSR
6398          112766          160000          .WORD  T$LOLIM
6399          112770          177776          .WORD  T$HILIM
6400          6396 112772          GPRMA  ASKVEC,2,0,0,776,YES          ; GET VECTOR ADDRESS
6401          112772          001031          .WORD  T$CODE
6402          112774          113047          .WORD  ASKVEC
6403          112776          000000          .WORD  T$LOLIM
6404          113000          000776          .WORD  T$HILIM
6405          6397 113002          GPRMD  ASKPRI,4,0,340,0,7,YES          ; GET PRIORITY LEVEL
6406          113002          002032          .WORD  T$CODE
6407          113004          113103          .WORD  ASKPRI
6408          113006          000340          .WORD  340
6409          113010          000000          .WORD  T$LOLIM
6410          113012          000007          .WORD  T$HILIM
6411
6412          6399 113014          ENDHRD
6413          113014          .EVEN
6414          L10016:
6415
6416          ;*****
6417          ; INSERT MESSAGES THAT ARE USED ONLY
6418          ; DURING THE HARDWARE PARAMETER CODING SECTION.
6419          ;*****
6420          6408 113014          127          110          101  ASKCSR: .ASCIZ  /WHAT IS THE PCSRO ADDRESS?/
6421          113017          124          040          111
6422          113022          123          040          124
6423          113025          110          105          040
6424          113030          120          103          123
6425          113033          122          117          040
6426          113036          101          104          104

```

	113041	122	105	123	
	113044	123	077	000	
6409	113047	127	110	101	ASKVEC: .ASCIZ /WHAT IS THE VECTOR ADDRESS?/
	113052	124	040	111	
	113055	123	040	124	
	113060	110	105	040	
	113063	126	105	103	
	113066	124	117	122	
	113071	040	101	104	
	113074	104	122	105	
	113077	123	123	077	
	113102	000			
6410	113103	127	110	101	ASKPRI: .ASCIZ /WHAT IS THE PRIORITY LEVEL?/
	113106	124	040	111	
	113111	123	040	124	
	113114	110	105	040	
	113117	120	122	111	
	113122	117	122	111	
	113125	124	131	040	
	113130	114	105	126	
	113133	105	114	077	
	113136	000			
6411					.EVEN
6412					

6414
6415
6416
6417
6418
6419
6420
6421
6422
6423
6424
6425 113140
113140 000000
113142
6426
6428
6429
6430
6431
6432
6433
6435
6436
6437
6438 113142
113142
6439
6440
6442
6443
6444
6445
6447
6448 113142
6449 113142
6450
6452
6453
6454
6455
6457
6458 113162
113162 000000
113164 000000
113166

```
.SBTTL SOFTWARE PARAMETER CODING SECTION

***
; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
; WITH THE OPERATOR.
;--

        BGNSFT

                                .WORD L10017-L$SOFT/2
L$SOFT::

;*****
; INSERT SOFTWARE PARAMETER INTERPRETIVE CODING HERE. THIS CODE
; IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR
; SOFTWARE INFORMATION WHICH WILL BE PLACED IN THE SOFTWARE
; TABLE. THIS SECTION IS OPTIONAL.
;*****

        .EVEN

        ENDSFT

                                .EVEN
L10017:

;*****
; INSERT MESSAGES THAT ARE USED ONLY
; DURING THE SOFTWARE PARAMETER CODING SECTION.
;*****

$PATCH::
        .BLKW 10

;*****
; THIS IS A PATCH AREA THAT SHOULD BE INCLUDED IN ALL DIAGNOSTICS.
; ADJUST THE SIZE TO FIT YOUR OWN PREFERENCES.
;*****

        LASTAD

                                .EVEN
                                .WORD 0
                                .WORD 0
L$LAST::
```

6460
6461
6463
6464
6465
6466
6467
6468
6469
6470
6471
6472
6474
6475
6476
6477
6478
6479
6480
6481

000001

```

*****
|      HARDCODED P-TABLES MAY BE PLACED HERE BY USING THE SETUP MACROS.
|      THIS SECTION IS OPTIONAL AND SHOULD BE REMOVED IF IT IS NOT BEING
|      USED.  CHANGE THE POINTER MACRO ARGUMENT TO REFLECT THE REMOVAL.
|
|      THE P-TABLES ARE DELIMITED BY THE "BGNSETUP" AND "ENDSETUP" MACROS.
|      THE "BGNSETUP" MACRO HAS ONE ARGUMENT WHICH IS THE NUMBER OF
|      P-TABLE ENTRIES.  EACH ENTRY IS DELIMITED BY THE "BGNPTAB" AND
|      "ENDPTAB" MACROS.  NEITHER OF THESE MACROS REQUIRE AN ARGUMENT.
*****
|      BGNSETUP      1
|      BGNPTAB
|      .WORD      0
|      ENDPYTAB
|      ENDSETUP
|
|      .END

```


ACTALP	103450	BAD	112114	CEXIT	000020 G	CNTR06	063260	C#ESEG	000005
ACTBLD	101166	BCOUNT	050526 G	CFLAG	003672	CNTR07	063317	C#ESUB	000003
ACTCMS	107206	BINDEC	075014 G	CFUNCT	000040	CNTR08	063367	C#ETST	000001
ACTCNL	110550	BINHEX	072672	CLIACT	100456	CNTR09	063442	C#EXIT	000032
ACTCNT	107312	BIT0	000001 G	CLIALN	000007	CNTR10	063513	C#GETB	000026
ACTCPY	103742	BIT00	000001 G	CLIALP	000006	CNTR11	063552	C#GETW	000027
ACTCSU	111036	BIT01	000002 G	CLIBIF	000003	CNTR12	063622	C#GMAN	000043
ACTCTT	103520	BIT02	000004 G	CLIBR	000002	CNTR13	063670	C#GPHR	000042
ACTDFT	111112	BIT03	000010 G	CLIBRX	052554	CNTR14	063735	C#GPLO	000030
ACTDIR	104776	BIT04	000020 G	CLIDEC	000011	CNTR15	063771	C#GPRI	000040
ACTEXT	103350	BIT05	000040 G	CLIERM	052445	CNTR16	064033	C#INIT	000011
ACTFCT	110634	BIT06	000100 G	CLIFRR	000000	CNTR17	064101	C#INLP	000020
ACTHLP	100620	BIT07	000200 G	CLIEXI	000001	CNTR18	064154	C#MANI	000050
ACTIDY	102264	BIT08	000400 G	CLINBG	052527	CNTR19	064221	C#MEM	000031
ACTMSG	103242	BIT09	001000 G	CLINUF	052476	COMAND	071350 G	C#MSG	000023
ACTNAD	104020	BIT1	000002 G	CLINUM	000005	COMPAR	062351	C#OPEN	000034
ACTNAL	104156	BIT10	002000 G	CLIOCT	000010	CONES	000017	C#PNTB	000014
ACTNOD	100656	BIT11	004000 G	CLISPA	000004	COPHSL	000024	C#PNTF	000017
ACTNUF	100610	BIT12	010000 G	CLISTR	000012	COUNT	050542 G	C#PNTS	000016
ACTNUL	100616	BIT13	020000 G	CLITRE	051070	CPATRN	000005	C#PNTX	000015
ACTONE	103460	BIT14	040000 G	CLIIPM	052436	CPYCNT	050562 G	C#QIO	000377
ACTOPR	103530	BIT15	100000 G	CLKBR	003676	CPYMT	053775	C#RDBU	000007
ACTPAT	107032	BIT2	000004 G	CLKCSR	003674	CRC	004000 G	C#REFG	000047
ACTRNA	104356	BIT3	000010 G	CLKEN	003704	CRNALL	000032	C#RESE	000033
ACTRNL	105424	BIT4	000020 G	CLKHZ	003702	CRUN	000004	C#REVI	000003
ACTRUN	104236	BIT5	000040 G	CLKINT	067702 G	CSAVE	000006	C#RFLA	000021
ACTSAV	110670	BIT6	000100 G	CLKSET	067656	CSAVR4	000014	C#RPT	000025
ACTSMS	107114	BIT7	000200 G	CLKVEC	003700	CSHCTR	000002 G	C#SEFG	000046
ACTSND	110374	BIT8	000400 G	CLRCNT	000013 G	CSHMSG	000034	C#SPRI	000041
ACTSR4	103442	BIT9	001000 G	CLRMSG	053716	CSIZE	000026	C#SVEC	000037
ACTSUM	101710	BLDAST	073606 G	CLRSTA	000017 G	CTARGT	000000 G	C#TPRI	000013
ACTSZE	103664	BLDBUF	074324 G	CLUPPR	000033	CTYPE	000025	C#COLL	000074 G
ACTTYP	103656	BLDFAS	073242 G	CMDBUF	002200	CUNSAV	000041	C#MREC	000010 G
ACTUNS	110734	BLDLD	072760 G	CMDTY1	062444	CUNSVF	000045	C#MXMT	000040 G
ACTUSF	111160	BLDMSG	052757	CMDTY2	062451	CZEROS	000020	C#PREC	000004 G
ACTXAD	103360	BLDREQ	074124 G	CMDTY3	062461	C#AU	000052	C#PXMD	000054 G
ACTZRO	103470	BOE	000400 G	CMDTY4	062467	C#AUTO	000061	C#PXMT	000034 G
ACTOAL	103510	BOOT	000005 G	CMDTY5	062474	C#BRK	000022	C#PXM2	000050 G
ACTIAL	103500	BUFL	100000 G	CMDTY6	062500	C#BSEG	000004	C#PXM3	000044 G
ADR	000020 G	BUFLEN	050566 G	CMDTY7	062510	C#BSUB	000002	C#RDAT	000020 G
ADRBUF	002314	BUILD	000003	CMDTY8	062515	C#CEFG	000045	C#RERB	000014 G
ADWDFL	054340	CADERR	053306	CMDTY9	062523	C#CLCK	000062	C#RERR	000016 G
ALLNOG	062151	CADRER	053232	CMPADR	112546	C#CLEA	000012	C#RLEX	000032 G
ALPHA	000000 G	CALPHA	000016	CMPBUF	050570 G	C#CLOS	000035	C#RLIN	000030 G
ANCHOR	070042	CASIST	000001 G	CMPER1	066754	C#CLP1	000006	C#RMD8	000024 G
ARGTY1	062536	CBOADR	002366	CMPER2	067051	C#CVEC	000036	C#SECS	000002 G
ARGTY2	062544	CCCITY	000023	CMPSTR	072530	C#DCLN	000044	C#XABB	000066 G
ARGTY3	062555	CCITT	000005 G	CNDADR	000030	C#DDDU	000051	C#XABT	000070 G
ARGTY4	062566	CCLMSG	000035	CNDLOG	000037	C#DRPT	000024	C#XDAT	000060 G
ARGTY5	062577	CCLNAD	000004 G	CNODEL	000031	C#DU	000053	C#XMDB	000064 G
ARGTY6	062603	CCLNAL	000010 G	CNODE	000015	C#EDIT	000003	COALT	000022
ARGTY7	062613	CCLSUM	000042	CNTR00	062703	C#ERDF	000055	C#ALT	000021
ASKCSR	113014	CCNTR	000036	CNTR01	062763	C#ERHR	000056	DATCHP	074406 G
ASKPRI	113103	CCPYS	000027	CNTR02	063032	C#ERRO	000060	DECSTR	075200 G
ASKVEC	113047	CDEFLT	000044	CNTR03	063066	C#ERSF	000054	DEF	002000 G
ASSEMB	000010	CDTR	000043	CNTR04	063133	C#ERSO	000057	DEPADR	047752 G
B	000007	CXADR	000013	CNTR05	063210	C#ESCA	000010	DESTIN	000000 G

DFPTBL	002170	G	ERRFLG	050530	G	GETCL	100274	HELP29	061346	LDADR1	000022	G		
DIAGMC	000000		ERRMSG	052432	G	GETCOM	074304	HELP3	056254	LDADR2	000032	G		
DIFCOM	105026		ERRNBR	052430	G	GETFNT	000002	G	HELP30	061416	LDFCT1	000020	G	
DIRECT	062173		ERROR	070150	G	GETPCB	000001	G	HELP4	056325	LDFCT2	000030	G	
DMPMEM	000020	G	ERRS	040000	G	GETRXN	074262	G	HELP5	056376	LDMEM	000021	G	
DNI	004000	G	FRTTYP	052426	G	GETXNX	074274	G	HELP6	056476	LDRESP	052574		
DNIFLG	050522	G	ERR1	067460	G	G#CNTD	000200		HELP7	056611	LDSKIP	000016	G	
EDPACK	072322		ERR2	067516	G	G#DELM	000372		HELP8	056722	LENGTH	062342		
EF.CON	000036	G	ERR3	067604	G	G#DISP	000003		HELP9	057012	LGERMS	067117		
EF.NEW	000035	G	EVL	000004	G	G#EXCP	000400		HEXBIN	072552	LOCDST	072206		
EF.PWR	000034	G	EXIT	006011		G#HILI	000002		HEXC	072650	LOE	040000	G	
EF.RES	000037	G	E#END	002100		G#LOLI	000001		HLPEND	003260	LOGDEL	054426		
EF.STA	000040	G	E#LOAD	000035		G#NO	000000		HLPTAB	003164	LOGNAM	053440		
EMSG0	003464	G	FAADR1	000022	G	G#OFFS	000400		HN	072526	LOPDIR	050720	G	
EMSG01	064272		FAADR2	000032	G	G#FSI	000376		HOE	100000	LOT	000010	G	
EMSG02	064321		FAADR3	000042	G	G#PRMA	000001		HXF0RM	072414	LST	072670		
EMSG03	064341		FAADR4	000052	G	G#PRMD	000002		HXN	072412	LUPAIR	062162		
EMSG04	064373		FAFCT1	000020	G	G#PRML	000000		IBE	010000	L#ACP	002110	G	
EMSG05	064415		FAFCT2	000030	G	G#RADA	000140		ICAB	040000	L#APT	002036	G	
EMSG06	064450		FAFCT3	000040	G	G#RADB	000000		IDENT	000010	L#AU	100266	G	
EMSG07	064510		FAFCT4	000050	G	G#RADD	000040		IDU	000040	L#AUT	002070	G	
EMSG08	064553		FASIST	051026	G	G#RADL	000120		IER	020000	L#AUTO	100140	G	
EMSG09	064613		FASKIP	000016	G	G#RADQ	000020		ILADMS	053051	L#CCP	002106	G	
EMSG1	003465	G	FATFLG	050512	G	G#XFER	000004		ILADM1	053135	L#CLEA	100142	G	
EMSG10	064656		FATI	000400	G	G#YES	000010		ILLADR	002650	L#CO	002032	G	
EMSG11	064716		FILENM	111554		HMMSG1	055342		INICLN	100130	L#DEPO	002011	G	
EMSG14	064755		FILLIN	111570		HMMSG2	055413		INIEXI	100132	L#DESC	002130	G	
EMSG15	065031		FINDSL	112410		HMMSG3	055466		INIT	076600	L#DESP	002076	G	
EMSG16	065064		FRAM	020000	G	HMMSG4	055522		INIT1	076620	L#DEVP	002060	G	
EMSG18	065127		FRADR	000004	G	HMMSG5	055577		INTE	000100	G	L#DISP	002164	G
EMSG19	065176		FREMEM	047662	G	HMMSG6	055650		INTR	000200	G	L#DLY	002116	G
EMSG2	003466	G	FRESTZ	047660	G	HMMSG7	055710		ISR	000100	G	L#DTP	002040	G
EMSG20	065234		FULAST	062207		HMMSG8	055751		IXE	004000	G	L#DTYP	002034	G
EMSG22	065266		FULSLT	112500		HMMSG9	056014		I#AU	000041		L#DU	100260	G
EMSG23	065321		FUNCT	071372	G	HCADER	000016	G	I#AUTO	000041		L#DUT	002072	G
EMSG24	065366		FUNTAB	047676	G	HELP	000001		I#CLN	000041		L#DVTY	002122	G
EMSG25	065442		F#AU	000015		HELP1	056060		I#DU	000041		L#EF	002052	G
EMSG3	003467	G	F#AUTO	000020		HELP10	057101		I#HRD	000041		L#ENVI	002044	G
EMSG30	065531		F#BGN	000040		HELP11	057172		I#INIT	000041		L#ERRT	052426	C
EMSG31	065576		F#CLEA	000007		HELP12	057270		I#MOD	000041		L#ETP	002102	G
EMSG32	065637		F#DU	000016		HELP13	057375		I#MSG	000041		L#EXP1	002046	G
EMSG33	065714		F#ENC	000041		HELP14	057474		I#PROI	000040		L#EXP4	002064	G
EMSG34	065732		F#HARD	000004		HELP15	057573		I#PTAB	000041		L#EXP5	002066	G
EMSG35	066002		F#HW	000013		HELP16	057676		I#PWR	000041		L#HARD	112762	G
EMSG36	066037		F#INIT	000006		HELP17	057765		I#RPT	000041		L#HIME	002120	G
EMSG37	066064		F#JMP	000050		HELP18	060070		I#SEG	000041		L#MFCP	002016	G
EMSG38	066130		F#MOD	000000		HELP19	060140		I#SETU	000041		L#MPTP	002022	G
EMSG4	003470	G	F#MSG	000011		HELP2	056161		I#SFT	000041		L#HW	002170	G
EMSG40	066174		F#PROT	000021		HELP20	060243		I#SRV	000041		L#ICP	002104	G
EMSG41	066263		F#PWR	000017		HELP21	060321		I#SUB	000041		L#INIT	076600	G
EMSG42	066353		F#RPT	000012		HELP22	060404		I#TST	000041		L#LADP	002026	G
EMSG5	003570	G	F#SEG	000003		HELP23	060505		J#IMP	000167		L#LAST	113166	G
ENP	000400	G	F#SOFT	000005		HELP24	060605		KEYWD1	002310		L#LOAD	002100	G
ENTRND	112332		F#SRV	000010		HELP25	060735		KEYWD2	002312		L#LUN	002074	G
EOFF	112116		F#SUB	000002		HELP26	061021		LCAR	004000	G	L#MREV	002050	G
EOFFND	112233		F#SW	000014		HELP27	061125		LCLKEN	000100	G	L#NAME	002000	G
ERRBLK	052434	G	F#TEST	000001		HELP28	061227		LCOL	010000	G	L#PRIO	002042	G

L\$PROT 076572 G
L\$PRT 002112 G
L\$REPP 002062 G
L\$REV 002010 G
L\$RPT 076560 G
L\$SOFT 113142 G
L\$SPC 002056 G
L\$SPCP 002020 G
L\$SPTP 002024 G
L\$STA 002030 G
L\$SW 002200 G
L\$TEST 002114 G
L\$TIML 002014 G
L\$UNIT 002012 G
L10000 002176
L10001 002200
L10002 067514
L10003 067602
L10004 067654
L10005 070022
L10006 076570
L10010 100136
L10011 100140
L10012 100256
L10013 100264
L10014 100272
L10015 112756
L10016 113014
L10017 113142
MATCH 111774
MESPAT 062054
MESPA1 062125
MORE 010000 G
MSGAD 003316 G
MSGCNT 003300 G
MSGPRM 054641
MSGTAB 003262
MSGTY0 062372
MSGTY1 062400
MSGTY2 062405
MSGTY3 062413
MSGTY4 062420
MSGTY5 062425
MSGTY6 062433
MSGOC 003300
MSG00 003334 G
MSG01 003464 G
MSG02 003465 G
MSG03 003466 G
MSG04 003467 G
MSG05 003470 G
MSG1 054711
MSG1C 003302
MSG11 055024
MSG12 055137
MSG2 055177
MSG2C 003304

MSG3 055235
MSG3C 003306
MSG4 055276
MSG4C 003310
MSG5C 003312
MSC5C 003314
NCHN 020000 G
NEW 100112
NIHLT 000006 G
NIRCNT 050516 G
NIUNI 000007 G
NOCLK 062622
NOCMPR 054160
NOD 053701
NODADR 053430
NODE 000002
NODTBL 002404
NODTY 002400
NODTYP 053453
NOD0 051070
NOD1 051074
NOD10 051154
NOD100 051720
NOD101 051722
NOD102 051724
NOD103 051730
NOD104 051734
NOD105 051740
NOD106 051744
NOD107 051746
NOD11 051156
NOD110 051752
NOD111 051756
NOD112 051772
NOD113 051776
NOD114 052012
NOD115 052016
NOD116 052032
NOD117 052036
NOD12 051170
NOD120 052052
NOD121 052056
NOD122 052072
NOD123 052076
NOD124 052112
NOD125 052116
NOD126 052132
NOD127 052136
NOD13 051174
NOD130 052142
NOD131 052146
NOD132 052152
NOD133 052166
NOD134 052172
NOD135 052176
NOD136 052202
NOD137 052220

NOD14 051200
NOD140 052224
NOD141 052230
NOD142 052234
NOD143 052240
NOD144 052244
NOD145 052264
NOD146 052270
NOD147 052302
NOD15 051212
NOD150 052306
NOD151 052324
NOD152 052330
NOD153 052346
NOD154 052352
NOD155 052366
NOD156 052372
NOD157 052376
NOD16 051216
NOD160 052400
NOD161 052404
NOD162 052410
NOD163 052412
NOD164 052416
NOD165 052422
NOD166 052424
NOD17 051234
NOD2 051100
NOD20 051236
NOD21 051250
NOD22 051252
NOD23 051254
NOD24 051256
NOD25 051272
NOD26 051276
NOD27 051316
NOD3 051102
NOD30 051322
NOD31 051340
NOD32 051344
NOD33 051362
NOD34 051366
NOD35 051402
NOD36 051404
NOD37 051424
NOD4 051116
NOD40 051430
NOD41 051432
NOD42 051434
NOD43 051440
NOD44 051444
NOD45 051450
NOD46 051454
NOD47 051456
NOD5 051120
NOD50 051462
NOD51 051476

NOD52 051502
NOD53 051520
NOD54 051524
NOD55 051544
NOD56 051550
NOD57 051554
NOD6 051134
NOD60 051556
NOD61 051562
NOD62 051576
NOD63 051602
NOD64 051606
NOD65 051612
NOD66 051624
NOD67 051630
NOD7 051140
NOD70 051634
NOD71 051640
NOD72 051644
NOD73 051650
NOD74 051654
NOD75 051672
NOD76 051676
NOD77 051714
NORESP 062302
NOTNUF 000012
NO.NTR 000006 G
NTBHDR 053462
NULL 000000
NULSTR 053362
N10 051074
N100 051456
N101 051462
N102 051502
N104 051524
N106 051550
N110 051554
N12 051102
N120 051556
N121 051562
N122 051602
N123 051630
N124 051644
N126 051650
N130 051654
N132 051676
N134 051720
N135 051722
N14 051120
N140 051724
N141 051730
N142 051734
N143 051744
N16 051140
N160 051746
N161 051752
N162 052016

N163 052036
N164 052056
N165 052076
N166 052116
N167 052136
N168 052146
N170 052152
N175 052202
N178 052234
N18 051156
N180 052240
N181 052244
N182 052270
N183 052306
N184 052330
N185 052346
N186 052352
N190 052376
N20 051174
N200 052400
N201 052404
N210 052412
N212 052416
N215 052424
N22 051216
N23 051236
N24 051252
N25 051256
N26 051276
N28 051322
N29 051344
N30 051366
N31 051404
N32 051430
N80 051434
N81 051440
N82 051444
N90 051450
N95 051454
OFLO 010000 G
OK 061661
OKFU 062013
OKRE 061702
OKTR 061746
ONE 004000 G
ONEALT 000003 G
ONES 000001 G
OPNERR 112120
OPRSEL 000006 G
OPSLBF 003570
OWN 100000 G
O\$APTS 000000
O\$AU 000000
O\$BGNR 000001
O\$BGNS 000000
O\$DU 000000
O\$ERRT 000000

O\$GNSW= 000000	P\$PASS 002376	RXI = 020000 G	TASIST 050742 G	T\$TEMP= 000005
O\$POIN= 000001	P\$RADX 003156	SAVCOM 111012	TBLLEN= 000050 G	T\$TEST= 000001
O\$SETU= 000000	P\$SIZE 002372	SAVED 054620	TEMP 050550 G	T\$TSTM= 177777
PART 075214 G	P\$TREE 003146	SAVTBL 002530	TEMP1 050552 G	T\$TSTS= 000001
PASABT 061526	P\$TRV 075216	SELMSG 112274	TEMP2 050554 G	T\$AU = 010014
PATCH 050572 G	P\$TR5 075226	SERI = 100000 G	TEMP3 050556 G	T\$AUT= 010011
PATRN 062262	P\$TYPE 002370	SFPTBL 002200 G	TENPWR 075130	T\$CLE= 010012
PCB80 047666 G	QUIT 111024	SIADDR= 000042 G	TIMERS 003720	T\$DU = 010013
PCB82 047670 G	RASIST 050774 G	SIDEV = 000053 G	TIMER1 003714	T\$HAR= 010016
PCB84 047672 G	RBFcnt 050524 G	SIECO = 000030 G	TIMER2 003716	T\$HW = 010000
PCB86 047674 G	RCBI = 002000 G	SIFCNT= 000035 G	TIMMIN 003706	T\$INI= 010010
PCCALL 050564 G	RCVBUF 050540 G	SIMSG1 066440	TIMOUT 050532 G	T\$MSG= 010004
PCEFLG 050514 G	RCVERR 050536 G	SIMSG2 066477	TIMSEC 003710	T\$PRO= 010007
PCEI = 040000 G	RDCNTS= 000012 G	SIMSG3 066532	TIMTCK 003712	T\$RPT= 010006
PCLKCT= 001600 G	RDDEFA= 000002 G	SIMSG4 066572	TMRF = 000012 G	T\$SOF= 010017
PCLKEN= 000111 G	RDERR 112154	SIMSG5 066613	TMR0 = 000011 G	T\$SRV= 010005
PCMSG 066714 G	RDLIN 111442	SIMSG6 066641	TRAST 062223	T\$SW = 010001
PCSFO 047632 G	RDMODE= 000014 G	SIMSG7 066667	TRVACT 075344	T\$TES= 010015
PCSHOC 047642 G	RDMULA= 000006 G	SIRCP1= 000022 G	TRVADR 076330	T1 100274 G
PCSR1 047634 G	RDPHYA= 000004 G	SIUECO= 000031 G	TRVALN 076144	UAM = 000200 G
PCSR1C 047644 G	RDRNGS= 000010 G	SIVERS= 000027 G	TRVALP 076100	UBTO = 040000 G
PCSR2 047636 G	RDRSTA = 000016 G	SIZLMT 054061	TRVBIF 075450	UCB10 050110 G
PCSR2C 047646 G	RDSYS = 000022 G	SLOT 002102 G	TRVBR 075440	UCB11 050134 G
PCSR3 047640 G	READY = 000002 G	SOURCC= 000006 G	TRVBRC 075364	UCB12 050160 G
PCSR3C 047650 G	RECAST 062243	SOURCE 072320	TRVDEC 075544	UCB13 050160 G
PCTO = 000200 G	RECERR 052631	START 076700	TRVERR 075402	UCB20 050340 G
PDMD = 000010 G	RECEVE 071776 G	STATBL 002656	TRVEXI 075422	UCB21 050340 G
PFNOP = 000000 G	REQID 050712 G	STATUS 050312 G	TRVNMA 075564	UCB22 050400
PHYADR 047762 G	RESET = 000000 G	STBLN= 000132 G	TRVNOB 075374	UCB23 050400
PLINE 112202	RESTOR 054627	STOP = 000017 G	TRVNUM 075556	UCB4 050010 G
PNOP = 000003 G	RESTR 100046	STP = 001000 G	TRVOCT 075556	UCB7 050050 G
PNT = 001000 G	RETRY 062316	STRBUF 002322	TRVSPA 075472	UOBB 050466 G
PREG14 070024	RETRYS 050534 G	STRBU1 002344	TRVSTR 076232	UNACSR 047652 G
PRI = 002000 G	RMTC = 000010 G	STRT = 000004 G	TSTMS1 061551	UNAIN1 070262 G
PRIMLD= 000001 G	RPKLEN= 002756 G	SUMM 053706	TSTMS2 061571	UNAI3R 071144 G
PRI00 = 000000 G	RRGCLR 003760 G	SUMRY= 000007	TSTMS3 061611	UNAPRI 047656 G
PRI01 = 000040 G	RRGLST 003770 G	SUMMS1 067215	TSTMS4 061624	UNAVEC 047654 G
PRI02 = 000100 G	RRGNXT 003764 G	SUMMS2 067236	TXI = 010000 G	UNBOND 054252
PRI03 = 000140 G	RRGSRT 003754 G	SUMMS3 067322	T\$ARGC= 000002	UNHLT= 000005 G
PRI04 = 000200 G	RRG001 026006 G	SUMMS4 067351	T\$CODE= 002032	UNIT 047664 G
PRI05 = 000240 G	RRG002 030764 G	SUMMS5 067434	T\$ERRN= 000036	UNSMG 054561
PRI06 = 000300 G	RRG003 033742 G	SUMMS6 067452	T\$EXCP= 000000	WAIT 070066 G
PRI07 = 000340 G	RRG004 036720 G	SVCGBL= 000000	T\$FLAG= 000040	WDMODE= 000015 G
PROTOT= 000014 G	RRG005 041676 G	SVCINS= 000001	T\$GMAN= 000000	WOMULA= 000007 G
PROT00 050544 G	RRG006 044654 G	SVCSUB= 000001	T\$HILI= 000007	WOPHYA= 000005 G
PROT02 050546 G	RRING 004066 G	SVCTAG= 000001	T\$LAST= 000001	WRNGS= 000011 G
PRINOD 112602	RRNGTB 003722 G	SVCTST= 000001	T\$LOLI= 000000	WOSYS = 000023 G
P\$ACT 003150	RSET = 000040 G	S\$LSYM= 010000	T\$LSYM= 010000	WRITES 074544
P\$AERR 003162	RSTT = 000015 G	S.BYTE 050506 G	T\$LTNO= 000001	XFER 050560 G
P\$BUFA 003144	RTRY = 002000 G	S.COMP 050504 G	T\$NEST= 177777	XFLAG 050520 G
P\$CNT 003152	RTRYER 052675	S.LEN 050502 G	T\$NSO = 000005	XMIT 071434 G
P\$CPYS 002374	RUN = 000003 G	S.NREC 050500 G	T\$PTNU= 000000	XPLEN= 002756 G
P\$EXIT 075342	RUNALL 104366	S.REC 050476 G	T\$SAVL= 177777	XPWR = 100000 G
P\$GDBD 003161	RUNCOM 106526	S.XFER 050510 G	T\$SEGL= 177777	XRGCUR 003756 G
P\$MERR 003163	RUNDIR 105006	TABCLR 054514	T\$SUBN= 000000	XRGLST 003766 G
P\$NNLF 003160	RUNLUP 105434	TABEMT 053632	T\$TAGL= 177777	XRGNXT 003762 G
P\$NUM 003154	RUNPAT 107042	TABFUL 053560	T\$TAGN= 010020	XRGSRT 003752 G

H15

SYMBOL TABLE

XRG001	004162 G	XRGTB	003736 G	ZROALT	000004 G	\$RDCN	050150 G	\$RDSY	050360 G
XRG002	007140 G	X\$	000167	\$CLRC	050260 G	\$RDDE	047750 G	\$WDMC	050040 G
XRG003	012116 G	X\$ALWA	000000	\$CLRS	050320 G	\$RDMC	050000 G	\$WDMO	050300 G
XRG004	015074 G	X\$FALS	000040	\$DMEM	050330 G	\$RDMO	050270 G	\$WDPH	047770 G
XRG005	020052 G	X\$OFFS	000400	\$LMEM	050350 G	\$RDPH	047760 G	\$WDRN	050124 G
XRG006	023030 G	X\$TRUE	000020	\$PATCH	113142 G	\$RDRN	050100 G	\$WTSY	050370 G
XRING	003772 G	ZEROS	000002 G	\$PNOP	047746 G	\$RDST	050310 G		

. ABS. 113166 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 31328 WORDS (123 PAGES)
DYNAMIC MEMORY: 20060 WORDS (77 PAGES)
ELAPSED TIME: 00:12:07
CZUACB.BIN,CZUACB.LST/-SP=SVC34R.MLB,CZUACB.P11

....B1	GLOBAL TEXT SECTIONB5	XMIT TRANSMIT UNA....E9	CLI ACTION TABLE AND....B13
....C1	GLOBAL TEXT SECTIONC5	XMIT TRANSMIT UNA....C9	CLI ACTION TABLE AND....C13
....D1	GLOBAL TEXT SECTIOND5	RECEIVE RECEIVE UNA....D9	CLI ACTION TABLE AND....D13
....E1	GLOBAL TEXT SECTIONE5	EDPACK ETHERNET DAT....E9	CLI ACTION TABLE AND....E13
....F1	GLOBAL TEXT SECTIONF5	HXFORM HEX FORMAT R....F9	CLI ACTION TABLE AND....F13
....G1	GLOBAL TEXT SECTIONG5	HXFORM HEX FORMAT R....G9	CLI ACTION TABLE AND....G13
....H1	GLOBAL TEXT SECTIONH5	HEXBIN HEX TO BINAR....H9	CLI ACTION TABLE AND....H13
....I1	GLOBAL TEXT SECTIONI5	HEXBIN HEX TO BINAR....I9	CLI ACTION TABLE AND....I13
....J1	GLOBAL TEXT SECTIONJ5	BINHEX BINARY TO HE....J9	CLI ACTION TABLE AND....J13
....K1	GLOBAL TEXT SECTIONK5	BLDLC BUILD LOOP D....K9	CLI ACTION TABLE AND....K13
....L1	GLOBAL TEXT SECTIONL5	BLDLD BUILD LOOP D....L9	CLI ACTION TABLE AND....L13
....M1	GLOBAL TEXT SECTIONM5	BLDFAS BUILD PACKET....M9	CLI ACTION TABLE AND....M13
....N1	GLOBAL TEXT SECTIONN5	BLDFAS BUILD PACKET....N9	CLI ACTION TABLE AND....N13
PROGRAM MACROSB2	GLOBAL TEXT SECTIONB6	BLDAST BUILD TRANSM....B10	CLI ACTION TABLE AND....B14
PROGRAM MACROSC2	GLOBAL TEXT SECTIONC6	BLDREQ BUILD REQUES....C10	CLI ACTION TABLE AND....C14
PROGRAM MACROSD2	GLOBAL TEXT SECTIOND6	GET?NX GET NEXT TRA....D10	CLI ACTION TABLE AND....D14
PROGRAM MACROSE2	GLOBAL TEXT SECTIONE6	BLDBUF BUILD MESSAG....E10	CLI ACTION TABLE AND....E14
PROGRAM MACROSF2	GLOBAL TEXT SECTIONF6	BLDBUF BUILD MESSAG....F10	CLI ACTION TABLE AND....F14
PROGRAM MACROSG2	GLOBAL TEXT SECTIONG6	DATCHP COMPARE DATA....G10	READ LINE OF OPENEDG14
PROGRAM MACROSH2	GLOBAL TEXT SECTIONH6	WRITES WRITE DATA O....H10	READ LINE OF OPENEDH14
PROGRAM MACROSI2	GLOBAL TEXT SECTIONI6	WRITES WRITE DATA O....I10	READ LINE OF OPENEDI14
DISPATCH TABLEJ2	GLOBAL TEXT SECTIONJ6	BINDEC CONVERT A 32....J10	READ LINE OF OPENEDJ14
DEFAULT HARDWARE P-T....K2	GLOBAL TEXT SECTIONK6	BINDEC CONVERT A 32....K10	READ LINE OF OPENEDK14
DEFAULT HARDWARE P-1....L2	GLOBAL TEXT SECTIONL6	COMMAND LINE TRAVERS....L10	READ LINE OF OPENEDL14
GLOBAL EQUATES SECTI....M2	GLOBAL TEXT SECTIONM6	COMMAND LINE TRAVERS....M10	READ LINE OF OPENEDM14
GLOBAL EQUATES SECTI....N2	GLOBAL TEXT SECTIONN6	COMMAND LINE TRAVERS....N10	HARDWARE PARAMETER C....N14
GLOBAL EQUATES SECTI....B3	GLOBAL TEXT SECTIONB7	COMMAND LINE TRAVERS....B11	SOFTWARE PARAMETER C....B15
GLOBAL EQUATES SECTI....C3	GLOBAL TEXT SECTIONC7	COMMAND LINE TRAVERS....C11	SOFTWARE PARAMETER C....C15
GLOBAL EQUATES SECTI....D3	GLOBAL TEXT SECTIOND7	COMMAND LINE TRAVERS....D11	SYMBOL TABLED15
GLOBAL EQUATES SECTI....E3	GLOBAL TEXT SECTIONE7	REPORT CODING SECTIO....E11	SYMBOL TABLEE15
GLOBAL EQUATES SECTI....F3	GLOBAL TEXT SECTIONF7	PROTECTION TABLEF11	SYMBOL TABLEF15
GLOBAL EQUATES SECTI....G3	GLOBAL TEXT SECTIONG7	INITIALIZE SECTIONG11	SYMBOL TABLEG15
GLOBAL DATA SECTIONH3	GLOBAL TEXT SECTIONH7	INITIALIZE SECTIONH11	SYMBOL TABLEH15
GLOBAL DATA SECTIONI3	GLOBAL TEXT SECTIONI7	INITIALIZE SECTIONI11	SYMBOL TABLEI15
GLOBAL DATA SECTIONJ3	GLOBAL TEXT SECTIONJ7	INITIALIZE SECTIONJ11	
GLOBAL DATA SECTIONK3	GLOBAL TEXT SECTIONK7	INITIALIZE SECTIONK11	
GLOBAL DATA SECTIONL3	GLOBAL TEXT SECTIONL7	AUTODROP SECTIONL11	
GLOBAL DATA SECTIONM3	GLOBAL TEXT SECTIONM7	CLEANUP CODING SECTI....M11	
GLOBAL DATA SECTIONN3	GLOBAL ERROR REPORTN7	DROP UNIT SECTIONN11	
GLOBAL DATA SECTIONB4	GLOBAL ERROR REPORTB8	ADD UNIT SECTIONB12	
GLOBAL DATA SECTIONC4	GLOBAL SUBROUTINES S....C8	TEST 1:C12	
GLOBAL DATA SECTIOND4	GLOBAL SUBROUTINES S....D8	CLI ACTION TABLE AND....D12	
GLOBAL DATA SECTIONE4	CLKSET CLOCK SETUPE8	CLI ACTION TABLE AND....E12	
GLOBAL DATA SECTIONF4	CLKINT CLOCK INTERRU....F8	CLI ACTION TABLE AND....F12	
GLOBAL DATA SECTIONG4	PREG14 PRESERVE REG....G8	CLI ACTION TABLE AND....G12	
GLOBAL DATA SECTIONH4	ERROR HANDLE UNA I....H8	CLI ACTION TABLE AND....H12	
COMMAND LINE ACTIONI4	UNAINI INITIALIZE T....I8	CLI ACTION TABLE AND....I12	
COMMAND LINE ACTIONJ4	UNAINI INITIALIZE T....J8	CLI ACTION TABLE AND....J12	
COMMAND LINE ACTIONK4	UNAINI INITIALIZE T....K8	CLI ACTION TABLE AND....K12	
GLOBAL TEXT SECTIONL4	UNAINI INITIALIZE T....L8	CLI ACTION TABLE AND....L12	
GLOBAL TEXT SECTIONM4	UNAIISR UNA INTERRUPT....M8	CLI ACTION TABLE AND....M12	
GLOBAL TEXT SECTIONN4	FUNCT SUBR TO PER....N8	CLI ACTION TABLE AND....N12	