

1

.REM 6

IDENTIFICATION

PRODUCT CODE: AC T585A MC  
PRODUCT NAME: CVNIAAO DEQNA NI EXERCISER DIAGNOSTIC  
PRODUCT DATE: 09 MAR 84  
MAINTAINER: MERRIMACK DIAGNOSTIC ENGINEERING  
AUTHOR: STEVE SKONETSKI

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) 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

## 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
2.0	OPERATING INSTRUCTIONS
2.1	COMMANDS
2.2	SWITCHES
2.3	FLAGS
2.4	HARDWARE QUESTIONS
2.5	NETWORK INTERCONNECT EXERCISER COMMANDS
2.5.1	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 DEQNA 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 CVNIA NIE PROGRAM, THE FOLLOWING MINIMUM HARDWARE IS REQUIRED:

A PDP 11 CPU WITH A Q-BUS (11/23, 11/23+, 11/03)  
MINIMUM OF 24K WORDS OF MEMORY  
A WORKING, LINE OR REAL TIME CLOCK  
A CONSOLE TERMINAL  
ANY XXDP SUPPORTED LOAD MEDIA  
DEQNA QBUS TO ETHERNET ADAPTER

### 1.3 RELATED DOCUMENTS AND STANDARDS

DEQNA FUNCTIONAL AND PROGRAMMING SPEC  
 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 DEQNA'S OR DEQNA'S AT EACH END OF THE LINK  
 HAVE ALREADY BEEN TESTED.

IF NO LINE OR REAL TIME CLOCK IS FOUND, THE PROGRAM WILL NOT RUN.

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

### 1.5 ASSUMPTIONS

IT IS ASSUMED THAT THE COMMUNICATIONS DEVICE (DEQNA) 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.

### 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 F_L_A_G_S (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. LAST 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
MOE	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 (D) ? 1<CR>
```

```
UNIT 0
DEVICE CSR ADDRESS : (0) 174440 ?<CR>
INTERRUPT VECTOR ADDRESS : (0) 300 ?<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.

UNDERLINED ITEMS IN THE EXAMPLES REPRESENT THE USER'S RESPONSE TO THE PROMPT TO EXECUTE THAT PARTICULAR COMMAND.

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

EXAMPLE:

NIE> (A) ? H

OR

NIE> (A) ? ?

SHOW NODES PRINTS OUT THE CONTENTS OF THE NODE TABLE.

EXAMPLE:

NIE> (A) ? SH N

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

EXAMPLE:

NIE> (A) ? SH M

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

\*\*\*\*\*

NOTE:

BIT 3 OF THE RECEIVE ERROR BIT MASK IS THE "RUNT PACKET" BIT. THIS IS NOT PART

OF THE COUNTERS INFO. THIS BIT INDICATES THAT A RUNT PACKET WAS FOUND IN THE RECEIVE BUFFER. THE SOFTWARE WILL DISCARD THE RUNT PACKET. IT IS NOT CONSIDERED AN ERROR AND WILL NOT BE RECORDED AS ONE. THE HARDWARE DESIGN WILL ALLOW A RUNT PACKET CONDITION TO OCCUR AND REQUIRES THAT THE SOFTWARE HANDLE SUCH A CONDITION. SEE THE QNA DESIGN SPECIFICATIONS FOR FURTHER INFORMATION ON THE NATURE OF RUNT PACKETS.

THE MEANING OF THE COUNTERS IS EXPLAINED IN THE DEUNA USER'S GUIDE NUMBER EK DEUNA UG.

THE ERROR BIT MASKS FOR XMIT AND RECEIVE ARE IN BINARY FORMAT, NOT OCTAL OR DECIMAL (I.E. 11 = BIT1 AND BIT0, NOT AN OCTAL OR DECIMAL ELEVEN.

\*\*\*\*\*

EXAMPLE:

NIE> (A) ? SM C  
--

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

\*\*\*\*\*

NOTE

IF THE NODE IS TO BE CHANGED FROM A TARGET TO AN ASSIST OR VICE VERSA, IT MUST BE CLEARED FROM THE TABLE FIRST!

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).

\*\*\*\*\*

EXAMPLE OF TARGET NODE ENTRY:

NIE> (A) ? N AA-00-04 FF-FF FO/T  
-----

EXAMPLE OF ASSIST NODE ENTRY:

NIE> (A) ? N AA 00-04 FF FF FO/A  
-----

EXAMPLE:

NIE> (A) ? N AA 00 04 FF-FF FO  
-----

EXAMPLE OF CHANGING A TARGET NODE TO AN ASSIST NODE:

NIE> (A) ? CL N/AA 00-04-FF FF FO (SEE CLEAR COMMAND)

NIE> (A) ? N AA-00 04 FF FF FO/A

MESSAGE/TYP E=/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 (000000....)
- 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 MAY BE BETWEEN 46 AND 512 BYTES.

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

EXAMPLE OF CHANGING TYPE:

NIE> (A) ? M/T=ZERO

EXAMPLE OF CHANGING SIZE:

NIE> (A) ? M/S=256

EXAMPLE OF CHANGING BOTH TYPE AND SIZE:

NIE> (A) ? M/S=512/T=ALPHA

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 INDEFINITELY. 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.

\*\*\* 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.

IF THERE ARE TWO ASSIST NODES IN A ROW, THE PROGRAM WILL CHANGE THE FIRST NODE TO A TARGET NODE, AND A SOFT ERROR WILL BE INDICATED.

BE VERY CAREFUL 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

RESP RUNS THE RESPONDER TEST, A SECTION OF CODE THAT PROVIDES LOOP-SERVER FUNCTIONS SUCH AS FORWARDING MESSAGES, ANSWERING CONSOLE ID REQUESTS, AND TRANSMITTING A SYSTEM ID EVERY 8 TO 9 MINUTES.

\*\*\*\*\*

I M P O R T A N T

THIS SECTION MUST BE RUN IF THE DEQNA IS TO BE USED AS A LOOP ASSIST OR TARGET NODE ON THE ETHERNET. THE OTHER TESTS WILL SIMPLY IGNORE FORWARDING REQUESTS, AND WILL NOT XMIT CONSOLE IDS.

THE ONLY WAY TO ESCAPE FROM A LARGE OR INFINITE NUMBER OF PASSES IS TO TYPE CONTROL C. BE CAREFUL!! IF YOU TYPE "START" TO DSR> AFTER THE CONTROL C YOU WILL DESTROY ALL SUMMARY STATISTICS AND COUNTERS. USE THE "RESTART" COMMAND TO DSR> TO GET BACK TO THE NIE> PROMPT AND PRESERVE THE COUNTERS.

\*\*\*\*\*

EXAMPLE OF RUNNING THE DIRECT TEST ONE PASS:

NIE> (A) ? R D

EXAMPLE OF RUNNING THE DIRECT TEST 5 PASSES:

NIE> (A) ? R D/P=5  
- - -

EXAMPLE OF RUNNING THE DIRECT TEST INFINITE PASSES:

NIE> (A) ? R D/P=0  
-----

EXAMPLE OF RUNNING LOOP PAIR TEST:

NIE> (A) ? R L  
-

EXAMPLE OF RUNNING RESPONDER TEST:

NIE> (A) ? R R

IDENTIFY ADR A REQUEST ID MESSAGE IS SENT TO THE NODE SPECIFIED BY ADR AND THE RESPONDED SYSTEM ID PARAMETERS ARE PRINTED.

## EXAMPLE:

NIE> (A) ? ID AA 00 04 FF-FF F0  
 - - - - -

## 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.

IT IS POSSIBLE TO MISS A TRANSMISSION WITHIN THE TEN MINUTE PERIOD. IF NO NODES SHOW UP AFTER A BUILD, TRY WAITING 4 OR 5 MINUTES AND DO THE BUILD AGAIN.

A FEATURE OF THIS PROGRAM IS THE ABILITY TO 'C OUT OF THE BUILD ROUTINE AT ANY TIME WITHOUT HAVING TO WAIT FOR THE BUILDING PROCESS TO COMPLETE. THIS IS NICE WHEN YOU WANT TO TEST A FEW NODES AND DON'T CARE ABOUT 'ALL' THE NODES ON THE NET.

## EXAMPLE:

NIE> (A) ? BU  
 -

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.

## EXAMPLE OF CLEARING A NODE USING THE NI ADDRESS:

NIE> (A) ? CL N/AA-00 04 FF-FF F0  
 - - - - -

## EXAMPLE OF CLEARING A NODE USING IT'S LOGICAL NAME:

NIE> (A) ? CL N/N3  
-- --

\*\*\*\*\*

NOTE:

TO FIND WHICH LOGICAL NAME IS ASSOCIATED WITH AN ADDRESS, EXECUTE THE "SHOW NODE" COMMAND.

ALSO, A CLEARED NODE CAN BE RESTORED TO THE NODE TABLE BY EXECUTING THE "UNSAVE" COMMAND.

\*\*\*\*\*

EXAMPLE OF CLEARING ALL NODES:

NIE> (A) ? CL N/ALL  
-- --

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 TRANSFERRED

\*\*\*\*\*

NOTE:

BYTES COMPARED REPRESENTS DATA MINUS THE LOOP-SERVER PROTOCOL OVERHEAD, AND THUS WILL BE LESS THAN BYTES TRANSFERRED WHICH REPRESENTS DATA PLUS LOOP SERVER PROTOCOL OVERHEAD

\*\*\*\*\*

EXAMPLE:

NIE> (A) ? SUMM  
---

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 SAVED INTERNALLY.

EXAMPLE:

NIE> (A) ? SAV  
--

UNSAVE THE UNSAVE COMMAND RESTORES THE CONTENTS OF THE NODE TABLE. USED. THE PDP 11 VERSION USES THE CONTENTS OF ITS INTERNALLY SAVED TABLE.

EXAMPLE:

NIE> (A) ? UNS  
---

EXIT RETURNS CONTROL TO THE DIAGNOSTIC SUPERVISOR (EITHER VDS OR DRS).

\*\*\*\*\*

NOTE:

THE DSR> RESTART AND CONTINUE COMMAND CANNOT BE USED IF THE EXIT COMMAND HAS BEEN USED TO LEAVE NIE>

\*\*\*\*\*

EXAMPLE:

NIE> (A) ? EXIT

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 INDEFINITELY.

2.5.1 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 "IER", "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 VALID 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 OCCURRED  
WHEN TYPING A "DUMP" COMMAND WHERE  
OCTAL ADDRESSES ARE EXPECTED.

EXAMPLE OF A LOST PACKET ERROR DURING LOOPPAIR TESTING

-----  
CVNIA MRD ERR 00028 ON UNIT 00 TST 001 SUB 000 PC:064442

TIMEOUT OCCURRED - LOOP MESSAGE TYPE RECEIVE ASSIST  
FAILING TARGET NODE ADDRESS: AA-00-03-00-00-00  
FAILING ASSIST NODE ADDRESS: AA-00 03-00-00-02

EXAMPLE OF A LOST PACKET ERROR DURING PATTERN TESTING

-----  
CVNIA MRD ERR 00028 ON UNIT 00 TST 001 SUB 000 PC:63730

TIMEOUT OCCURRED 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	174440	;	[0]	CSR ADDRESS
.WORD	300	;	[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					



```

PROGRAM HEADER
927          .SBTTL PROGRAM HEADER
947
948 000000   .ENABL ABS,AMA
949          .      "      2000
950
951
952          .SBTTL PROGRAM MACROS
953
954
955          ;I$STACK MACRO
956          ; ---
957
958          ;***
959          ;THE I$STACK MACRO FACILITATES INITIALIZING THE R6 (HARDWARE) STACK
960          ;AND THE R5 (PARAMETER) STACK. R5 IS SET TO THE STACK LOW LIMIT
961          ;(STAKLO) AND THE PARAMETER STACK GROWS UPWARD. R6 IS SET TO THE
962          ;STACK HIGH LIMIT (STAKHI) AND THE HARDWARE STACK GROWS DOWNWARD.
963          ;IF THERE IS A STACK OVER-RUN, IT WILL BE DETECTED BY THE PREG14
964          ;ROUTINE.
965          ;---
966
967          .MACRO I$STACK STAKLO,STAKHI
968          .LIST
969          MOV STAKLO,R5      ;INITIALIZE THE PARAMETER STACK POINTER.
970          MOV STAKHI,SP     ;INITIALIZE THE HARDWARE STACK POINTER.
971          .NLIST
972          .ENDM I$STACK
973
974          ;PUSH MACRO
975          ; - - - - -
976
977          ;***
978          ;THE "PUSH" MACRO FACILITATES PUSHING ITEMS ON THE HARDWARE STACK.
979          ;UP TO SEVEN ITEMS MAY BE PLACED ON THE STACK WITH ONE MACRO.
980          ;---
981          .MACRO PUSH A,B,C,D,E,F,G
982
983          .IF NB G
984          .LIST
985          MOV G,(SP)
986          .NLIST
987          .ENDC
988
989          .IF NB F
990          .LIST
991          MOV F,(SP)
992          .NLIST
993          .ENDC
994
995          .IF NB E
996          .LIST
997          MOV E,(SP)
998          .NLIST
999          .ENDC
1000
1001          .IF NB D
1002          .LIST

```

```

1003                                MOV     D, (SP)
1004                                .NLIST
1005                                .ENDC
1006
1007                                .IF NB C
1008                                .LIST
1009                                MOV     C, (SP)
1010                                .NLIST
1011                                .ENDC
1012
1013                                .IF NB B
1014                                .LIST
1015                                MOV     B, (SP)
1016                                .NLIST
1017                                .ENDC
1018
1019                                .IF NB A
1020                                .LIST
1021                                MOV     A, (SP)
1022                                .NLIST
1023                                .ENDC
1024
1025                                .ENDM   PUSH
1026
1027                                ;POP MACRO
1028                                ;-----
1029
1030                                ;***
1031                                ;THE "POP" MACRO FACILITATES RETRIEVING ITEMS FROM THE HARDWARE STACK.
1032                                ;UP TO SEVEN ITEMS MAY BE RETRIEVED WITH ONE MACRO.
1033                                ;--
1034
1035                                .MACRO POP      A,B,C,D,E,F,G
1036
1037                                .IF NB A
1038                                .LIST
1039                                MOV     (SP),A
1040                                .NLIST
1041                                .ENDC
1042
1043                                .IF NB B
1044                                .LIST
1045                                MOV     (SP),B
1046                                .NLIST
1047                                .ENDC
1048
1049                                .IF NB C
1050                                .LIST
1051                                MOV     (SP),C
1052                                .NLIST
1053                                .ENDC
1054
1055                                .IF NB D
1056                                .LIST
1057                                MOV     (SP),D
1058                                .NLIST
1059                                .ENDC

```

G2

```

1060
1061      .IF NB E
1062      .LIST
1063
1064      .NLIST
1065      .ENDC
1066
1067      .IF NB F
1068      .LIST
1069
1070      .NLIST
1071      .ENDC
1072
1073      .IF NB G
1074      .LIST
1075
1076      .NLIST
1077      .ENDC
1078
1079      .ENDM POP
1080
1081      ;CALL MACRO
1082      ;-----
1083
1084      ;***
1085      ;THE CALL MACRO FACILITATES CALLING A SUBROUTINE VIA THE REGISTER
1086      ;PRESERVE ROUTINE (PREG14). IT PLACES THE PARAMETERS TO BE PASSED ON
1087      ;THE PARAMETER STACK. UP TO 7 PARAMETERS MAY BE PASSED USING THIS
1088      ;MACRO.
1089      ;---
1090
1091      .MACRO CALL    S A,B,C,D,E,F,G
1092
1093      .IF NB G
1094      .LIST
1095
1096      .NLIST
1097      .ENDC
1098
1099      .IF NB F
1100      .LIST
1101
1102      .NLIST
1103      .ENDC
1104
1105      .IF NB E
1106      .LIST
1107
1108      .NLIST
1109      .ENDC
1110
1111      .IF NB D
1112      .LIST
1113
1114      .NLIST
1115      .ENDC
1116

```

MOV (SP),E

MOV (SP),F

MOV (SP),G

MOV G,(R5)

MOV F,(R5)

MOV E,(R5)

MOV D,(R5)

```

1117      .IF NB C
1118      .LIST
1119
1120      .NLIST
1121      .ENDC
1122
1123      .IF NB B
1124      .LIST
1125
1126      .NLIST
1127      .ENDC
1128
1129      .IF NB A
1130      .LIST
1131
1132      .NLIST
1133      .ENDC
1134
1135      .LIST
1136
1137      JSR      R4,PREG14
1138      .WORD   S ANCHOR
1139
1140      .NLIST
1141      .ENDM   CALL
1142
1143      ;RETURN MACRO
1144      ;-----
1145
1146      ;***
1147      ;THE RETURN MACRO FACILITATES PASSING PARAMETERS BACK TO A CALLING
1148      ;ROUTINE. UP TO 7 PARAMETERS MAY BE PASSED BACK ON THE PARAMETER
1149      ;STACK.
1150      ;---
1151
1152      .MACRO  RETURN  A,B,C,D,E,F,G
1153
1154      .IF NB G
1155      .LIST
1156
1157      .NLIST
1158      .ENDC
1159
1160      .IF NB F
1161      .LIST
1162
1163      .NLIST
1164      .ENDC
1165
1166      .IF NB E
1167      .LIST
1168
1169      .NLIST
1170      .ENDC
1171
1172      .IF NB D
1173      .LIST
1174
1175      .NLIST

```

```

1174 .ENDC
1175
1176 .IF NB C
1177 .LIST
1178 MOV C,(R5).
1179 .NLIST
1180 .ENDC
1181
1182 .IF NB B
1183 .LIST
1184 MOV B,(R5).
1185 .NLIST
1186 .ENDC
1187
1188 .IF NB A
1189 .LIST
1190 MOV A,(R5).
1191 .NLIST
1192 .ENDC
1193 .LIST
1194 RTS PC
1195 .NLIST
1196 .ENDM RETURN
1197
1198 ;P$PUSH MACRO
1199 ;-----
1200
1201 ;***
1202 ;THE P$PUSH MACRO FACILITATES PUSHING PARAMETERS ON THE PARAMETER
1203 ;STACK. UP TO SEVEN ITEMS MAY BE PUSHED WITH ONE MACRO.
1204 ;---
1205
1206 .MACRO P$PUSH A,B,C,D,E,F,G
1207
1208 .IF NB G
1209 .LIST
1210 MOV G,(R5).
1211 .NLIST
1212 .ENDC
1213
1214 .IF NB F
1215 .LIST
1216 MOV F,(R5).
1217 .NLIST
1218 .ENDC
1219
1220 .IF NB E
1221 .LIST
1222 MOV E,(R5).
1223 .NLIST
1224 .ENDC
1225
1226 .IF NB D
1227 .LIST
1228 MOV D,(R5).
1229 .NLIST
1230 .ENDC

```

```

1231
1232      .IF NB C
1233      .LIST
1234      MOV      C,(R5).
1235      .NLIST
1236      .ENDC
1237
1238      .IF NB B
1239      .LIST
1240      MOV      B,(R5).
1241      .NLIST
1242      .ENDC
1243
1244      .IF NB A
1245      .LIST
1246      MOV      A,(R5).
1247      .NLIST
1248      .ENDC
1249
1250      .ENDM   P$PUSH
1251
1252      ;P$POP MACRO
1253      ; -----
1254
1255      ;***
1256      ;THE P$POP MACRO FACILITATES RETRIEVING PARAMETERS FROM THE PARAMETER
1257      ;STACK.  UP TO 7 PARAMETERS MAY BE RETRIEVED.
1258      ;
1259      ;THE ROUTINE THAT RECEIVES THE PARAMETERS HAS THE RESPONSIBILITY OF
1260      ;CLEANING UP THE PARAMETER STACK.  THIS MACRO IS AN AID TO MAKING
1261      ;A LOCAL COPY OF PASSED PARAMETERS AND CLEANING UP THE PARAMETER STACK.
1262      ;- -
1263
1264      .MACRO P$POP  A,B,C,D,E,F,G
1265
1266      .IF NB A
1267      .LIST
1268      MOV      (R5),A
1269      .NLIST
1270      .ENDC
1271
1272      .IF NB B
1273      .LIST
1274      MOV      -(R5),B
1275      .NLIST
1276      .ENDC
1277
1278      .IF NB C
1279      .LIST
1280      MOV      -(R5),C
1281      .NLIST
1282      .ENDC
1283
1284      .IF NB D
1285      .LIST
1286      MOV      (R5),D
1287

```



1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1384  
002000  
002000  
002000  
002001  
002002  
002003  
002004  
002005  
002006  
002007  
002010  
002010  
002011  
002011  
002012  
002012  
002014  
002014  
002016  
002016  
002020  
002020  
002022  
002022  
002024  
002024  
002026  
002026  
002030  
002030  
002032  
002032  
002034  
002034  
002036  
002036  
002040  
002040

002000  
103  
126  
116  
111  
101  
000  
000  
000  
101  
060  
000000  
000000  
117114  
000000  
002170  
000000  
117312  
000000  
000000  
000001  
000000  
002164

```

      .LIST
      .WORD 0 ; FLAG WORD
      .WORD 100000 ; SET VALID BIT (DSCR/HIGH ADDRESS WRD)
      .NLIST
      NEXT A,\B ; BUFFER LOW ADDRESS WORD
B=B+1
      .LIST
      .WORD C ; BUFFER SIZE
      .WORD 0 ; STATUS WORD 1
      .WORD 0 ; STATUS WORD 2
      .NLIST
.ENDM
; **
; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
; -
      POINTER BGNRPT
      HEADER CVNIA,A,0,0,1,PRI07

```

```

L$NAME::
      .ASCII /C/
      .ASCII /V/
      .ASCII /N/
      .ASCII /I/
      .ASCII /A/
      .BYTE 0
      .BYTE 0
      .BYTE 0
L$REV::
      .ASCII /A/
L$OEPO::
      .ASCII /O/
L$UNIT::
      .WORD 0
L$TIML::
      .WORD 0
L$MPCP::
      .WORD L$HARD
L$SPCP::
      .WORD 0
L$MPTP::
      .WORD L$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

```



PROGRAM MACROS

```

002042
002042 000340
002044
002044 000000
002046
002046 000000
002050
002050 003
002051 003
002052
002052 000000
002054 000000
002056
002056 000000
002060
002060 002122
002062
002062 102106
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 102126
002106
002106 103372
002110
002110 103370
002112
002112 102120
002114
002114 000000
002116
002116 000000
002120
002120 000000

```

```

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
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$PRT:: .WORD L$PROT
L$TEST:: .WORD 0
L$DLY:: .WORD 0
L$HIME:: .WORD 0

```

```

1385
1396
1397
1398
1399
1400

```

```

;
; NAMES OF DEVICES SUPPORTED BY PROGRAM
;
DEV TYP <DEQNA>

```

```

002122
002122 104 105 121
002125 116 101 000

```

```

L$DVTYP:: .ASCIZ /DEQNA

```

.EVEN

1401  
1407  
1408  
1409  
1410

: TEST DESCRIPTION  
:

DESCRPT <CVNID DEQNA NI EXERCISER>

L#DESC:: .ASCIZ /CVNID DEQNA NI EXER

CISER/

002130	103	126	116
002133	111	104	040
002136	104	105	121
002141	116	101	040
002144	116	111	040
002147	105	130	105
002152	122	103	111
002155	123	105	122
002160	000		

.EVEN

.EVEN

1411  
1412  
1419  
1420  
1421  
1422  
1423  
1434  
1435

:  
: FORMAT STATEMENTS USED IN PRINT CALLS  
:

1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451 002162  
002162 000001  
002164  
002164 103544  
1452

.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

1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468

.SBTTL DEFAULT HARDWARE P TABLE

;  
; THE DEFAULT HARDWARE P TABLE CONTAINS DEFAULT VALUES OF  
; THE TEST-DEVICE PARAMETERS. THE STRUCTURE OF THIS TABLE  
; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P TABLES,  
; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P TABLES.  
;

1469 002166  
002166 000003  
002170  
002170

BGNHW DFPTBL

.WORD L10000 L\$HW/2  
L\$HW::  
DFPTBL::

1470  
1471 002170 174440  
1472 002172 000300  
1473 002174 000240

.WORD 174440  
.WORD 300  
.WORD PRI05

; CSR  
; VECTOR  
; PRIORITY

1474  
1484  
1485 002176  
002176

ENDHW

L10000:

```

1487 .SBTTL SOFTWARE TABLE
1488
1489 ;**
1490 ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
1491 ; PROGRAM AS OPERATIONAL PARAMETERS. THESE PARAMETERS ARE
1492 ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
1493 ; AT RUN TIME.
1494 ;
1495 ;
1496 002176          BGNSW  SFPTBL
002176 000000
002200
002200
1497
1505
1506 002200          ENDSW
002200
1507
1508 .SBTTL GLOBAL EQUATES SECTION
1509
1510 ;**
1511 ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
1512 ; ARE USED IN MORE THAN ONE TEST.
1513 ;
1514 ;
1515          EQUALS
1516 ;
1517 ; BIT DIFINITIONS
1518 ;
1519 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
1520
1521 ;
1522 BIT9-- BIT09
1523 BIT8-- BIT08
1524 BIT7-- BIT07
1525 BIT6-- BIT06
1526 BIT5-- BIT05
1527 BIT4-- BIT04
1528 BIT3-- BIT03
1529 BIT2-- BIT02
1530
1531 002200

```

```

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          IDU==         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          MOE==     100000

```

```

1543
1544
1545           ;
1546           ;CSR DEFINITIONS
1547           BD      ==      BIT03      ;BOOT/DIAGNOSTIC ROM
1548           EL      ==      BIT09      ;ELOOP
1549           XC      ==      BIT12      ;TRANSCIEVER
1550           ;
1551           ;DESCRIPTOR FIELDS
1552           ;
1553           FLAGWORD==      0           ;FLAG WORD
1554           STAT3  ==      8.         ;STATUS WORD 1
1555           STAT4  ==      10.        ;STATUS WORD 2
1556           ;
1557           ;DESCRIPTOR BITS
1558           ;
1559           V       ==      BIT15      ;VALID BIT
1560           ;
1561           CITVECTOR==      4         ;OFFSET IN 4K ROM FOR VECTOR TO CITIZENSHIP TEST
1562           ;
1563           ; TERMINAL CONTROL CHARACTERS
1564           ;
1565           CR      == 15
1566           LF      == 12
1567           ;
1568           :::EQUATES FOR FLAG WORD:::
1569           CTARGET==0
1570           CASIST==1
1571           CSHCTR==2
1572           CCLNAD==4
1573           CCLNAL==8.
1574           CEXIT==16.
1575           ;
1576           :::CLOCK ENABLE VALUES TO BE LOADED IN CLK'S CSR:::
1577           ;
1578           LCLKEN==100
1579           PCLKEN==111
1580           PCLKCT==1600
1581           ;
1582           ; SPECIAL CLI CODES FOR "CHAR" ARGUMENT IN CLI CALLS
1583           ; (COMMAND LINE INTERPRETER DEFINITIONS)
1584           CLIERR= 0
1585           CLIEXT= 1
1586           CLIBF = 2
1587           CLIFIF= 3
1588           CLISPA= 4
1589           CLINUM= 5
1590           CLIALP= 6
1591           CLIALN= 7
1592           CLIOCT= 8.
1593           CLIDEC= 9.
1594           CLISTR= 10.
1595           ;
1596           ;DEFS FOR COMMAND LINE INTERPRETATION ACTION VALUES
1597           ;
1598           NULL=0
1599           HELP=1

```

1600	000002	NODE=2
1601	000003	BUILD=3
1602	000004	CRUN=4
1603	000005	CPATRN=5
1604	000006	CSAVE=6
1605	000007	SUMRY=7
1606	000010	IDENT=10
1607	000011	EXIT=11
1608	000012	NOTNUF=12
1609	000013	CEXADR=13
1610	000014	CSAVR4=14
1611	000015	CNODE=15
1612	000016	CALPHA=16
1613	000017	CONES=17
1614	000020	CZEROS=20
1615	000021	C1ALT=21
1616	000022	COALT=22
1617	000023	CCCITT=23
1618	000024	COPRSL=24
1619	000025	CTYPE=25
1620	000026	Csize=26
1621	000027	CCPYS=27
1622	000030	CNDADR=30
1623	000031	CNDAL=31
1624	000032	CRNALL=32
1625	000033	CLUPPR=33
1626	000034	CSHMSG=34
1627	000035	CCLMSG=35
1628	000036	CCNTR=36
1629	000037	CNDLOG=37
1630	000040	CFUNCT=40
1631	000041	CUNSAV=41
1632	000042	CCLSUM=42
1633	000043	CDIR=43
1634	000044	CDEFLLT=44
1635	000045	CUNSVF=45
1636	000046	RSPOND=46
1637		:
1638		:
1639		:
1640	000000	ALPHA==0
1641	000001	ONES==1
1642	000002	ZEROS==2
1643	000003	ONEALT==3
1644	000004	ZROALT==4
1645	000005	CCITT==5
1646	000006	OPRSEL==6
1647		:
1648		:
1649		:

MESSAGE TYPE VALUES

GLOBAL EQUATES FOR THE DEQNA DRIVER



```

1651                                     .SBTTL BIT DEFINITIONS
1652                                     ;*
1653                                     ; THIS SECTION CONTAINS BIT DEFINITIONS FOR THE QNA CSRS AND BUFFER
1654                                     ; DESCRIPTORS. ALL UNDEFINED CSR BITS ARE RESERVED AND/OR UNUSED.
1655                                     ; ALL $V DEFINITIONS REPRESENT BIT NUMBERS IN THE CSR, ALL $M
1656                                     ; DEFINITIONS ARE A MASK OF THAT CSR BIT
1657                                     ;
1658      000000      VRCVEN      ** 00      ; RECEIVER ENABLE BIT
1659      000001      MRCVEN      ** 1       ;
1660      000001      VRESET      ** 01      ; SOFTWARE RESET BIT
1661      000002      MRESET      ** 2       ;
1662      000002      VNXM        ** 02      ; NON EXISTENT MEMORY BIT
1663      000004      MNXM        ** 4       ;
1664      000003      VBOOT       ** 03      ; BOOT/DIAGNOSTIC ROM LOAD
1665      000010      MBOOT       ** 10      ;
1666      000004      VINVM       ** 04      ; XMIT LIST INVALID BIT
1667      000020      MINVM       ** 20      ;
1668      000005      VINVR       ** 05      ; RECEIVE LIST INVALID BIT
1669      000040      MINVR       ** 40      ;
1670      000006      VINTEN      ** 06      ; ENABLE INTERRUPT ON BITS 2, 7, OR 15
1671      000100      MINTEN      ** 100     ;
1672      000017      VRCVIN      ** 17      ; RECEIVE INTERRUPT REQUEST
1673      100000      MRCVIN      ** 100000  ;
1674      000010      VILOOP      ** 10      ; INTERNAL LOOPBACK BIT
1675      000400      MILOOP      ** 400     ;
1676      000011      VELOOP      ** 11      ; EXTERNAL LOOPBACK BIT
1677      001000      MELOOP      ** 1000    ;
1678      000012      VSANITY      ** 12      ; SANITY TIMER ENABLE BIT
1679      002000      MSANITY      ** 2000    ;
1680      000015      VCARRIER     ** 15      ; CARRIER SENSE BIT
1681      003000      MCARRIER     ** 3000    ;
1682      000007      VXMTIN       ** 07      ; XMIT INTERRUPT REQUEST
1683      000200      MXMTIN       ** 200     ;
1684                                     ;
1685                                     ; THE FOLLOWING DEFINITIONS ARE FOR THE DESCRIPTOR BITS WHICH ARE A PART
1686                                     ; OF THE BUFFER DESCRIPTOR LIST ENTRIES
1687                                     ;
1688      000006      VHIBYT       ** 6       ; XMIT BUFFER BEGINS ON HIGH BYTE
1689      000100      MHIBYT       ** 100     ;
1690      000007      VLOBYT       ** 7       ; XMIT BUFFER BEGINS ON LOW BYTE
1691      000200      MLOBYT       ** 200     ;
1692      000014      VXSETP       ** 14      ; PACKET IS A SETUP PACKET
1693      010000      MXSETP       ** 10000   ;
1694      000015      VEOM         ** 15      ; END OF MESSAGE
1695      020000      MEOM         ** 20000   ;
1696      000016      VCHADR       ** 16      ; ADRS IS ANOTHER LIST ADRS, NOT DATA
1697      040000      MCHADR       ** 40000   ;
1698      000017      VVALID       ** 17      ; VALID LIST ENTRY
1699      100000      MVALID       ** 100000  ;
1700                                     ;
1701                                     ; THE FOLLOWING DEFINITIONS ARE FOR THE FIRST STATUS WORD FROM THE QNA
1702                                     ; FOUND IN THE BUFFER DESCRIPTOR LIST ENTRIES WHEN A XMIT COMPLETES
1703                                     ;
1704      000004      VCOLLO       ** 4       ; XMIT COLLISION COUNTER BIT ZERO
1705      000004      SCOLL        ** 4       ; SIZE OF THE COLLISION COUNTER FIELD
1706      000010      VFAIL        ** 10      ; HEARTBEAT COLLISION CHECK FAILURE
1707      000400      MFAIL        ** 400     ;

```

```

1708      000013      VABORT      .. 13      ; XMISSION ABORTED, TOO MANY COLLISIONS
1709      004000      MABORT      .. 4000
1710      000013      VNOCAR      .. 13      ; NO CARRIER SEEN DURING XMISSION
1711      002000      MNOCAR      .. 2000
1712      000014      VLOSS       .. 14      ; CARRIER LOST DURING TRANSMISSION
1713      010000      MLOSS       .. 10000
1714      000016      VXUSED      .. 16      ; XMIT BUFFER IS USED/IN USE
1715      040000      MXUSED      .. 40000
1716      000016      VXERRS      .. 16      ; XMIT ERROR OCCURRED
1717      040000      MXERRS      .. 40000
1718      000015      VXLAST      .. 15      ; LAST XMIT BUFFER OF A MESSAGE
1719      100000      MXLAST      .. 100000
1720      ;
1721      ; FOLLOWING ARE THE DEFINITIONS FOR STATUS WORD 2 WRITTEN BY THE QNA. THESE
1722      ; ARE XMIT STATUS DEFINITIONS
1723      ;
1724      000000      VTDR        .. 0      ; TIME DOMAIN REFLECT. BIT 0
1725      000016      STDR        .. 16     ; SIZE OF TIME DOMAIN REFLECT. FIELD
1726      ;
1727      ; THE FOLLOWING DEFINITIONS ARE FOR STATUS WORD 1 ON A RECEIVE OPERATION
1728      ;
1729      000000      VOVF        .. 0      ; RECEIVER OVERFLOW OCCURRED
1730      000001      MOVF        .. 1
1731      000001      VCRC        .. 1      ; CRC ERROR, IGNORE IF DISCARD NOT SET
1732      000002      MCRC        .. 2
1733      000002      VFRAM      .. 2      ; RECEIVE FRAMING ERROR OCCURRED
1734      000004      MFRAM      .. 4
1735      000003      VSHORT     .. 3      ; PACKET OF LESS THAN TEN BYTES RCVD
1736      000010      MSHORT     .. 10
1737      000010      VRBL        .. 10     ; RECEIVED BYTE LENGTH BIT 8
1738      000003      SRBL        .. 3      ; SIZE OF RECEIVED BYTE LENGTH FIELD
1739      000013      VRUNT       .. 13     ; RUNT PACKET RECEIVED, DISCARD MESSAGE
1740      004000      MRUNT       .. 4000
1741      000014      VDISC      .. 14     ; DISCARD THE CURRENT MESSAGE
1742      010000      MDISC       .. 10000
1743      000015      VRLONG     .. 15     ; PACKET RECEIVED WAS TOO LARGE
1744      020000      MRLONG     .. 20000
1745      000015      VRSETP     .. 15
1746      020000      MRSETP     .. 20000
1747      000016      VRUSED      .. 16     ; RECEIVE BUFFER IS OR BEING USED
1748      040000      MRUSED      .. 40000
1749      000016      VRERRS      .. 16     ; RECEIVE ERROR OCCURRED
1750      040000      MRERRS      .. 40000
1751      000017      VRLAST     .. 17     ; LAST RECEIVE BUFFER OF A MESSAGE
1752      100000      MRLAST     .. 100000
1753      ;
1754      ; THE FOLLOWING DEFINITIONS ARE FOR STATUS WORD 2 AFTER A RECEIVE OPERATION
1755      ;
1756      000000      VRBL        .. 0      ; RECEIVED BYTE LENGTH
1757      000010      SRBL        .. 10     ; SIZE OF RECEIVED BYTE LENGTH FIELD
1758      ;
1759      ; THE FOLLOWING DEFINITIONS ARE FOR THE FLAG WORD IN THE BUFFER DESCRIPTOR
1760      ; ENTRIES
1761      ;
1762      000016      VFUSED      .. 16     ; THIS BUFFER IS OR BEING USED FLAG
1763      040000      MFUSED      .. 40000
1764      000016      VFERRS      .. 16     ; ERROR OCCURRED FLAG

```

```

1765      040000      MFERRS      == 40000
1766      000017      VFLAST      == 17      ; LAST BUFFER OF A MESSAGE
1767      100000      MFLAST      == 100000
1768      ;
1769      ;          ETHERNET PACKET OFFSETS
1770      ;
1771      000016      HEADER      == 14.      ; OFFSET (SIZE) TO END OF HEADER IN BYTES
1772      000000      DESTIN      == 0          ; DESTINATION ADDRESS
1773      000006      SOURCC      == 6          ; SOURCE ADDRESS
1774      000014      PROTOT      == 12.      ; PROTOCOL TYPE FIELD
1775      ;
1776      ; MISCELLANEOUS DEFS
1777      ;
1778      001100      XPKLEN      == 1100      ; XMT PACKET LENGTH IN BYTES
1779      001100      RPKLEN      == 1100      ; RCV PACKET LENGTH IN BYTES
1780      000050      TBLLEN      == 40.      ; NODE TABLE LENGTH (CHANGE STBLEN IF
1781      ;          THIS IS CHANGED, OR ELSE!)
1782      000132      STBLEN      == 90.      ; SUMMARY TABLE LENGTH (= 2.25 x TBLLEN)
1783      000004      FRDADR      == 4          ; OFFSET FOR MESSAGE HEADERS
1784      ;
1785      ; SYSTEM ID REPLY MESSAGE OFFSETS
1786      ;
1787      000020      RIFUNC      == 20
1788      000022      SIRCPT      == 22
1789      000027      SIVERS      == 27
1790      000030      SIECO      == 30
1791      000031      SIUECO      == 31
1792      000035      SIFNCT      == 35
1793      000042      SIADDR      == 42
1794      000053      SIDEV      == 53
1795      ;
1796      ; LOOP DIRECT OFFSETS
1797      ;
1798      000016      LDSKIP      == 16
1799      000020      LDFCT1      == 20
1800      000022      LDADR1      == 22
1801      000030      LDFCT2      == 30
1802      000032      LDADR2      == 32
1803      ;
1804      ; FULL ASSIST OFFSETS
1805      ;
1806      ;
1807      000016      FASKIP      == 16
1808      000020      FAFCT1      == 20
1809      000022      FAADR1      == 22
1810      000030      FAFCT2      == 30
1811      000032      FAADR2      == 32
1812      000040      FAFCT3      == 40
1813      000042      FAADR3      == 42
1814      000050      FAFCT4      == 50
1815      000052      FAADR4      == 52
1816      ;
1817      ; COUNTER OFFSETS
1818      ;
1819      000000      C.SIZ      == 0
1820      000002      C.SECONDS  == 2
1821      000004      C.PREC      == 4
  
```

```

1822      000010      C.MREC  == 10
1823      000014      C.RERB  == 14
1824      000016      C.RERR  == 16
1825      000020      C.RDAT  == 20
1826      000024      C.RMOB  == 24
1827      000030      C.RLIN  == 30
1828      000032      C.RLEX  == 32
1829      000034      C.PXMT  == 34
1830      000040      C.MXMT  == 40
1831      000044      C.PXM3  == 44
1832      000050      C.PXM2  == 50
1833      000054      C.PXMD  == 54
1834      000060      C.XDAT  == 60
1835      000064      C.XMOB  == 64
1836      000070      C.XABB  == 70
1837      000072      C.XABT  == 72
1838      000074      C.COLL  == 74
1839
1840      ; XMIT ABORT REASON MASK BIT DEFINITIONS
1841      ;
1842      000001      C.RTRY  == BIT00
1843      000002      C.LCAR  == BIT01
1844      000004      C.SHRT  == BIT02
1845      000010      C.OPEN  == BIT03
1846
1847      ; RECEIVE ERROR REASON MASK BIT DEFINITIONS
1848      ;
1849      000001      C.CRC   == BIT00
1850      000002      C.FRAME  == BIT01
1851      000004      C.MLEN  == BIT02
1852      000010      C.RUNT  == BIT03
1853
1854      ; ROUTINE FUNCTION CODES
1855      ;
1856      000000      BMRPVC  == 0      ; UPDATE RECEIVE COUNTERS
1857      000001      BMPXMT  == 1      ; UPDATE TRANSMIT COUNTERS
1858      000000      INIADR  == 0      ; INITIALIZE ADDRESS SETUP TABLE
1859      000001      ADDMUL  == 1      ; ADD MULTICAST ADDRESS TO SETUP TABLE
1860      000002      KILMUL  == 2      ; KILL MULTICAST ADDRESS IN SETUP TABLE
1861      000000      XMTSET  == 0      ; XMIT AN ADDRESS SETUP PACKET
1862      000001      XMTDAT  == 1      ; XMIT A DATA PACKET
1863
1864      ; FUNCTION CODES FOR LOOP-SERVER PACKETS
1865      ;
1866      000002      FORWRD  == 2
1867      000001      REPLY   == 1
1868      000005      IDFUNC  == 5
1869
1870      ; QNA DEVICE REGISTER OFFSETS
1871      ;
1872      000000      NETADD  == 0
1873      000004      LORCV   == 4
1874      000006      HIRCV   == 6
1875      000010      LOXMT   == 10
1876      000012      HIXMT   == 12
1877      000014      VECTOR  == 14
1878      000016      CSR     == 16

```

1879		:
1880		;BUFFER DESCRIPTOR OFFSET DEFINITIONS
1881		:
1882	000000	FLAG    ** 0
1883	000002	DESC    ** 2
1884	000002	HIADD   ** 2
1885	000004	LOADD   ** 4
1886	000006	WRDCNT  ** 6
1887	000010	STAT1   ** 10
1888	000012	STAT2   ** 12
1889		;-

```

1891          .SBTTL  GLOBAL DATA SECTION
1892          ;
1893          ;COMMAND LINE BUFFER, DATA LOCATIONS AND MESSAGES FOR ACTION ROUTINES
1894
1895 002200 000000  BLDFLG: .WORD 0          ; INDICATES THAT WE ARE IN TABLE BUILD MODE
1896 002202          CMDBUF: .BLKB 72.        ; BUFFER FOR OPERATOR COMMANDS
1897 002312 000000  KEYWD1: .WORD 0          ;
1898 002314 000000  KEYWD2: .WORD 0          ;
1899 002316 000000  ADRBUF: .WORD 0          ; BUFFER FOR NODE ADDRESS
1900 002320 000000          .WORD 0
1901 002322 000000          .WORD 0
1902 002324          STRBUF: .BLKB 18.        ; BUFFER FOR ALPHANUM. ADDRESS STRING
1903 002346          STRBU1: .BLKB 18.
1904 002370 000000  CBOADR: .WORD 0          ; POINTER FOR BEGINING OF ADDRESS STRING
1905 002372 000000  P#TYPE: .WORD 0          ; LOC. TO HOLD MESSAGE TYPE
1906 002374 000000  P#SIZE: .WORD 0          ; LOC. TO HOLD MESSAGE SIZE
1907 002376 000000  P#CPYS: .WORD 0          ; LOC. TO HOLD NO. OF MESSAGE COPIES
1908 002400 000000  P#PASS: .WORD 0          ; LOC. TO HOLD NO. OF PASSES
1909 002402 000000  LSRTRY: .WORD 0          ; NUMBER OF RETRIES FOR LOSS OF CARRIER
1910 002404 000000  NCRTRY: .WORD 0          ; NUMBER OF RETRIES FOR NO CARRIER
1911 002406 000000  NODTY: .WORD 0          ; LOC. TO HOLD NODE TYPE FOR NODE TABLE SETUP
1912 002410 000000  SLOT: .WORD 0          ; USED BY NODE TABLE SUBROUTINES
1913 002412          NODTBL: .BLKW TBLEN      ; SPACE FOR NODE TABLE
1914 002532 177777          .WORD 177777    ; FILL LAST FOUR BYTES OF TABLE WITH ONES
1915 002534 177777          .WORD 177777
1916 002536          SAVTBL: .BLKW TBLEN      ; SPACE FOR SAVE TABLE
1917 002656 177777  ILLADR: .WORD 177777    ; ILLEGAL ADDRESS FOR COMPARISON
1918 002660 177777          .WORD 177777    ; (MUST NOT BE PHYSICALLY SEPARATED FROM
1919 002662 177777          .WORD 177777    ; END OF SAVTBL)
1920 002664          STATBF: .BLKB 30        ; BUFFERS SUMMARY TABLE FOR SUMMARY REPORT
1921 002714          STATBL: .BLKW STBLEN    ; SPACE FOR SUMMARY TABLE
1922 003200 177777          .WORD 177777
1923 003202 000000  RSPFLG: .WORD 0          ; INDICATES IF WE ARE IN RESPONDER MODE
1924          ;
1925          ;COMMAND LINE TRAVERSE LOCATIONS (USED BY "P#TRV")
1926          ;
1927 003204 000000  P#BUFA: .WORD 0          ; LOC. TO HOLD ADDR. OF CMD LINE BUFFER
1928 003206 000000  P#TREE: .WORD 0          ; LOC. TO HOLD ADDR. OF PARSING TREE
1929 003210 000000  P#ACT: .WORD 0          ; LOC. TO HOLD ADDR. OF ACTION ROUTINE
1930 003212 000000  P#CNT: .WORD 0          ; LOC. TO BE A COUNTER LOCATION
1931 003214 000000  P#NUM: .WORD 0          ; LOC. TO HOLD NUMERIC VALUE FROM PARSE
1932 003216 000000  P#RADX: .WORD 0          ; LOC. TO HOLD RADIX(LO) & +/- (HI BYTE)
1933 003220 000          P#NNUF: .BYTE 0          ; RETURN -0 IF ENOUGH OF COMMAND FOUND
1934 003221 000          P#GDBD: .BYTE 0          ; RETURN CODE 0 IF NO ERROR FOUND
1935 003222 000          P#AERR: .BYTE 0          ; RETURN 0 IF 12 DIGIT ADDRESS ENTERED
1936 003223 000          P#MERR: .BYTE 0          ; RETURN -1 IF ERROR IN OPERATOR SELECTED
1937          ;MESSAGE INPUT OCCURED, 0 FOR GOOD INPUT
1938 003224 055120  HLPTAB: .WORD HELP1
1939 003226 055221          .WORD HELP2
1940 003230 055314          .WORD HELP3
1941 003232 055365          .WORD HELP4
1942 003234 055436          .WORD HELP5
1943 003236 055536          .WORD HELP6
1944 003240 055651          .WORD HELP7
1945 003242 055762          .WORD HELP8
1946 003244 056052          .WORD HELP9
1947 003246 056141          .WORD HELP10

```

```

1948 003250 056232 .WORD HELP11
1949 003252 056330 .WORD HLP115
1950 003254 056435 .WORD HELP12
1951 003256 056542 .WORD HELP13
1952 003260 056641 .WORD HELP14
1953 003262 056740 .WORD HELP15
1954 003264 057043 .WORD HELP16
1955 003266 057132 .WORD HELP17
1956 003270 057235 .WORD HELP18
1957 003272 057305 .WORD HELP19
1958 003274 057410 .WORD HELP20
1959 003276 057466 .WORD HELP21
1960 003300 057551 .WORD HELP22
1961 003302 057652 .WORD HELP23
1962 003304 057752 .WORD HELP24
1963 003306 060102 .WORD HELP25
1964 003310 060166 .WORD HELP26
1965 003312 060272 .WORD HELP27
1966 003314 060374 .WORD HELP28
1967 003316 060513 .WORD HELP29
1968 003320 060563 .WORD HELP30
1969 003322 000000 HLPEND: .WORD 0
1970 003324 061536 MSGTAB: .WORD MSGTY0
1971 003326 061544 .WORD MSGTY1
1972 003330 061551 .WORD MSGTY2
1973 003332 061557 .WORD MSGTY3
1974 003334 061564 .WORD MSGTY4
1975 003336 061571 .WORD MSGTY5
1976 003340 061577 .WORD MSGTY6
1977
1978
1979

```

;MESSAGE TYPE ASCII ADDRESS TABLE

; THIS SECTION DEFINES THE DATA PATTERNS USED BY THE EXERCISER

```

1980 003342 MSGCNT::
1981 003342 000130 MSG0C: .WORD EMSG0-MSG00
1982 003344 000001 MSG1C: .WORD EMSG1-MSG01
1983 003346 000001 MSG2C: .WORD EMSG2-MSG02
1984 003350 000001 MSG3C: .WORD EMSG3-MSG03
1985 003352 000001 MSG4C: .WORD EMSG4-MSG04
1986 003354 000100 MSG5C: .WORD EMSG5-MSG05
1987 003356 000000 MSG6C: .WORD 0
1988
1989 003360 MSGAD::
1990 003360 003376 .WORD MSG00
1991 003362 003526 .WORD MSG01
1992 003364 003527 .WORD MSG02
1993 003366 003530 .WORD MSG03
1994 003370 003531 .WORD MSG04
1995 003372 003532 .WORD MSG05
1996 003374 003632 .WORD OPSLBF
1997

```

; THE NUMBER OF BYTES IN EACH MESSAGE

```

1998 003376 040 041 042 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 \
003401 043 044 045
003404 046 047 050
003407 051 052 053
003412 054 055 057
003415 060 061 062
003420 063 064 065

```

	003423	066	067	070		
	003426	071	072	073		
	003431	074	075	076		
	003434	077	100	101		
	003437	102	103	104		
	003442	105	106	107		
	003445	110	111	112		
	003450	113	114	115		
	003453	116	117	120		
	003456	121	122	123		
	003461	124	125	126		
	003464	127	130	131		
	003467	132				
1999	003470	133	135	136	.ASCII	\(\)† ABCDEFGHIJKLMNOPQRSTUVWXYZ\ ; ALPHANUMERIC
	003473	055	101	102		
	003476	103	104	105		
	003501	106	107	110		
	003504	111	112	113		
	003507	114	115	116		
	003512	117	120	121		
	003515	122	123	124		
	003520	125	126	127		
	003523	130	131	132		
2000	003526				EMSG0::	
2001	003526	377			MSG01::	.BYTE 377 ; MESSAGE OF ALL ONES
2002	003527				EMSG1::	
2003	003527	000			MSG02::	.BYTE 0 ; MESSAGE OF ALL ZEROS
2004	003530				EMSG2::	
2005	003530	252			MSG03::	.BYTE 252 ; MESSAGE OF ALTERNATING ONES
2006	003531				EMSG3::	
2007	003531	125			MSG04::	.BYTE 125 ; MESSAGE OF ALTERNATING ZEROS
2008	003532				EMSG4::	
2009	003532				EMSG5::	
2010	003532	177603	157427	031011	MSG05::	.WORD 177603,157427,031011,047321,163715,105221 ; CCITT 511 BIT TEST PATTERN
	003540	047321	163715	105221		
2011	003546	143325	142304	040041	.WORD	143325,142304,040041,104116,052606,172334
	003554	104116	052606	172334		
2012	003562	105025	123754	111337	.WORD	105025,123754,111337,111523,030030,145064
	003570	111523	030030	145064		
2013	003576	137642	143531	063617	.WORD	137642,143531,063617,135075,066730,026575
	003604	135075	066730	026575		
2014	003612	052012	053627	070071	.WORD	052012,053627,070071,151172,165044,031605
	003620	151172	165044	031605		
2015	003626	166632	016147		.WORD	166632,016147
2016	003632				EMSG5::	
2017	003632				OPSLBF:	.BLKB 66. ;BUFFER FOR OPERATOR SELECTED MESSAGE TYPE
2018						
2019						
2020	003734	000000			CFLAG:	.WORD 0 ;ACTION ROUTINE CMD ARGUMENT FLAG
2021						
2022					;;CLOCK TABLES, EVENT LOG AND POINTERS	
2023	003736	000000			CLKCSR:	.WORD 0 ; CLOCK CSR ADDRESS
2024	003740	000000			CLKBR:	.WORD 0 ; CLOCK INTERRUPT LEVEL
2025	003742	000000			CLKVEC:	.WORD 0 ; CLOCK INTERRUPT VECTOR
2026	003744	000000			CLKHZ:	.WORD 0 ; CLOCK'S FREQUENCY IN HERTZ
2027	003746	000000			CLKEN:	.WORD 0 ; CLOCK'S CSR VALUE TO INTRPT. ENABLE IT
2028						



```

2029 003750 000000    TIMMIN: .WORD 0           ; PLACE TO KEEP TIME SINCE-START
2030 003752 000000    TIMSEC: .WORD 0
2031 003754 000000    TIMTCK: .WORD 0           ; PLACE TO KEEP NO. OF TICKS/SEC.
2032 003756 000000    TIMER1: .WORD 0          ; EVENT TIMER #1 (TICKS)
2033 003760 000000    TIMER2: .WORD 0          ; EVENT TIMER #2 (TICKS)
2034 003762 000000    TIMERS: .WORD 0          ; EVENT TIMER #3 (SECONDS)
2035                      .EVEN
2036
2037                      ;
2038                      ; BUS ADDRESSES OF QNA HARDWARE REGISTERS
2039 003764 000000    ROMAD1:: .WORD 0           ; ON READ, GET QNA ADDRESS. LOWER BYTE OF 1ST
2040 003766 000000                      .WORD 0           ; THREE REGISTERS.
2041 003770 000000    RLSTLO:: .WORD 0          ; LOW BYTE OF STARTING ADDRESS OF RECEIVE LIST
2042 003772 000000    RLSTHI:: .WORD 0          ; HIGH BYTE OF STARTING ADDRESS OF RECEIVE LIST
2043                      ;
2044                      ; DEFAULT PHYSICAL NODE ADRES
2045                      ;
2046 003774 000000    DEPADR:: .WORD 0           ; PHYSICAL ADDRESS WORD 0
2047 003776 000000                      .WORD 0           ; PHYSICAL ADDRESS WORD 1
2048 004000 000000                      .WORD 0           ; PHYSICAL ADDRESS WORD 2
2049                      ;
2050                      ; ADDRESS OF ALL ZEROES, USED TO INIT THE TARGET ADDRESS TABLE
2051                      ;
2052 004002 000000    NOADR:: .WORD 0
2053 004004 000000                      .WORD 0
2054 004006 000000                      .WORD 0
2055                      ;
2056                      ; CURRENT PHYSICAL ADDRESS
2057                      ;
2058 004010 000000    PHYADR:: .WORD 0           ; PHYSICAL ADDRESS WORD 0
2059 004012 000000                      .WORD 0           ; PHYSICAL ADDRESS WORD 1
2060 004014 000000                      .WORD 0           ; PHYSICAL ADDRESS WORD 2
2061                      ;
2062                      ; BROADCAST ADDRESS FF-FF-FF FF-FF FF
2063                      ;
2064 004016 177777    BRDADR: .WORD -1
2065 004020 177777                      .WORD 1
2066 004022 177777                      .WORD -1
2067                      ;
2068                      ; LOOP-SERVER MULTICAST ADDRESS CF 00 00 00 00-00
2069                      ;
2070 004024 000317    LPADR:: .WORD 317
2071 004026 000000                      .WORD 0
2072 004030 000000                      .WORD 0
2073                      ;
2074                      ; DECNET MULTICAST ADDRESS FOR CONSOLE ID AB 00 00 02 00 00
2075                      ;
2076 004032 000253    MCSTAD: .WORD 253
2077 004034 001000                      .WORD 1000
2078 004036 000000                      .WORD 0
2079                      ;
2080                      ; RCV RING DESCRIPTOR ENTRY FOR SETUP MODE
2081                      ;
2082 004040                      LORTGT:                      ; LOW ADRS OF TARGET ADRS TBL DESCRPTR
2083 004040 000000                      .WORD 0           ; AGAIN FOR LAST ENTRY
2084 004042 100000                      .WORD 100000      ; SET VALID BIT
2085 004044 023772                      .WORD RCVTGT      ; ADDRESS OF RECEIVE BUFFER

```

```

2086 004046 000000 .WORD 0
2087 004050 000000 .WORD 0
2088 004052 000000 .WORD 0
2089 004054 000000 .WORD 0 ; JUST ENOUGH OF THE NEXT DESCRIPTOR
2090 004056 000000 .WORD 0 ; TO INVALID LIST
2091
2092 ;
2093 ; XMIT RING DESCRIPTOR ENTRY FOR SETUP MODE
2094 004060
LOXTGT: ;
2095 004060 000000 .WORD 0 ; LOW ADRS OF TARGET ADRS TBL DESCRIPTR
2096 004062 100000 .WORD 100000 ; AGAIN FOR LAST ENTRY
2097 004064 023570 .WORD TGTADR ; SET VALID BIT
2098 004066 000000 .WORD 0 ; ADDRESS OF TARGET FIELD
2099 004070 000000 .WORD 0
2100 004072 000000 .WORD 0
2101 004074 000000 .WORD 0 ; JUST ENOUGH OF THE NEXT DESCRIPTOR
2102 004076 000000 .WORD 0 ; TO INVALID LIST
2103
2104 ;
2105 ; CITIZENSHIP TEST SETUPS
2106 ;RECEIVE BUFFERS
2107 000100 SMLSIZ == 64. ;SMALLEST SIZE IN BYTES OF ETHERNET PACKET
2108 000151 RBUFNR == 105. ;NUMBER OF RECEIVE BUFFERS
2109 004100 RBUFB: .BLKB RBUFNR*SMLSIZ ;100 RECEIVE BUFFERS, 64 BYTES EACH.
2110
2111 ;
2112 ;RECEIVE DESCRIPTORS
2113 ;
2114 ; THERE ARE RBUFNR+1 DESCRIPTORS. ALL OF THEM EXCEPT THE LAST ONE
2115 ; ARE DATA DESCRIPTORS, I.E. THE ONLY CHAIN DESCRIPTOR IS THE LAST
2116 ; ONE TO CREATE A RING. COMMUNICATIONS WITH THE TESTER IS
2117 ; HALF-DUPLEX, I.E. PACKETS ARE RECEIVED ONE AT A TIME UNTIL THE
2118 ; COMPLETE MESSAGE IS RECEIVED.
2119
2120 000014 RDESSZ == 12. ;SIZE IN BYTES OF EACH RECEIVE DESCRIPTOR
2121 002354 RDESAL == RDESSZ*RBUFNR ;SIZE IN BYTES OF ALL DESCRIPTORS
2122 023554 RDESC: .BLKB RDESAL ;STORAGE FOR RBUFNR RECEIVE DESCRIPTORS
2123 .BLKB RDESSZ ; PLUS CHAIN DESCRIPTOR TO GIVE A RING
2124 .EVEN
2125
2126 ;
2127 ;CITIZENSHIP WORK BUFFER
2128 ;
2129 ; THE BOOT/DIAG ROM LOADS INTO THE FIRST 4KB OF THE RECEIVE BUFFERS. THE 4KW
2130 ; WORK BUFFER EXTENDS FROM THERE THROUGH THE TRANSMIT BUFFERS INTO THE
2131 ; STORAGE AREA THAT IS CREATED BELOW.
2132 014100 CITWORK == RBUFB+4096. ;BEGINNING OF 4KW WORK BUFFER
2133 024100 CITWEND == CITWORK+4096. ;END+1 OF WORK BUFFER
2134
2135 ;
2136 ; TABLE OF TARGET ADDRESSES USED IN SETUP MODE
2137
2138 023570 TGTADR:
2139 023600 .BLKB 8. ; BYTE 0 OF ALL ADDRESSES 1
2140 023610 .BLKB 8. ; BYTE 1 "
2141 023620 .BLKB 8. ; BYTE 2 "
2142 023630 .BLKB 8. ; BYTE 3 "

```

2143 023640  
 2144 023650  
 2145 023660  
 2146 023670  
 2147 023700  
 2148 023710  
 2149 023720  
 2150 023730  
 2151 023740  
 2152 023750  
 2153 023760  
 2154 023770  
 2155 023770 000200  
 2156  
 2157  
 2158  
 2159 023772  
 2160 023772  
 2161 024002  
 2162 024012  
 2163 024022  
 2164 024032  
 2165 024042  
 2166 024052  
 2167 024062  
 2168 024072  
 2169 024102  
 2170 024112  
 2171 024122  
 2172 024132  
 2173 024142  
 2174 024152  
 2175 024162  
 2176  
 2177  
 2178  
 2179 024172  
 2180 024172 025674  
 2181 024174 026774  
 2182 024176 030074  
 2183 024200 031174  
 2184 024202 032274  
 2185 024204 033374  
 2186 024206 034474  
 2187 024210 035574  
 2188  
 2189  
 2190  
 2191 024212  
 2192 024212 024574  
 2193  
 2194  
 2195 024214 024236  
 2196 024216 024260  
 2197 024220 024236  
 2198 024222 024260  
 2199 024224 024236

```

      .BLKB 8.          ; BYTE 5 ' ' ' '
      .BLKB 8.          ; BYTE 6 ' ' ' '
      .BLKB 8.          ; BYTE 7 ' ' ' '
      .BLKB 8.          ; BYTE 0 OF ALL ADDRESSES 8 14
      .BLKB 8.          ; BYTE 1 ' ' ' '
      .BLKB 8.          ; BYTE 2 ' ' ' '
      .BLKB 8.          ; BYTE 3 ' ' ' '
      .BLKB 8.          ; BYTE 4 ' ' ' '
      .BLKB 8.          ; BYTE 5 ' ' ' '
      .BLKB 8.          ; BYTE 6 ' ' ' '
      .BLKB 8.          ; BYTE 7 ' ' ' '

TGTSIZ: .WORD TGTSIZ - TGTADR ; SIZE OF THE TARGET ADDRESS
; RECEIVE BUFFER FOR TARGET ADDRESSES LOOPED BACK IN SETUP MODE
;
RCVTGT:
      .BLKB 8.          ; BYTE 0 OF ALL ADDRESSES 1 7
      .BLKB 8.          ; BYTE 1 ' ' ' '
      .BLKB 8.          ; BYTE 2 ' ' ' '
      .BLKB 8.          ; BYTE 3 ' ' ' '
      .BLKB 8.          ; BYTE 4 ' ' ' '
      .BLKB 8.          ; BYTE 5 ' ' ' '
      .BLKB 8.          ; BYTE 6 ' ' ' '
      .BLKB 8.          ; BYTE 7 ' ' ' '
      .BLKB 8.          ; BYTE 0 OF ALL ADDRESSES 8 14
      .BLKB 8.          ; BYTE 1 ' ' ' '
      .BLKB 8.          ; BYTE 2 ' ' ' '
      .BLKB 8.          ; BYTE 3 ' ' ' '
      .BLKB 8.          ; BYTE 4 ' ' ' '
      .BLKB 8.          ; BYTE 5 ' ' ' '
      .BLKB 8.          ; BYTE 6 ' ' ' '
      .BLKB 8.          ; BYTE 7 ' ' ' '

; TABLE OF START ADDRESS OF RECEIVE RING BUFFERS
;
RRNGTB::
      .WORD RRG001
      .WORD RRG002
      .WORD RRG003
      .WORD RRG004
      .WORD RRG005
      .WORD RRG006
      .WORD RRG007
      .WORD RRG010

; TABLE OF START ADDRESS OF TRANSMIT RING BUFFERS
;
XRNGTB::
      .WORD XRG001

; POINTERS TO DESCRIPTOR RING ENTRIES
;
XRGSR1:: .WORD XRING ; FIRST ENTRY IN TRANSMIT RING
RRGSR1:: .WORD RRING ; FIRST ENTRY IN RECEIVE RING
XRGCUR:: .WORD XRING ; CURRENT ENTRY IN TRANSMIT RING
RRGCUR:: .WORD RRING ; CURRENT ENTRY IN RECEIVE RING
XRGNXT:: .WORD XRING ; NEXT ENTRY IN TRANSMIT RING

```

```

2200 024226 024260      RRGNXT::      .WORD  RRING      ; NEXT ENTRY IN RECEIVE RING
2201 024230 024236      XRGLST::      .WORD  XRING      ; LAST ENTRY IN TRANSMIT RING
2202 024232 024544      RRGLST::      .WORD  RRINGH     ; LAST ENTRY IN RECEIVE RING
2203 024234 024544      RRGPRV::      .WORD  RRINGH     ; PREVIOUS DESCRIPTOR POINTER
2204
2208 024236      ;
2209 024236      XRING::
      RRGFRM  XRG00,B,0
      .WORD  0
      .WORD  100000
      .WORD  XRG001
      .WORD  0
      .WORD  0
      .WORD  0
      .WORD  0
      ; FLAG WORD
      ; SET VALID BIT (DSCR/HIGH ADDRESS WRD)
      ; SEGMENT BUFFER ADDRESS
      ; BUFFER SIZE
      ; STATUS WORD 1
      ; STATUS WORD 2
2210 024252      XRINGH:
2211 024252      XRGINV:
      .WORD  0
      .WORD  0
      ; JUST ENOUGH OF THE NEXT DESCRIPTOR
      ; TO INVALIDATE LIST
2212 024252 000000
2213 024254 000000
2214 024256      RINGSIZ:
      .WORD  . - XRINGH
      ; SIZE OF DESCRIPTORS
2215 024256 000004
2219 024260      RRING::
      .REPT 7
      .NLIST
      RRGFRM  RRG00,B,-RPKLEN/2
      .LIST
      .ENDR
      ; ENTER BUFF ADD, BUF OFFSET, BUF SIZE
2220 000007
2221
2222
2223
2224
2225
2226 024404 000000      ;
      .WORD  0
      .WORD  100000
      .WORD  RRG010
      .WORD  -RPKLEN/2
      .WORD  0
      .WORD  0
      ; AGAIN FOR LAST ENTRY (FLAG WRD)
      ; DSCR/HIGH ADDRESS WORD
      ; LOW ADDRESS WORD
      ; BUFFER SIZE (TWO'S COMPLEMENT)
      ; STATUS WRD 1
      ; STATUS WRD 2
2227 024406 100000
2228 024410 035574
2229 024412 177340
2230 024414 000000
2231 024416 000000
2232
2236
2237 000007      ;
      .REPT 7
      .NLIST
      RRGFRM  RRG01,B,-RPKLEN/2
      .LIST
      .ENDR
      ; ENTER BUFF ADD, BUF OFFSET, BUF SIZE
2238
2239
2240
2241
2242 024544      RRINGH::
      .WORD  0
      .WORD  0
      .WORD  RRG020
      .WORD  -RPKLEN/2
      .WORD  0
      .WORD  0
      ; AGAIN FOR LAST ENTRY (FLAG WRD)
      ; LEAVE THIS ONE INVALID
      ; LOW ADDRESS WORD
      ; BUFFER SIZE (TWO'S COMPLEMENT)
      ; STATUS WRD 1
      ; STATUS WRD 2
2243 024544 000000
2244 024546 000000
2245 024550 046574
2246 024552 177340
2247 024554 000000
2248 024556 000000
2249 024560      RRGCHN::
      .WORD  0
      .WORD  140000
      .WORD  RRING
      .WORD  0
      .WORD  0
      .WORD  0
      ; DESCRIPTOR TO CHAIN TO BEGINNING OF RING
      ; ADDRESS OF START OF DESCRIPTOR RING
      ; BYTE COUNT (UNNECESSARY)
      ; STAT1
      ; STAT2
2250 024560 000000
2251 024562 140000
2252 024564 024260
2253 024566 000000
2254 024570 000000
2255 024572 000000
2256
2257
2258 024574      XRG001::      .BLKB  XPKLEN     ; XMIT RING BUFFER
2259

```

```

2260 025674          RRG001::      .BLKB  RPKLEN          ; RECEIVE RING BUFFERS
2261 026774          RRG002::      .BLKB  RPKLEN
2262 030074          RRG003::      .BLKB  RPKLEN
2263 031174          RRG004::      .BLKB  RPKLEN
2264 032274          RRG005::      .BLKB  RPKLEN
2265 033374          RRG006::      .BLKB  RPKLEN
2266 034474          RRG007::      .BLKB  RPKLEN
2267 035574          RRG010::      .BLKB  RPKLEN
2268 036674          RRG011::      .BLKB  RPKLEN
2269 037774          RRG012::      .BLKB  RPKLEN
2270 041074          RRG013::      .BLKB  RPKLEN
2271 042174          RRG014::      .BLKB  RPKLEN
2272 043274          RRG015::      .BLKB  RPKLEN
2273 044374          RRG016::      .BLKB  RPKLEN
2274 045474          RRG017::      .BLKB  RPKLEN
2275 046574          RRG020::      .BLKB  RPKLEN
2276
2277                ;
2278                ; CONSOLE ID INFORMATION
2279                ;
2279 047674          IDTDAT:
2280 047674          .BLKB  6          ; DESTINATION ADDRESS FIELD
2281 047702          .BLKB  6          ; SOURCE ADDRESS FIELD
2282 047710          001140        .WORD  1140        ; CONSOLE ID TYPE CODE
2283 047712          000034        .WORD  28.         ; NUMBER OF VALID DATA BYTES TO FOLLOW
2284 047714          007          .BYTE  7          ; SYSTEM ID FUNCTION CODE
2285 047715          000          .BYTE  0          ; PAD BYTE
2286 047716          046507        .WORD  46507       ; RECEIPT NUMBER ;; N.M. (CHANGED FROM 5115 TO 46507)
2287 047720          000000        .WORD  0          ; MOP VERSION TYPE
2288 047722          000          .BYTE  0          ; MOP VERSION LENGTH
2289 047723          000          .BYTE  0          ; MOP VERSION VERSION
2290 047724          000          .BYTE  0          ; MOP VERSION ECO
2291 047725          000          .BYTE  0          ; MOP VERSION USER ECO
2292 047726          000002        .WORD  2          ; FUNCTION TYPE
2293 047730          001          .BYTE  1          ; FUNCTION LENGTH
2294 047731          000          .BYTE  0          ; MAINTENANCE FUNCTIONS SUPPORTED 1
2295 047732          000          .BYTE  0          ; MAINTENANCE FUNCTIONS SUPPORTED 2
2296 047733          007          .BYTE  7          ; HARDWARE ADDRESS TYPE
2297 047734          000          .BYTE  0          ; SECOND BYTE OF ADDRESS TYPE
2298 047735          006          .BYTE  6          ; HARDWARE ADDRESS LENGTH
2299 047736          000          .BLKB  6          ; HARDWARE ADDRESS VALUE
2300 047744          144          .BYTE  100.       ; DEVICE TYPE FIELD
2301 047745          000          .BYTE  0          ; DEVICE TYPE FIELD BYTE 2
2302 047746          001          .BYTE  1          ; LENGTH OF DEV TYPE FIELD
2303 047747          005          .BYTE  5          ; DEQNA DEVICE TYPE
2304 047750          000          .BLKB  65.-49.    ; PAD THE REST OF THE PACKET ;; N.M. (CHANGED 43 TO 49)
2305                ;
2306 047770          IDTSIZ:
2307 047770          000074        .WORD  . - IDTDAT
2308                ;
2309                ; INFORMATION ABOUT THE CURRENT UNIT AS OBTAINED FROM THE HARDWARE P TABLE
2310                ;
2311                ; PTABLE INFO STORAGE FOR UNIT 0
2312                ;
2313 047772          000000        QNAADO::      .WORD  0          ; DEVICE ADDRESS FROM PTABLE
2314 047774          000000        QNAVCO::      .WORD  0          ; VECTOR ADDRESS FROM PTABLE
2315 047776          000000        QNAPRO::      .WORD  0          ; PRIORITY FROM PTABLE
2316                ;

```

```

2317 ; REGISTER CONTENT HOLDING LOCATIONS FOR UNIT NUMBER 0
2318 ;
2319 050000 NETADO:: .BLKW 2 ; HOLDS CONTENTS OF STATION ADDRESS ROM
2320 050004 CSRO:: .BLKW 1 ; HOLDS CONTENTS OF CSR REGISTER
2321 ;
2322 ; PTABLE INFO STORAGE FOR UNIT 1
2323 ;
2324 050006 000000 QNAAD1:: .WORD 0 ;CSR ADDRESS FROM PTABLE
2325 050010 000000 QNAVCI:: .WORD 0 ;VECTOR ADDRESS FROM PTABLE
2326 050012 000000 QNAPR1:: .WORD 0 ;PRIORITY FROM PTABLE
2327 ;
2328 ; CSR LOCATIONS FOR UNIT NUMBER 1
2329 ;
2330 050014 NETAD1:: .BLKW 2 ; HOLDS CONTENTS OF STATION ADDRESS ROM
2331 050020 CSR1:: .BLKW 1 ; CONTENTS OF CSR REGISTER
2332 ;
2333 ;
2334 ;
2335 050022 000000 FRESIZ:: .WORD 0 ;POINTER TO WORD CONTAINING SIZE OF FREE MEMORY
2336 050024 000000 FREMEM:: .WORD 0 ;POINTER TO FREE MEMORY SPACE
2337 050026 000000 UNIT:: .WORD 0 ;CURRENT UNIT NUMBER BEING TESTED
2338 ;
2339 ; SUMMARY DATA COUNTERS
2340 ;
2341 050030 000000 S.REC:: .WORD 0 ; MESSAGES RECEIVED
2342 050032 000000 S.NREC:: .WORD 0 ; MESSAGES NOT RECEIVED
2343 050034 000000 S.LEN:: .WORD 0 ; LENGTH ERRORS
2344 050036 000000 S.COMP:: .WORD 0 ; COMPARE ERRORS
2345 050040 000000 S.BYTE:: .WORD 0 ; BYTES COMPARED
2346 050042 000000 S.XFER:: .WORD 0 ; BYTES TRANSFERED
2347 ;
2348 ; SUMMARY LINK STATISTICS
2349 ;
2350 050044 UCB12: ; THIS AREA BUFFERS THE COUNTERS WHEN PRINTED
2351 050044 .BLKB 100 ; SIZE OF BUFFER FOR COUNTER
2352 050144 CNTRS:
2353 050144 000076 .WORD CNTEND - CNTRS ; SIZE OF COUNTER DATA BASE
2354 050146 000000 .WORD 0 ; SECONDS SINCE LAST ZEROED
2355 050150 .BLKW 2 ; PACKETS RECEIVED
2356 050154 .BLKW 2 ; MULTICAST PACKETS RECEIVED
2357 050160 000000 .WORD 0 ; RECEIVE ERROR REASON MASK
2358 050162 000000 .WORD 0 ; NUMBER OF PACKETS RECEIVED WITH ERRS
2359 050164 .BLKW 2 ; NUMBER OF DATA BYTES RECEIVED
2360 050170 .BLKW 2 ; MULTICAST DATA BYTES RECEIVED
2361 050174 000000 .WORD 0 ; RCV PACKETS LOST - INTERNAL BUFF ERR
2362 050176 000000 .WORD 0 ; RCV PACKETS LOST - LOCAL BUFFER ERR
2363 050200 .BLKW 2 ; PACKETS XMITTED
2364 050204 .BLKW 2 ; MULTICAST PACKETS XMITTED
2365 050210 .BLKW 2 ; XMIT SUCCESS AFTER 3 OR MORE TRIES
2366 050214 .BLKW 2 ; XMIT SUCCESS AFTER 2 TRIES
2367 050220 .BLKW 2 ; XMISSIONS THAT HAD TO BE DEFERRED
2368 050224 .BLKW 2 ; BYTES TRANSMITTED
2369 050230 .BLKW 2 ; NUMBER OF MULTICAST BYTES XMITTED
2370 050234 000000 .WORD 0 ; ABORTED XMISSIONS, REASON MASK
2371 050236 000000 .WORD 0 ; COUNT OF ABORTED TRANSMISSIONS
2372 050240 000000 .WORD 0 ; COUNT OF XMIT COLLISON CHECK FAILURES
2373 050242 CNTEND:

```

```

2374 ;
2375 ; DEQNA DRIVER AND ASSOCIATED SUBROUTINES DATA
2376 ;
2377 050242 000000 CITTST: .WORD 0 ; HOLDS STARTING ADDRESS OF CITIZENSHIP TEST AFTER LOADING F
ROM ROM
2378 050244 000000 CITSUM: .WORD 0 ; HOLDS ADDRESS OF THE ROM CHECKSUM AFTER LOADING
2379 050246 000000 CSRBUF: .WORD 0 ; BUFFERS THE CSR, BIS AND BIC HAPPEN TO THIS
2380 ; LOCATION WHICH IS THEN MOVED TO THE CSR.
2381 ; PREVENTS LOSING BITS THAT ARE "WRITE A 1 TO
2382 ; CLEAR"
2383 050250 000000 FLAG1: .WORD 0 ; FLAG TO INDICATE IF CITIZ. TEST HAS RUN.
2384 050252 000000 FATFLG: .WORD 0 ; FATAL ERROR FLAG
2385 050254 000000 PCEFLG: .WORD 0 ; PORT COMMAND ERROR FLAG
2386 050256 000000 NIRCNT: .WORD 0 ; QNA RECEIVE MESSAGE COUNTER
2387 050260 000000 XFLAG: .WORD 0 ; FRAME TRANSMITTED FLAG
2388 050262 000000 DNIFLG: .WORD 0 ; DONE INTERRUPT FLAG
2389 050264 000000 RBFCNT: .WORD 0 ; RECEIVE BUFFERS LOST COUNTER
2390 050266 000000 BCOUNT: .WORD 0 ; UNEXPLAINED INTERRUPTS COUNTER
2391 050270 000000 ERRFLG: .WORD 0 ; ERROR FLAG
2392 050272 000000 TIMEOUT: .WORD 0 ; TIME OUT COUNTER
2393 050274 000000 RETRYS: .WORD 0 ; COUNTER FOR FRAMES FAILING DUE TO RTRY ERROR
2394 050276 000000 RCVERR: .WORD 0 ; COUNTS NO. OF BUFFERS RECEIVED WITH ERRORS
2395 050300 000000 RCVBUF: .WORD 0 ; COUNTS NO. OF GOOD BUFFERS RECEIVED
2396 050302 010000 ROMSIZ: .WORD 4096. ; SIZE OF BOOT/DIAGNOSTIC ROM IN BYTES
2397 050304 000000 COUNT: .WORD 0 ; USED IN BLDBUF SUBROUTINE AS COUNTER
2398 050306 000220 PROT00: .WORD 000220 ; PROTOCOL TYPE FOR LOOPBACK MESSAGES
2399 050310 001140 PROT02: .WORD 001140 ; PROTOCOL TYPE FOR REMOTE CONSOLE
2400 050312 000000 SLFTST: .WORD 0 ; EXECUTE SELF TEST FLAG
2401 050314 000000 TEMP: .WORD 0 ; USED IN XMIT AS TEMPORARY STORAGE
2402 050316 000000 TEMP1: .WORD 0 ; USED FOR TEMPORARY STORAGE
2403 050320 000000 TEMP2: .WORD 0 ; USED FOR TEMPORARY STORAGE
2404 050322 000000 TEMP3: .WORD 0 ; USED FOR TEMPORARY STORAGE
2405 050324 000000 XFER: .WORD 0 ; STORES 'BYTES TRANSFERED'
2406 050326 000000 CPYCNT: .WORD 0 ; 'NO. OF COPIES' COUNTER FOR LOOPING
2407 050330 000000 PCCALL: .WORD 0 ; STORES PC OF CALLING ROUTINE FOR ERROR REPORTS
2408 050332 001100 BUFLN: .WORD XPKLEN ; STORES TRANSMIT BUFFER LENGTH
2409 050334 000000 CMPBUF: .WORD 0 ; STORES LOCATION OF DATA BUFFER TO BE COMPARED
2410 050336 000000 LAPCNT: .WORD 0 ; COUNTS HOW MANY LAPS "CURRENT RECEIVE
2411 ; DESCRIPTOR" POINTER IS AHEAD OF THE
2412 ; "NEXT RECEIVE DESCRIPTOR" POINTER
2413 ;
2414 ; REQUEST ID MESSAGE FORMAT
2415 ;
2416 050340 REQID: .
2417 050340 000003 .WORD 3 ; BYTE COUNT (=3 FOR REQUEST ID)
2418 050342 000005 .WORD IDFUNC ; FUNCTION CODE FOR REQUEST ID
2419 050344 051115 .WORD "MR" ; RECEIPT NUMBER
2420 ;
2421 ; LOOP DIRECT MESSAGE
2422 ;
2423 .EVEN
2424 050346 LOPDIR: .
2425 050346 000000 .WORD 0 ; SKIP COUNT
2426 050350 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2427 050352 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2428 050360 000001 .WORD 1 ; FUNCTION = REPLY
2429 050362 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2430 ;

```

```

2431 ; TRANSMIT ASSIST MESSAGE
2432 ;
2433 050370 ;TASIST::
2434 050370 000000 .WORD 0 ; SKIP COUNT
2435 050372 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2436 050374 000000 000000 000000 .WORD 0,0,0 ; TRANSMIT ASSIST ADDRESS
2437 050402 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2438 050404 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2439 050412 000001 .WORD REPLY ; FUNCTION = REPLY
2440 050414 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2441 ;
2442 ; RECEIVE ASSIST MESSAGE
2443 ;
2444 050422 ;RASIST::
2445 050422 000000 .WORD 0 ; SKIP COUNT
2446 050424 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2447 050426 000000 000000 000000 .WORD 0,0,0 ; TRANSMIT ASSIST ADDRESS
2448 050434 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2449 050436 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2450 050444 000001 .WORD REPLY ; FUNCTION = REPLY
2451 050446 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2452 ;
2453 ; FULL ASSIST MESSAGE
2454 ;
2455 050454 ;FASIST::
2456 050454 000000 .WORD 0 ; SKIP COUNT
2457 050456 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2458 050460 000000 000000 000000 .WORD 0,0,0 ; TARGET NODE ADDRESS
2459 050466 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2460 050470 000000 000000 000000 .WORD 0,0,0 ; ASSIST NODE ADDRESS
2461 050476 000002 .WORD FORWRD ; FUNCTION = FORWARD DATA
2462 050500 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS
2463 050506 000001 .WORD REPLY ; FUNCTION = REPLY
2464 050510 000000 000000 000000 .WORD 0,0,0 ; LOCAL NODE ADDRESS

```



```

2466          .SBTTL  COMMAND LINE ACTION TREE
2467 ;SAMPLE CLI TREE NODE (ALWAYS AT LEAST 1 WORD)
2468 ;
2469 ;-----
2470 ; ! ACTION ! CHAR CODE !
2471 ;-----
2472 ; ! MISS DISPLACEMENT !          ONLY IF "MISS" ARGUMENT DEFINED
2473 ;-----
2474 ; ! NEXT MODE DISPLMNT !          ONLY IF "ASCII" ARGUMENT DEFINED
2475 ;-----
2476 ; ! ASCIZ MATCH STRING !          ONLY IF "ASCII" ARGUMENT DEFINED
2477 ; !           (.EVEN)           !
2478 ;-----
2479 ;
2480 ; FIRST KEYWORD
2481 ;
2482 CLITRE:
2483 CLI      CLISPA,0,N10$          ;SKIP ANY LEADING SPACES
2484 N10$:    CLI      <' ?>,HELP,N12$          ;IS THE FIRST NON-SP CHAR. A ?
2485         CLI      CLIEXI,0          ; IF YES DO "HLP" AND EXIT
2486 N12$:    CLI      CLISTR,HELP,N14$,<'HELP'> ;ELSE IS FIRST WORD A "HELP"
2487         CLI      CLIEXI,0          ; IF YES DO "HLP" AND EXIT
2488 N14$:    CLI      CLISTR,NOTNUF,N16$,<'NODE'> ;ELSE IS FIRST WORD A "NODE"
2489         CLI      CLIBR,0,N80$          ; IF YES, BR N80$
2490 N16$:    CLI      CLISTR,BUILD,N18$,<'BUILD'> ;ELSE IS FIRST WORD A "BUILD"
2491         CLI      CLIEXI,0          ; IF YES DO "BUILD" AND EXIT
2492 N18$:    CLI      CLISTR,NOTNUF,N20$,<'RUN'> ;ELSE IS FIRST WORD A "RUN"
2493         CLI      CLIBR,0,N180$          ; IF YES, BR N180$
2494 N20$:    CLI      <'S>,NOTNUF,N25$          ;ELSE IS FIRST CHAR. A "S"
2495         CLI      CLISTR,0,N22$,<'HOW'> ; IF YES IS REST OF WORD "HOW"
2496         CLI      CLIBR,0,N100$          ; IF YES, BR N100$
2497 N22$:    CLI      CLISTR,SUMMARY,N23$,<'SUMMARY'> ; ELSE IS REST OF WORD "SUMMARY"
2498         CLI      CLIEXI,0          ; IF YES, DO "SUMM" AND EXIT
2499 N23$:    CLI      CLISTR,CSAVE,N24$,<'AVE'> ; ELSE IS REST OF WORD "AVE"
2500         CLI      CLIEXI,0          ; IF YES, DO "SAVE" AND EXIT
2501 N24$:    CLI      CLIERR,0          ; ELSE "ILL COMMAND"
2502         CLI      CLIEXI,0          ; EXIT
2503 N25$:    CLI      CLISTR,NOTNUF,N26$,<'CLEAR'> ;ELSE IS FIRST WORD A "CLEAR"
2504         CLI      CLIBR,0,N120$          ; IF YES, BR N120$
2505 N26$:    CLI      CLISTR,NOTNUF,N28$,<'IDENTIFY'> ;ELSE IS FIRST WORD "IDENTIFY"
2506         CLI      CLIBR,0,N140$          ; IF YES, GET ADDRS. BR N140$
2507 N28$:    CLI      CLISTR,NOTNUF,N29$,<'MESSAGE'> ;ELSE IS FIRST WORD "MESSAGE"
2508         CLI      CLIBR,0,N160$          ; IF YES, BR N160$
2509 N29$:    CLI      CLISTR,CUNSAV,N30$,<'UNSAVE'> ;ELSE IS FIRST WORD "UNSAVE"
2510         CLI      CLIBR,0,N210$          ; IF YES, BR TO N210$
2511 N30$:    CLI      CLISTR,EXIT,N31$,<'EXIT'> ;ELSE IS FIRST WORD 'EXIT'
2512         CLI      CLIEXI,0          ; IF YES EXIT
2513 N31$:    CLI      CLISTR,NOTNUF,N32$,<'FUNCTION'> ;ELSE IS FIRST WORD "FUNCTION"
2514         CLI      CLIBR,0,N200$          ; IF YES, BR N200$
2515 N32$:    CLI      CLISTR,RSPOND,N33$,<'RESPOND'> ;ELSE IS FIRST WORD "RESPOND"
2516         CLI      CLIEXI,0          ; IF YES LEAVE
2517 N33$:    CLI      CLIERR,0          ; OTHERWISE 'ILL CMD".
2518         CLI      CLIEXI,0          ; EXIT
2519 ;
2520 ; SECOND KEYWORD (ADR/TYPE) FOR NODE COMMAND
2521 ;
2522 N80$:    CLI      CLISPA,0,N81$          ;SKIP ANY LEADING SPACES

```

```

2523 051106 N81$: CLI CLIBR,CSAVR4,N82$ ;SAVE STRING POINTER LOCATION
2524 051112 N82$: CLI CLIBR,NODE,N90$ ;PARSE THROUGH ADDRESS,CHECK
2525 ; ;FOR TARGET OR ASSIST, DO NODE
2526 051116 N90$: CLI CLIBIF,0,N32$ ;TAKE ERROR BRANCH IF ERROR EXISTS
2527 051122 N95$: CLI CLIEXI,0 ;EXIT
2528 ;
2529 ; SECOND KEYWORD FOR SHOW COMMAND
2530 ;
2531 051124 N100$: CLI CLISPA,0,N101$ ;SKIP LEADING SPACES
2532 051130 N101$: CLI CLISTR,CNODE,N102$, <'NODES'> ;IS NEXT WORD "NODES"
2533 051144 ; CLI CLIBR,0,N110$ ; IF YES, SET FLAG, BR N110$
2534 051150 N102$: CLI CLISTR,CSHMSG,N104$, <'MESSAGE'> ;ELSE IS NEXT WORD "MESSAGE"
2535 051166 ; CLI CLIBR,0,N110$ ; IF YES, SET FLAG, BR N110$
2536 051172 N104$: CLI CLISTR,CCNTR,N106$, <'COUNTERS'> ;ELSE IS NEXT WORD "COUNTERS"
2537 051212 ; CLI CLIBR,0,N110$ ; GO TO COUNTERS ROUTINE
2538 051216 N106$: CLI CLIBR,0,N32$ ;ELSE ILL COMMAND"
2539 051222 N110$: CLI CLIEXI,0 ;EXIT
2540 ;
2541 ; SECOND KEYWORD FOR CLEAR COMMAND
2542 ;
2543 051224 N120$: CLI CLISPA,0,N121$ ;SKIP LEADING SPACES
2544 051230 N121$: CLI CLISTR,0,N130$, <'NODE'> ;IS NEXT WORD "NODE"
2545 051244 ; CLI CLISPA,0,N122$ ; IF YES SKIP SPACES
2546 051250 N122$: CLI <'/'>,CSAVR4,N32$ ; LOOK FOR DELIMITER, ELSE "ILL COM"
2547 051254 ; CLI <'A'>,0,N123$ ; IS NEXT CHAR. AN "A"
2548 051260 ; CLI CLISTR,CNODAL,N124$, <'LL'> ; IF YES, IS WORD "ALL"
2549 051272 ; CLI CLIBR,0,N135$ ; IF YES, SET FLAG, BR N135$
2550 051276 N123$: CLI <'N'>,0,N124$ ; ELSE IS NEXT CHAR. AN "N"
2551 051302 ; CLI CLIDEC,0,N32$ ; IF YES, STORE NODE LOGICAL NAME
2552 051306 ; CLI CLIBR,CNDLOG,N135$ ; BR TO CLR. NODE LOGICAL ROUTINE
2553 051312 N124$: CLI CLIBR,CEXADR,N126$ ; ELSE, EXTRACT ADDRESS
2554 051316 N126$: CLI CLIBR,CNDADR,N135$ ; SET FLAG, BR N135$
2555 051322 N130$: CLI CLISTR,CCLMSG,N132$, <'MESSAGE'> ;ELSE IS NEXT WORD "MESSAGE"
2556 051340 ; CLI CLIBR,0,N135$ ; IF YES, SET FLAG, BR N135$
2557 051344 N132$: CLI CLISTR,CCLSUM,N134$, <'SUMMARY'> ;ELSE IS NEXT WORD "SUMMARY"
2558 051362 ; CLI CLIBR,0,N135$ ; IF YES, CLEAR TABLE AND EXIT
2559 051366 N134$: CLI CLIERR,0 ;ELSE, "ILL COMMAND".
2560 051370 N135$: CLI CLIEXI,0 ;EXIT
2561 ;
2562 ; ADDRESS FOR IDENTIFY COMMAND
2563 ;
2564 051372 N140$: CLI CLISPA,0,N141$ ;SKIP LEADING SPACES
2565 051376 N141$: CLI CLIBR,CSAVR4,N142$ ;SAVE POINTER TO FIRST CHAR. OF ADDRESS
2566 051402 N142$: CLI CLIALN,0,N32$ ;CHECK THAT ADDRESS HAS LEGAL CHAR.S
2567 051406 ; CLI CLIBR,CEXADR,N143$ ;GET ADDRESS
2568 051412 N143$: CLI CLIEXI,IDENT ;DO "IDENTIFY", EXIT
2569 ;
2570 ; REMAINING COMMAND LINE FOR MESSAGE COMMAND
2571 ;
2572 051414 N160$: CLI CLISPA,0,N161$ ;SKIP LEADING SPACES
2573 051420 N161$: CLI <'/'>,0,N178$ ;IF CHAR. "/" , CONT., ELSE BR N178$
2574 051424 ; CLI CLISTR,0,N170$, <'TYPE'> ;IS NEXT WORD "TYPE"
2575 051440 ; CLI <'='>,0,N32$ ; IF YES, FOLLOWED BY '='?
2576 051444 ; CLI CLISTR,CALPHA,N162$, <'ALPHA'> ; IF "ALPHA", SET FLAG
2577 051460 ; CLI CLIBR,0,N168$ ; CONTINUE AT N168$
2578 051464 N162$: CLI CLISTR,CONES,N163$, <'ONES'> ; IF "ONES", SET FLAG
2579 051500 ; CLI CLIBR,0,N168$ ; CONTINUE AT N168$

```

```

2580 051504      N163$: CLI      CLISTR,CZEROS,N164$,<'ZEROS'>      ; IF "ZEROS", SET FLAG
2581 051520      CLI      CLIBR,0,N168$                          ; CONTINUE AT N168$
2582 051524      N164$: CLI      CLISTR,C1ALT,N165$,<'1ALT'>      ; IF "1ALT", SET FLAG
2583 051540      CLI      CLIBR,0,N168$                          ; CONTINUE AT N168$
2584 051544      N165$: CLI      CLISTR,COALT,N166$,<'OALT'>      ; IF "OALT", SET FLAG
2585 051560      CLI      CLIBR,0,N168$                          ; CONTINUE AT N168$
2586 051564      N166$: CLI      CLISTR,CCITT,N167$,<'CCITT'>      ; IF "CCITT", SET FLAG
2587 051600      CLI      CLIBR,0,N168$                          ; CONTINUE AT N168$
2588 051604      N167$: CLI      <'>,CSAVR4,N32$                    ; IF "OPERATOR", SET FLAG
2589 051610      CLI      CLIBR,COPRSL,N168$                      ; AND INPUT SPECIFIED STRING
2590 051614      N168$: CLI      CLIBR,CTYPE,N160$                ; DO "TYPE", CHECK FOR MORE INPUT
2591 051620      N170$: CLI      CLISTR,0,N175$,<'SIZE'>          ; ELSE IS WORD "SIZE"
2592 051634      CLI      <'>,0,N32$                              ; IF YES, FOLLOWED BY "="?
2593 051640      CLI      CLIDEC,CSIZE,N32$                       ; STORE NUMBER IN M$SIZE
2594 051644      CLI      CLIBR,0,N160$                          ; CHECK FOR MORE INFO
2595 051650      N175$: CLI      CLISTR,0,N32$,<'COPIES'>        ; ELSE IS WORD "COPIES"
2596 051666      CLI      <'>,0,N32$                              ; IF YES, FOLLOWED BY "="?
2597 051672      CLI      CLIDEC,CCPYS,N32$                      ; STORE NUMBER IN M$CPYS
2598 051676      CLI      CLIBR,0,N160$                          ; CHECK FOR MORE INFO
2599 051702      N178$: CLI      CLIBR,0,N32$                    ; ELSE "ILL COMMAND"
2600
2601      ; SECOND KEYWORD FOR RUN COMMAND
2602
2603 051706      N180$: CLI      CLISPA,0,N181$                      ; SKIP LEADING SPACES
2604 051712      N181$: CLI      CLISTR,CLUPPR,N182$,<'LOOPPAIR'> ; IS NEXT WORD "LOOPPAIR"
2605 051732      CLI      CLIBR,0,N185$                          ; IF YES, SET "LOOPPAIR" FLAG
2606 051736      N182$: CLI      CLISTR,CRNALL,N183$,<'ALL'>      ; ELSE IS NEXT WORD "ALL"
2607 051750      CLI      CLIBR,0,N185$                          ; IF YES, SET "ALL" FLAG
2608 051754      N183$: CLI      CLISTR,CDIR,N184$,<'DIRECT'>    ; ELSE IS NEXT WORD "DIRECT"
2609 051772      CLI      CLIBR,0,N185$                          ; IF YES, SET "DIRECT" FLAG
2610 051776      N184$: CLI      CLISTR,CPATRN,N32$,<'PATTERN'> ; ELSE IS NEXT WORD "PATTERN"
2611 052014      N185$: CLI      CLIBR,CDEFLT,N186$                ; SEE IF DEFAULT OF 1 PASS
2612 052020      N186$: CLI      CLISTR,0,N32$,<' /PASS'>        ; PARSE THROUGH SWITCH
2613 052034      CLI      <'>,0,N32$                              ; PARSE THROUGH "="
2614 052040      CLI      CLIDEC,0,N32$                            ; GET PASS COUNT
2615 052044      N190$: CLI      CLIEXI,CRUN                       ; RUN TEST AND EXIT
2616
2617      ; REMAINING COMMAND LINE F. R FUNCTION COMMAND
2618
2619 052046      N200$: CLI      CLISPA,0,N201$                      ; SKIP SPACES
2620 052052      N201$: CLI      CLIOCT,CFUNCT,N32$                ; GET OCTAL NUMBER AND DO FUNCT
2621 052056      CLI      CLIEXI,0                                  ; EXIT
2622
2623      ; REMAINING COMMAND LINE FOR UNSAVE COMMAND
2624
2625 052060      N210$: CLI      CLISPA,0,N212$                      ; SKIP SPACES
2626 052064      N212$: CLI      <' />,0,N215$                    ; PARSE THROUGH '/'
2627 052070      CLI      CLIEXI,CUNSVF                            ; SAVE POINTER TO FILE NAME
2628 052072      N215$: CLI      CLIEXI,0

```

```

2630                                     .SBTTL GLOBAL TEXT SECTION
2631                                     .NLIST BIN
2632                                     ;**
2633                                     ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
2634                                     ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
2635                                     ; MORE THAN ONE TEST.
2636                                     ;
2637                                     ;--
2638 052074                                ERRRTBL                                L$ERRRTBL::
052074                                .WORD 0
052074 ERRRTYP::                          .WORD 0
052076 ERRNBR::                          .WORD 0
052100 ERRMSG::                          .WORD 0
052102 ERRBLK::                          .WORD 0
2639 052104 CLI$PM: .ASCIZ <12><15>/NIE>/ ;NIE PROMPT
2640 052113 CLIERM: .ASCIZ /%N%A?ILL CMD-BAD SYNTAX?/
2641 052144 CLINUF: .ASCIZ /%N%A?INCOMPLETE COMMAND?/
2642 052175 CLINBG: .ASCIZ /%N%A?NUMBER TOO BIG?/
2643 052222 CLIBRX: .ASCIZ /%N%A?BAD RADIX?/
2644 052242 L5060: .ASCIZ <CR><LF>\IS YOUR WALL OUTLET POWER 60 HERTZ?\
2645 052310 LDRESP: .ASCIZ /%N%ANODE %T%A HAS RESPONDED./
2646 052345 RECERR: .ASCIZ /%N%APACKET RECEIVED WITH QNA ERROR./
2647 052411 RTRYER: .ASCIZ /%N%ATRANSMISSION ABORTED -- EXCESSIVE COLLISIONS./
2648 052473 BLDMSG: .ASCIZ /%N%D2%A NODE ADDRESSES ADDED, ELAPSED TIME: %D2%A MINUTE./
2649 052565 ILADMS: .ASCIZ /%N%ACANNOT USE BROADCAST ADDRESS (FF-FF-FF-FF-FF-FF)/
2650 052651 ILADM1: .ASCIZ /%N%AFOR LOOP TESTING. ADDRESS WAS NOT ADDED TO NODE TABLE.%N/
2651 052746 CADRER: .ASCIZ /%N%PLEASE ENTER TWELVE HEXADECIMAL DIGITS./
2652 053022 CADERR: .ASCIZ /%N%ATWELVE HEX-DIGITS REQUIRED FOR ADDRESS./
2653 053076 MULSTR: .ASCIZ /%N%AA ZERO LENGTH STRING WAS ENTERED./
2654 053144 NODADR: .ASCIZ /%N%S2%/
2655 053154 LOGNAM: .ASCIZ /%S14%AN%D2/
2656 053167 NODTYP: .ASCIZ /%S14%/
2657 053176 NTBHDR: .ASCIZ \%N%ANODE PHYSICAL ADDRESS      NODE LOGICAL NAME      TYPE(A/T)\
2658 053274 TABFUL: .ASCIZ /%N%ATHE %T%A TABLE IS FILLED TO CAPACITY!//
2659 053346 TABEMT: .ASCIZ /%N%ATHE %T%A TABLE IS CURRENTLY EMPTY!//
2660 053415 NOD: .ASCIZ /NODE/
2661 053422 SUMM: .ASCIZ /SUMMARY/
2662 053432 CLRMSG: .ASCIZ /%N%ATHE MESSAGE PARAMETERS HAVE BEEN RESET TO://
2663 053511 CPYLMT: .ASCIZ /%N%ATHE NUMBER OF COPIES MUST BE BETWEEN 1 AND 255./
2664 053575 SIZLMT: .ASCIZ /%N%ATHE MESSAGE SIZE MUST BE BETWEEN 32 AND 512 BYTES./
2665 053664 NOCMPR: .ASCIZ /%N%ATHE ADDRESS MARKED FOR DELETION WAS NOT IN THE TABLE./
2666 053756 UNBOND: .ASCIZ /%N%AA UNBOUNDED "OPERATOR INPUT" STRING WAS ENTERED./
2667 054044 ADRDEL: .ASCIZ /%N%ATHE ADDRESS HAS BEEN DELETED FROM THE NODE TABLE./
2668 054132 LOGDEL: .ASCIZ /%N%ANODE %D1%A HAS BEEN DELETED FROM THE NODE TABLE./
2669 054220 TABCLR: .ASCIZ /%N%ATHE %T%A TABLE HAS BEEN CLEARED./
2670 054265 UNSMSG: .ASCIZ /%N%ATHE NODE TABLE HAS BEEN %T//
2671 054324 SAVED: .ASCIZ /SAVED./
2672 054333 RESTOR: .ASCIZ /RESTORED./
2673 054345 MSGPRM: .ASCIZ /%N%ATHE CURRENT MESSAGE PARAMETERS ARE://
2674 054415 MSG1: .ASCIZ /%N%ATHE COLLECTION OF ALL NODE ADDRESSES COULD TAKE AS LONG AS 40 MINUTES./
2675 054530 MSG11: .ASCIZ /%N%AAHOWEVER, IF NO NEW NODES ARE ADDED TO THE TABLE FOR A 10 MINUTE PERIOD/
2676 054643 MSG12: .ASCIZ /%N%ATHE COLLECTION WILL STOP.%N/
2677 054703 MSG2: .ASCIZ /%N%AYOU ENTERED %T%A NODE: %T//
2678 054741 MSG3: .ASCIZ /%N%ATHE SPECIFIED ADDRESS IS: %T//
2679 055002 MSG4: .ASCIZ /%N%ATYPE=%T%A, SIZE=%D4%A, COPIES=%D3/
2680 .EVEN
2681 055046 HDMMSG1: .ASCIZ /%N%AA ETHERNET DEFAULT ADDRESS (HEX): %T//

```

```

2682 .EVEN
2683 055120 HELP1: .ASCIZ \#N#ACOMMAND SUMMARY FOR THE NETWORK INTERCONNECT EXERCISER (NIE)\
2684 055221 HELP2: .ASCIZ \#N#A(IT IS ONLY NECESSARY TO TYPE THE LETTERS IN BRACKETS)\
2685 055314 HELP3: .ASCIZ \#N2#A(M)HELP OR ? TYPES THIS HELP TEXT.\
2686 055365 HELP4: .ASCIZ \#N2#A(E)XIT - RETURN TO THE SUPERVISOR.\
2687 055436 HELP5: .ASCIZ \#N2#A(S)H)OW (N)ODES - PRINTS INFORMATION IN CURRENT NODE TABLE.\
2688 055536 HELP6: .ASCIZ \#N2#A(S)H)OW (M)ESSAGE - PRINTS THE SELECTED MESSAGE TYPE, SIZE AND COPIES.\
2689 055651 HELP7: .ASCIZ \#N2#A(S)H)OW (C)OUNTERS - PRINTS THE LOW LEVEL COUNTERS OF THE MOST NODE.\
2690 055762 HELP8: .ASCIZ \#N2#A(R)UN (L)OOPPAIR/PASS=NN - RUNS THE LOOPPAIR TEST.\
2691 056052 HELP9: .ASCIZ \#N2#A(R)UN (A)LL/PASS=NN - RUNS THE NODE-TO-NODE TEST.\
2692 056141 HELP10: .ASCIZ \#N2#A(R)UN (D)IRECT/PASS=NN - RUNS THE LOOP DIRECT TEST.\
2693 056232 HELP11: .ASCIZ \#N2#A(R)UN (P)ATTERN/PASS=NN - RUNS THE MESSAGE PATTERN TEST.\
2694 056330 HLP115: .ASCIZ \#N2#A(R)UN (R)ESPONDER - RUNS LOOP-SERVER/CONSOLE-ID RESPONDER TEST.\
2695 056435 HELP12: .ASCIZ \#N2#A(M)ESSAGE/(T)YPE=A/(S)IZE=N/(C)OPIES=M - ALLOWS THE OPERATOR TO\
2696 056542 HELP13: .ASCIZ \#N#AMODIFY THE DEFAULT MESSAGE TYPE, SIZE AND COPY PARAMETERS.\
2697 056641 HELP14: .ASCIZ \#N2#A(N)ODE ADR/TYPER - ENTERS A PHYSICAL ADDRESS INTO THE NODE\
2698 056740 HELP15: .ASCIZ \#N#ATABLE. THE TYPE CAN BE EITHER (T)ARGET (DEFAULT) OR (A)SSIST.\
2699 057043 HELP16: .ASCIZ \#N2#A(SU)MMARY - PRINTS A SUMMARY OF THE TEST RESULTS.\
2700 057132 HELP17: .ASCIZ \#N2#A(B)UILD - BUILDS A TABLE OF REMOTE NODE PHYSICAL ADDRESSES BY\
2701 057235 HELP18: .ASCIZ \#N#ALISTENING TO ID MESSAGES ON THE NI.\
2702 057305 HELP19: .ASCIZ \#N2#A(C)LEAR (N)ODE/ADR - REMOVES THE NODE SPECIFIED BY EITHER ADR\
2703 057410 HELP20: .ASCIZ \#N#AOR NODE LOGICAL NAME FROM THE NODE TABLE.\
2704 057466 HELP21: .ASCIZ \#N2#A(C)LEAR (N)ODE/(A)LL - CLEARS THE NODE TABLE.\
2705 057551 HELP22: .ASCIZ \#N2#A(C)LEAR (M)ESSAGE - SETS ALL MESSAGE PARAMETERS TO DEFAULT.\
2706 057652 HELP23: .ASCIZ \#N2#A(C)LEAR (S)UMMARY - CLEARS THE TABLE OF SUMMARY TEST DATA.\
2707 057752 HELP24: .ASCIZ \#N2#A(I)DENTIFY ADR - USES THE REQUEST ID FUNCTION TO IDENTIFY A REMOTE NODE ON THE NI.\
2708 060102 HELP25: .ASCIZ \#N2#A(SA)VE - SAVES THE CONTENTS OF THE NODE TABLE.\
2709 060166 HELP26: .ASCIZ \#N2#A(U)NSAVE - REPLACES THE CURRENT NODE TABLE WITH THE SAVED ONE.\
2710 060272 HELP27: .ASCIZ \#N#S#ANOTES: 1) ADR IS THE PHYSICAL ADDRESS OF A NODE ON THE NI.\
2711 060374 HELP28: .ASCIZ \#N#S#A 2) PASS COUNT IS A DECIMAL NUMBER BETWEEN 1 AND 65534. A DEFAULT\
2712 060513 HELP29: .ASCIZ \#N#S#A VALUE OF 1 IS ASSUMED.\
2713 060563 HELP30: .ASCIZ \#N#S#A SPECIFYING -1 CAUSES THE TEST TO BE RUN INDEFINATELY.\
2714 .EVEN
2715 ;
2716 ; TEST MESSAGES AND ARGUMENTS
2717 ;
2718 060672 PASABT: .ASCIZ /#N#A PASS ABORTED!/
2719 060715 TSTMS1: .ASCIZ /#N#T#A TEST -- /
2720 060735 TSTMS2: .ASCIZ /#N#T#A NODE: #T/
2721 060755 TSTMS3: .ASCIZ /#T#A ERROR/
2722 060770 TSTMS4: .ASCIZ /#N#T#A NODE: #T#T#A NODE: #T/
2723 061025 OK: .ASCIZ /#A - RESPONSE OK/
2724 061046 OKRE: .ASCIZ /#N#A - RECEIVE ASSIST - RESPONSE OK/
2725 061112 OKTR: .ASCIZ /#N#A - TRANSMIT ASSIST - RESPONSE OK/
2726 061157 OKFU: .ASCIZ /#N#A - FULL ASSIST RESPONSE OK/
2727 061220 MESPAT: .ASCIZ /#N#AERROR OCCURED WITH #T#A MESSAGE TYPE/
2728 061271 MESPAT1: .ASCIZ /#A DATA PATTERN: #T/
2729 061315 ALLNOD: .ASCIZ /ALL NODE/
2730 061326 LUPAIR: .ASCIZ /LOOPPAIR/
2731 061337 DIRECT: .ASCIZ /LOOP DIRECT/
2732 061353 FULLAST: .ASCIZ /FULL ASSIST/
2733 061367 TRAST: .ASCIZ /TRANSMIT ASSIST/
2734 061407 RECAST: .ASCIZ /RECEIVE ASSIST/
2735 061426 PATTRN: .ASCIZ /MESSAGE PATTERN/
2736 061446 NORESP: .ASCIZ /NO RESPONSE/
2737 061462 RETRY: .ASCIZ /EXCESSIVE COLLISION/
2738 061506 LENGTH: .ASCIZ /LENGTH/

```

```

2739 061515 COMPAR: .ASCIZ /DATA COMPARISON/
2740 .EVEN
2741 ;
2742 ;
2743 061536 MSGTY0: .ASCIZ /ALPHA/ ;MESSAGE TYPES
2744 061544 MSGTY1: .ASCIZ /ONES/
2745 061551 MSGTY2: .ASCIZ /ZEROS/
2746 061557 MSGTY3: .ASCIZ /1ALT/
2747 061564 MSGTY4: .ASCIZ /OALT/
2748 061571 MSGTY5: .ASCIZ /CCITT/
2749 061577 MSGTY6: .ASCIZ /OPER SEL/
2750 061610 CMDTY1: .ASCIZ /EXIT/ ;COMMAND TYPES
2751 061615 CMDTY2: .ASCIZ /SUMMARY/
2752 061625 CMDTY3: .ASCIZ /BUILD/
2753 061633 CMDTY4: .ASCIZ /SHOW/
2754 061640 CMDTY5: .ASCIZ /RUN/
2755 061644 CMDTY6: .ASCIZ /MESSAGE/
2756 061654 CMDTY7: .ASCIZ /NODE/
2757 061661 CMDTY8: .ASCIZ /CLEAR/
2758 061667 CMDTY9: .ASCIZ /REQUEST ID/
2759 061702 ARGTY1: .ASCIZ /NODES/ ;ARGUMENT TYPES
2760 061710 ARGTY2: .ASCIZ /MESSAGES/
2761 061721 ARGTY3: .ASCIZ /COUNTERS/
2762 061732 ARGTY4: .ASCIZ /LOOPPAIR/
2763 061743 ARGTY5: .ASCIZ /ALL/
2764 061747 ARGTY6: .ASCIZ /ASSIST/
2765 061757 ARGTY7: .ASCIZ /TARGET/
2766 .EVEN
2767 061766 NOCLK: .ASCIZ /#N#ABAD CLOCK - PROGRAM WILL HANG ON "TIMEOUT"!!/
2768 ;
2769 ;
2770 ; QNA COUNTER INFORMATION MESSAGES
2771 ;
2772 ;
2773 062047 CNTR00: .ASCIZ /#N#S#A#CONTENTS OF NODE #T#A INTERNAL COUNTERS:/
2774 062127 CNTR01: .ASCIZ /#N#Z#BASECONDS SINCE LAST ZEROED:#S15#Z5/
2775 062176 CNTR02: .ASCIZ /#N#A#PACKETS RECEIVED:#S19#T/
2776 062232 CNTR03: .ASCIZ /#N#M#MULTICAST PACKETS RECEIVED:#S9#T/
2777 062277 CNTR04: .ASCIZ /#N#A#PACKETS REC'D WITH ERROR - BITMAP:#S8#B4/
2778 062354 CNTR05: .ASCIZ /#N#A#PACKETS RECEIVED WITH ERROR:#S13#Z5/
2779 062424 CNTR06: .ASCIZ /#N#A#DATA BYTES RECEIVED:#S16#T/
2780 062463 CNTR07: .ASCIZ /#N#M#MULTICAST DATA BYTES RECEIVED:#S6#T/
2781 062533 CNTR08: .ASCIZ /#N#A#RECEIVED PACKETS LOST-INTERNAL:#S10#Z5/
2782 062606 CNTR09: .ASCIZ /#N#A#RECEIVED PACKETS LOST-LOCAL:#S12#Z5/
2783 062657 CNTR10: .ASCIZ /#N#A#PACKETS TRANSMITTED:#S16#T/
2784 062716 CNTR11: .ASCIZ /#N#M#MULTICAST PACKETS TRANSMITTED:#S6#T/
2785 062766 CNTR12: .ASCIZ /#N#A#PACKETS TRANSMITTED 3. TRYS:#S8#T/
2786 063034 CNTR13: .ASCIZ /#N#A#PACKETS TRANSMITTED 2 TRYS:#S9#T/
2787 063101 CNTR14: .ASCIZ /#N#A#PACKETS DEFERRED:#S19#T/
2788 063135 CNTR15: .ASCIZ /#N#A#DATA BYTES TRANSMITTED:#S13#T/
2789 063177 CNTR16: .ASCIZ /#N#M#MULTICAST BYTES TRANSMITTED:#S8#T/
2790 063245 CNTR17: .ASCIZ /#N#A#TRANSMIT PACKETS ABORTED-BITMAP:#S8#B6/
2791 063320 CNTR18: .ASCIZ /#N#A#TRANSMIT PACKETS ABORTED:#S16#Z5/
2792 063365 CNTR19: .ASCIZ /#N#A#XMIT COLLISION CHECK FAILURE:#S12#Z5/
2793 ;
2794 ;
2795 ; ERROR MESSAGES FOR DEQNA DRIVER

```

```

2796
2797
2798 063436  EMSG01: .ASCIZ /DEQNA PORT COMMAND ERROR/          ; SHOULD NEVER GET THIS ERROR
2799 063467  EMSG02: .ASCIZ /DEQNA NON EXISTENT MEMORY ERROR/
2800 063527  EMSG03: .ASCIZ /UNEXPLAINED DEQNA INTERRUPT/
2801 063563  EMSG04: .ASCIZ /UNKNOWN QNA ERROR/
2802 063605  EMSG08: .ASCIZ /TIMEOUT!--DEQNA HARDWARE FAILED TO TRANSMIT PACKET/
2803 063670  EMSG10: .ASCIZ /DEQNA STILL OWNS TRANSMIT DESCRIPTOR/
2804 063735  EMSG11: .ASCIZ /RECEIVE RING BOOKKEEPING ERROR/
2805 063774  EMSG14: .ASCIZ /MESSAGE SIZE TOO BIG FOR MAX. PACKET LENGTH/
2806 064050  EMSG22: .ASCIZ /NO ANSWER RESPONSE FROM NODE./
2807 064103  EMSG24: .ASCIZ /TRANSMIT ERROR. ALL PACKETS NOT TRANSMITTED/
2808 064157  EMSG25: .ASCIZ /ERROR WHILE WRITING MULTICAST ADDRESS LIST INTO DEQNA/
2809 064245  EMSG32: .ASCIZ \MAILLEGAL TARGET/ASSIST PAIR IN NODE TABLE\
2810 064322  EMSG34: .ASCIZ <15><12>/TIMEOUT WAITING FOR LOOPBACK REPLY/
2811 064367  EMSG35: .ASCIZ /AFAILING NODE ADDRESS: #T#N/
2812 064424  EMSG36: .ASCIZ /ADATA PATTERN: #T#N/
2813 064451  EMSG37: .ASCIZ /AFAILING TARGET NODE ADDRESS: #T#N/
2814 064515  EMSG38: .ASCIZ /AFAILING ASSIST NODE ADDRESS: #T#N/
2815 064561  EMSG40: .ASCIZ <15><12>/TIMEOUT WAITING FOR REPLY - LOOP MESSAGE TYPE - RECEIVE ASSIST/
2816 064662  EMSG41: .ASCIZ <15><12>/TIMEOUT WAITING FOR REPLY - LOOP MESSAGE TYPE - TRANSMIT ASSIST/
2817 064764  EMSG42: .ASCIZ <15><12>/TIMEOUT WAITING FOR REPLY - LOOP MESSAGE TYPE - FULL ASSIST/
2818 065062  EMSG43: .ASCIZ \NON-EXISTENT MEMORY ERROR DURING ADDRESS SETUP\
2819 065141  EMSG44: .ASCIZ \SETUP ADDRESS MISMATCH ERROR - DEQNA INTERNAL LOOPBACK FAILED\
2820 065237  EMSG45: .ASCIZ \ADDRESS SETUP-PACKET TRANSMIT ERROR\
2821 065303  EMSG46: .ASCIZ \ADDRESS SETUP-PACKET TRANSMITTED, BUT NOT RECEIVED\
2822 065366  EMSG47: .ASCIZ \PACKET RECEIVED WAS NOT AN ADDRESS SETUP-PACKET\
2823 065446  EMSG48: .ASCIZ \ADDRESS SETUP-PACKET TRANSMIT AND RECEIVE BYTE COUNT MISMATCH\
2824 065544  EMSG50: .ASCIZ \CASE OFFSET RANGE UNDERFLOW\
2825 065600  EMSG51: .ASCIZ \CASE OFFSET RANGE OVERFLOW\
2826 065633  EMSG52: .ASCIZ \LOOP-SERVER FUNCTION CODE IS NEITHER FORWARD OR REPLY\
2827 065721  EMSG53: .ASCIZ \RECEIVE DATA TOO LARGE FOR RECEIVE BUFFER\
2828 065773  EMSG54: .ASCIZ \MESSAGE ADDRESS WAS GOOD, BUT RUNT PACKET WAS RECEIVED.\
2829 066063  EMSG55: .ASCIZ \RECEIVER OVERFLOW OCCURRED\
2830 066116  EMSG56: .ASCIZ \RECEIVER CRC ERROR OCCURRED\
2831 066152  EMSG57: .ASCIZ \RECEIVER FRAMING ERROR OCCURRED\
2832 066211  EMSG58: .ASCIZ \RECEIVED PACKET WITH ERRORS, THROWING IT AWAY\
2833 066267  EMSG59: .ASCIZ \UNEXPECTED LOOP-SERVER REPLY PACKET RECEIVED\
2834 066344  EMSG60: .ASCIZ \RECEIVED PACKET OF LESS THAN 10 BYTES\
2835 066412  EMSG61: .ASCIZ \UNKNOWN RECEIVE ERROR OCCURRED\
2836 066451  EMSG62: .ASCIZ \ERROR OCCURRED WHILE TRANSMITTING CONSOLE ID\
2837 066526  EMSG63: .ASCIZ \NEXT RCV DESCRIPTOR POINTER PASSED CURRENT RCV POINTER\
2838 066615  EMSG64: .ASCIZ \OPERATOR ERROR- CANNOT USE THE DEVICE'S OWN NI ADDRESS\
2839 066705  EMSG65: .ASCIZ \TIMEOUT WAITING FOR BOOT/DIAGNOSTIC CODE TO LOAD FROM QNA\
2840 066777  EMSG66: .ASCIZ \LOADED BOOT/DIAGNOSTIC ROM CODE AND ROM CHEKCSUM MISMATCH\
2841 067071  EMSG67: .ASCIZ \CITIZENSHIP TEST FAILED\
2842 067121  EMSG68: .ASCIZ \QNA HARDWARE RECEIVED A PACKET NOT ADDRESSED TO IT\
2843 067204  EMSG69: .ASCIZ \AN ASSIST NODE HAS BEEN CHANGED TO A TARGET NODE\
2844
2845 ;
2846 067266  SIMSG1: .ASCIZ /#N#A NODE DEFAULT ADDRESS: #T/
2847 067325  SIMSG2: .ASCIZ /#N#S8#ARECEIPT NUMBER: #06/
2848 067360  SIMSG3: .ASCIZ /#N#A MAINTENANCE VERSION: #Z2/
2849 067420  SIMSG4: .ASCIZ /#N#S19#AECO: #Z2/
2850 067441  SIMSG5: .ASCIZ /#N#S14#AUSER ECO: #Z2/
2851 067467  SIMSG6: .ASCIZ /#N#S14#AFUNCTION: #02/
2852 067515  SIMSG7: .ASCIZ /#N#S16#ADEVICE: #02/
    
```

```

2853      ;
2854      .EVEN
2855 067542 PCMSG: .ASCIZ /#N#APC OF CALLING ROUTINE = #06/
2856      .EVEN
2857 067602 CMPE1: .ASCIZ /#N#ABYTE NUMBER:#D4#A(D) EXPECTED=#06#A(O) RECEIVED=#06#A(O)/
2858 067677 CMPE2: .ASCIZ /#N#ATOTAL MISMATCHES IN MESSAGE = #D4/
2859 067745 LGERMS: .ASCIZ /#N#ALENGTH ERROR -- BYTES EXPECTED: #06#A BYTES RECEIVED: #06/
2860 070043 SUMMS1: .ASCIZ /#N2#S8#ANODE: #T/
2861 070064 SUMMS2: .ASCIZ /#N#ARX NOT COMPLETE      RX COMPLETE      LENGTH ERRORS/
2862 070150 SUMMS3: .ASCIZ /#N#S6#Z5#S12#Z5#S10#Z5/
2863 070177 SUMMS4: .ASCIZ /#N#ACOMPARE ERRORS      BYTES COMPARED      BYTES XFER/
2864 070262 SUMMS5: .ASCIZ /#N#S6#Z5#S8#T/
2865 070300 SUMMS6: .ASCIZ /#S5#T/
2866      .EVEN
2867
2868      .LIST  BIN

```



```

2870          .SBTTL GLOBAL ERROR REPORT SECTION
2871
2872          ;**
2873          ; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
2874          ; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION. PRINTB
2875          ; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
2876          ;--
2877
2878
2879 070306          BGNMSG ERR1
2880 070306 012737 000020 003734          MOV      #CEXIT,CFLAG          ERR1::
2881 070314          PRINTX #PCMSG,PCCALL
2882 070314 013746 050330          MOV      PCCALL,-(SP)
2883 070320 012746 067542          MOV      #PCMSG,(SP)
2884 070324 012746 000002          MOV      #2,-(SP)
2885 070330 010600          MOV      SP,R0
2886 070332 104415          TRAP   C$PNTX
2887 070334 062706 000006          ADD     #6,SP
2888 070340          DOCLN          TRAP   C$DCLN
2889 070340 104444
2890 070342          ENDMSG
2891 070342 104423          L10002: TRAP   C$MSG
2892 070344          BGNMSG ERR2          ERR2::
2893 070344 010146          MOV      R1,-(SP)
2894 070346 013701 002372          MOV      P$TYPE,R1
2895 070352 006301          ASL     R1
2896 070354 062701 003324          ADD     #MSGTAB,R1
2897 070360          PRINTX #EMSG35,#STRBUF
2898 070360 012746 002324          MOV      #STRBUF,(SP)
2899 070364 012746 064367          MOV      #EMSG35,-(SP)
2900 070370 012746 000002          MOV      #2,(SP)
2901 070374 010600          MOV      SP,R0
2902 070376 104415          TRAP   C$PNTX
2903 070400 062706 000006          ADD     #6,SP
2904 070404          PRINTX #EMSG36,(R1)
2905 070404 011146          MOV      (R1),(SP)
2906 070406 012746 064424          MOV      #EMSG36,(SP)
2907 070412 012746 000002          MOV      #2,-(SP)
2908 070416 010600          MOV      SP,R0
2909 070420 104415          TRAP   C$PNTX
2910 070422 062706 000006          ADD     #6,SP
2911 070426 012601          MOV      (SP)+,R1
2912 070430          ENDMSG          L10003: TRAP   C$MSG
2913 070430 104423
2914 070432          BGNMSG ERR3          ERR3::
2915 070432          PRINTX #EMSG37,#STRBUF
2916 070432 012746 002324          MOV      #STRBUF,(SP)
2917 070436 012746 064451          MOV      #EMSG37,(SP)
2918 070442 012746 000002          MOV      #2,(SP)
2919 070446 010600          MOV      SP,R0
2920 070450 104415          TRAP   C$PNTX

```

070452 062706 000006  
 2896 070456 012746 002346  
 070462 012746 064515  
 070466 012746 000002  
 070472 010600  
 070474 104415  
 070476 062706 000006  
 2897 070502  
 070502  
 070502 104423

PRINTX @MSG38,@STRBU1  
  
 ENDMSG

ADD @6,SP  
 MOV @STRBU1,(SP)  
 MOV @MSG38,(SP)  
 MOV @2,-(SP)  
 MOV SP,RO  
 TRAP C@PNTX  
 ADD @6,SP

L10004:  
 TRAP C@MSG

2899  
 2900  
 2901  
 2902  
 2903  
 2904  
 2905  
 2906  
 2907  
 2908  
 2909  
 2910  
 2911

```

;*****
;   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.
;*****

```

2914  
2915  
2916  
2917  
2918  
2919  
2920  
2921  
2922  
2924  
2925  
2926  
2927  
2929  
2930  
2931  
2933  
2934  
2935  
2937  
2938  
2939  
2941  
2942  
2943  
2944  
2946  
2947  
2948  
2950  
2951  
2952  
2954  
2955  
2956  
2958  
2959  
2960  
2961  
2963  
2964  
2965  
2967  
2968  
2969  
2971  
2972  
2973  
2975  
2976  
2977  
2978  
2979  
2981  
2982  
2983  
2985

```
.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:  
; *****
```

2986  
2987  
2988  
2989  
2990  
2992  
2993  
2995  
2996  
2997  
2998  
3000  
3002  
3003  
3004  
3006

```

;          GIVE THE EXACT CALLING SEQUENCE USED TO ACCESS THIS SUBROUTINE.
;          FOR EXAMPLE:  MOV COUNT,R1      ;MOVE INPUT TO R1
;                          JSR   PC,ROUTINE ;GO TO ROUTINE
;                          BCS   ERROR     ;CARRY SET IF ROUTINE HAD ERROR
;*****
; -
;*****
;          INSERT THE CODE FOR THIS SUBROUTINE.  THE NAME OF THE SUBROUTINE SHOULD
;          BE DEFINED WITH A DOUBLE-COLON (::); THIS WILL MAKE THE SUBROUTINE GLOBAL.
;*****
;*****
;          BEGIN EACH SUBROUTINE AT THE TOP OF A NEW PAGE.
;*****
;
;
```

.SBTTL CLKSET CLOCK SETUP SUBROUTINE

3008  
3009  
3010  
3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024  
3025  
3026  
3027  
3028

;  
; \*  
; FUNCTIONAL DESCRIPTION:  
; THIS SUBROUTINE SETS UP THE CLOCK INFORMATION TABLE FOLLOWING  
; A "CLOCK" CALL EXECUTED IN THE INITIALIZATION CODE. BUT SINCE  
; THE "CLOCK" CALL SAYS NOTHING ABOUT AN LSI 11'S CLOCK, THE  
; ROUTINE IS ONLY USED IF A LINE OR P CLOCK IS FOUND.  
;  
; INPUTS R1 - POINTS TO SUPERVISOR SPACE WHERE CLOCK INFO WAS RETURNED  
; R2 - POINTS TO "CLK" TABLE WHERE CLOCK INFO WILL BE KEPT  
;  
; OUTPUTS - "CLKCSR" GETS LOADED WITH THE CLOCK'D CSR ADDRESS  
; "CLKBR" GETS LOADED WITH THE CLOCK'S INTERRUPT LEVEL  
; "CLKVEC" GETS LOADED WITH THE CLOCK'S INTERRUPT VECTOR  
; "CLKHZ" GETS LOADED WITH THE LINE FREQ. (IN HERTZ)  
;  
; CALLING PROCEDURE:  
; JSR PC,CLKSET ; CALL CLOCK SETUP WITH R1 AND R2 SETUP  
;  
; ---  
; CLKSET:

3029 070504  
3030 070504 012122  
3031 070506 012112  
3032 070510 006312  
3033 070512 006312  
3034 070514 006312  
3035 070516 006312  
3036 070520 006322  
3037 070522 012122  
3038 070524 012122  
3039 070526 000207

MOV (R1), (R2) ; LOAD CLOCK'S CSR ADDR. INTO "CLKCSR"  
MOV (R1), (R2) ; LOAD CLOCK'S INTR. LEVEL INTO "CLKBR"  
ASL (R2) ; ADJUST THE INTR. LEVEL FOR LOADING  
ASL (R2) ; INTO THE PSW WITH A "SETVEC" CALL  
ASL (R2)  
ASL (R2)  
ASL (R2)  
MOV (R1), (R2) ; LOAD CLOCK'S INTR. VEC INTO "CLKVEC"  
MOV (R1), (R2) ; LOAD CLOCK'S FREQ. INTO "CLKHZ"  
PC

.SBTTL CLKINT CLOCK INTERRUPT SERVICE ROUTINE

3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071

;-+  
: 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  
:  
:--+

3072 070530  
070530

BGNSRV CLKINT

CLKINT::

3073  
3074 070530  
3075 070534  
3076 070540  
3077 070542  
3078 070550  
3079 070554  
3080 070560  
3081 070566  
3082 070570  
3083 070574  
3084 070600  
3085 070600  
3086 070604  
3087 070606  
3088 070612  
3089 070612  
3090 070616  
3091 070620  
3092 070624  
3093 070624  
3094 070630  
3095 070632  
3096 070640

005077 113202  
005337 003754  
001017  
013737 003744 003754  
005237 050146  
005237 003752  
022737 000074 003752  
001004  
005237 003750  
005037 003752  
005737 003756  
001402  
005337 003756  
005737 003760  
001402  
005337 003760  
005737 003762  
001406  
023737 003744 003754  
001002

:  
: CLR BCLKCSR  
: DEC TIMTCK  
: BNE 10\$  
: MOV CLKHZ,TIMTCK  
: INC CNTRS+C.SECONDS  
: INC TIMSEC  
: CMP #60.,TIMSEC  
: BNE 10\$  
: INC TIMMIN  
: CLR TIMSEC  
:  
10\$: TST TIMER1  
: BEQ 20\$  
: DEC TIMER1  
:  
20\$: TST TIMER2  
: BEQ 30\$  
: DEC TIMER2  
:  
30\$: TST TIMERS  
: BEQ 40\$  
: CMP CLKHZ,TIMTCK  
: BNE 40\$

: DISABLE THE CLOCK FROM INTERRUPTING  
: DECREMENT THE NO. OF TICKS/SEC  
: GO CHECK TIMERS  
: RESET THE NO. OF TICKS/SEC.  
: INCREMENT QNA'S NUMBER OF SECS  
: INC. NO OF SECS-SINCE-START  
: SEE IF WE'VE COUNTED 60 SEC.S YET  
: IF NOT, GO CHECK TIMERS  
: ELSE, INC. MINUTES-SINCE-START  
: AND RESTART SECOND COUNTER  
:  
: SEE IF TIMER1 TIMING ANYTHING  
: IF=0, NO, CHECK NEXT TIMER  
: ELSE DECREMENT THE TIMER VALUE (BY 1 TICK)  
:  
: SEE IF TIMER2 TIMING ANYTHING  
: IF=0, NO, CHECK NEXT TIMER  
: ELSE DECREMENT TIMER VALUE (BY 1 TICK)  
:  
: SEE IF TIMERS TIMING ANYTHING  
: IF=0, NOTHING BE TIMED, LEAVE  
: SEE IF A SECOND HAS BEEN COUNTED OFF  
: BR IF NO

3097 070642 005337 003762  
3098 070646 40\$:  
3099 070646 013777 003746 113062  
3100 070654  
070654  
070654 000002

DEC TIMERS  
MOV CLKEN, @CLKCSR  
ENDSRV

; ELSE, DECREMENT TIMER VALUE (BY 1 SEC.)  
; REENABLE THE CLOCK TO INTERRUPT  
L10005:  
RTI

```

3102 .SBTTL PREG14 PRESERVE REGISTERS 1 THROUGH 4 ACROSS SUBROUTINE CALLS
3103 ;*
3104 ;INPUTS: THE RELATIVE ADDRESS OF THE CALLED ROUTINE MUST FOLLOW THE
3105 ; CALL TO THIS ROUTINE (SEE CALLING SEQUENCE).
3106 ;
3107 ;OUTPUTS: REGISTERS 1 THROUGH 4 ARE PRESERVED ACROSS THE CALLED ROUTINE.
3108 ; A CHECK IS MADE TO ENSURE THE HARDWARE STACK HASN'T OVER RUN
3109 ; THE PARAMETER STACK.
3110 ;
3111 ;CALLING SEQUENCE: THIS IS BEST HANDLED BY THE "CALL' MACRO. THE ACTUAL
3112 ; CALL IS:
3113 ; JSR R4,PREG14
3114 ; .WORD [SUBROUTINE NAME] ANCHOR
3115 ;
3116 ;COMMENTS: THIS ROUTINE IS RE-ENTRANT AND RELOCATABLE.
3117 ; THIS ROUTINE IS DRS COMPATIBLE.
3118 ;
3119 ;SUBORDINATE ROUTINES CALLED: THE ROUTINE SPECIFIED IN THE CALL.
3120 ;-
3121 ;R4 IS ALREADY ON THE R6 STACK.
3122 070656 PREG14: MOV R3,-(SP) ;PUSH R3, R2, R1
3123 070656 010346 MOV R2,-(SP) ;
3124 070660 010246 MOV R1,-(SP) ;
3125 070662 010146 MOV R4,PCCALL ;
3126 070664 010437 050330 MOV (R4)+,R1 ;GET THE RELATIVE ADDRESS OF THE CALLED
3127 070670 012401 ;ROUTINE.
3128 ;MAKE IT AN ABSOLUTE ADDRESS.
3129 070672 060701 ANCHOR: ADD PC,R1
3130 070674 ;SAVE THE RETURN TO THE CALLING ROUTINE.
3131 070674 010446 MOV R4,-(SP) ;CHECK FOR STACK OVER-RUN.
3132 070676 020506 CMP R5,SP
3133 070700 103401 BLO 10$ ;
3134 070702 000000 HALT ;HANDLE STACK OVER RUN CONDITION.
3135 070704 10$: JSR PC,(R1) ;CALL THE SPECIFIED ROUTINE.
3136 070704 004711 MOV (SP)+,R4 ;RESTORE THE RETURN TO THE CALLING ROUTINE.
3137 070706 012604 MOV (SP)+,R1 ;RESTORE THE REGISTERS.
3138 070710 012601 MOV (SP)+,R1
3139 070712 012602 MOV (SP)+,R2
3140 070714 012603 MOV (SP)+,R3
3141 070716 000204 RTS R4 ;BACK TO THE CALLING ROUTINE.

```



```

3143 .SBTTL WAIT WAIT FOR DEQNA INTERRUPT WITH TIMEOUT
3144
3145 ;**
3146 ; FUNCTIONAL DESCRIPTION:
3147 ; THIS SUBROUTINE WAITS FOR INTERRUPTS FROM THE DEQNA
3148 ; OR REPORTS A TIMEOUT. IF A QNA INTERRUPT HAS NOT
3149 ; OCCURED WITHIN THE TIMEOUT PERIOD THE SUBROUTINE ERROR
3150 ; IS CALLED TO HANDLE IT.
3151 ;
3152 ; SUCCESS OR FAILURE IS REPORTED VIA P1.
3153 ;
3154 ; TNPPTS
3155 ; NONE
3156 ;
3157 ; OUTPUTS-
3158 ; P1: SUCCESS/FAILURE 0=SUCCESS/ 1=FAILURE
3159 ;
3160 ; CALLING SEQUENCE:
3161 ; CALL WAIT
3162 ; P$POP P1
3163 ;--
3164 070720 WAIT::
3165 070720 012703 000012 MOV #10,R3 ; MOVE NO. OF COUNTS TO R3
3166 070724 012704 003756 MOV #TIMER1,R4 ; AND TIMER TO BE USED TO R4
3167 070730 005002 CLR R2 ; LOCAL STATUS PARAMETER
3168 070732 010314 MOV R3,(R4) ; SET NUMBER OF TICKS. (GLOBAL)
3169 070734 10$:
3170 070734 005737 050270 TST ERRFLG ; CHECK IF ERROR OCCURED
3171 070740 001011 BNE 30$ ; BR IF YES
3172 070742 005737 050262 TST DNIFLG ; CHECK FOR INTERRUPT
3173 070746 001403 BEQ 20$ ; BR IF INTERRUPT RECEIVED
3174 070750 005037 050262 CLR DNIFLG
3175 070754 000410 BR 50$
3176 070756 20$:
3177 070756 005714 TST (R4) ; HAS TIMER EXPIRED?
3178 070760 001365 BNE 10$ ; BR IF NO TO WAIT FOR INTERRUPT
3179 070762 000403 BR 40$ ; BR TO 40$
3180 070764 30$:
3181 070764 CALL ERROR ; CALL ERROR ROUTINE
3182 070764 004437 070656 JSR R4,PREG14
3183 070770 000106 .WORD ERROR ANCHOR
3184 070772 40$:
3185 070772 012702 177777 MOV # 1,R2 ; INDICATE FAILURE
3186 070776 50$:
3187 070776 RETURN R2 ; RETURN WITH SUCCESS/FAILURE INDICATION
3188 070776 010225 MOV R2,(R5)+
3189 071000 000207 RTS PC
    
```

3187  
 3188  
 3189  
 3190  
 3191  
 3192  
 3193  
 3194  
 3195  
 3196  
 3197  
 3198  
 3199  
 3200  
 3201  
 3202  
 3203  
 3204 071002  
 3205 071002 005337 050270  
 3206  
 3207 071006 005737 050254  
 3208 071012 001013  
 3209 071014 005737 050252  
 3210 071020 001017  
 3211 071022 005737 050266  
 3212 071026 001023  
 3213 071030  
 071030 104455  
 071032 000004  
 071034 063563  
 071036 070306  
 3214 071040 000424  
 3215 071042  
 071042 104455  
 071044 000001  
 071046 063436  
 071050 070306  
 3216 071052 005337 050254  
 3217 071056 000415  
 3218 071060  
 071060 104455  
 071062 000002  
 071064 063467  
 071066 070306  
 3219 071070 005337 050252  
 3220 071074 000406  
 3221 071076  
 071076 104455  
 071100 000003  
 071102 063527  
 071104 070306  
 3222 071106 005337 050266  
 3223 071112  
 071112 000207  
 3224

.SBTTL ERROR HANDLE QNA INTERRUPT ERRORS

```

;
; *
; FUNCTIONAL DESCRIPTION:
; THIS SUBROUTINE CHECKS THE ERROR FLAGS SET BY
; QNAISR THE INTERRUPT SERVICE ROUTINE AND PRINTS
; OUT THE APPROPRIATE ERROR MESSAGES.
;
; INPUTS
; IMPLICIT: ERROR FLAGS SHOULD BE SET BY UNAIISR ROUTINES.
; OUTPUTS
; IMPLICIT: ERROR MESSAGES ARE PRINTED OUT TO THE OPERATOR CONSOLE.
;
; CALLING SEQUENCE:
; CALL ERROR
;
; --
;
; ERROR::
;
; DEC ERRFLG ; DECREMENT ERROR COUNTER TO SHOW
; THAT IT HAS BEEN HANDLED
; TST PCEFLG ; SEE IF PORT COMMAND ERROR
; BNE 10$ ; IF YES, BRANCH
; TST FATFLG ; SEE IF QNA FATAL ERROR
; BNE 20$ ; IF YES, BRANCH
; TST BCOUNT ; SEE IF UNEXPLAINED INTERRUPT
; BNE 30$ ; IF YES, BRANCH
; ERRDF 4,EMSG04,ERR1 ; ELSE UNKNOWN ERROR
;
; TRAP C$ERDF
; .WORD 4
; .WORD EMSG04
; .WORD ERR1
;
; BR 40$ ; EXIT
; 10$: ERRDF 1,EMSG01,ERR1 ; PORT COMMAND ERROR
;
; TRAP C$ERDF
; .WORD 1
; .WORD EMSG01
; .WORD ERR1
;
; DEC PCEFLG ; INDICATE THAT IT WAS HANDLED
; BR 40$ ; EXIT
; 20$: ERRDF 2,EMSG02,ERR1 ; QNA FATAL ERROR
;
; TRAP C$ERDF
; .WORD 2
; .WORD EMSG02
; .WORD ERR1
;
; DEC FATFLG ; KEEP UP ON BOOK KEEPING
; BR 40$ ; EXIT
; 30$: ERRDF 3,EMSG03,ERR1 ; UNEXPLAINED INTERRUPT
;
; TRAP C$ERDF
; .WORD 3
; .WORD EMSG03
; .WORD ERR1
;
; DEC BCOUNT ; BOOK KEEPING
; 40$: RETURN ; RETURN
;
; RTS PC

```

3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239  
3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259 071114  
3260 071114 013702 047772  
3261 071120 052737 000002 050246  
3262 071126 013762 050246 000016  
3263 071134 012737 000001 003756  
3264 071142  
3265 071142 005737 003756  
3266 071146 001375  
3267  
3268 071150 012762 000002 000016  
3269 071156 005062 000016  
3270  
3271  
3272  
3273 071162 005737 050250  
3274 071166 001162  
3275 071170 012737 000001 050250  
3276 071176 012701 000002  
3277  
3278 071202 012704 021200  
3279 071206 012703 004100  
3280 071212  
3281 071212 012724 177777  
3282 071216 012724 100000

```

.SBTTL      QNAINI  INITIALIZE THE QNA
;
; SUBROUTINE TO
;           1) SETS QNA IN THE READY STATE
;           2) INITIALIZES ALL QNA GLOBAL DATA LOCATIONS
;              TO DEFAULT VALUES.
; ROUTINE STEPS:
;
;   SET RESET BIT RESET THE QNA
;   SET TIMER TO ALLOW 100 USECONDS FOR INIT
;   INVALIDATE THE TRANSMIT LIST(S)
;   VALIDATE THE RECEIVE LIST(S)
;   ZERO RECEIVED BUFFER COUNT
;   WAIT FOR TIMER
;   CLEAR THE RESET BIT
;   SET VECTOR ADDRESS
;   READ PHYSICAL ADDRESS ROM AND STORE IT
;   WRITE THE RECEIVE RING LOWER AND UPPER ADDRESS(S)
;   ENABLE INTERRUPTS AND DATA RECEPTION
;   EXIT
;
; CALLED BY:      CALL QNAINI
;
; INPUTS:         NONE
;
; OUTPUTS:        NONE
;
; SIDE EFFECTS:   ALL GLOBAL LOCATIONS ARE ZEROED
;
; RESET AND STOP QNA HARDWARE
;
QNAINI::
MOV     QNAADO,R2           ; GET THE DEVICE ADDRESS
BIS     @MRESET,CSRBUF     ; SET SOFTWARE INIT BIT
MOV     CSRBUF,CSR(R2)     ; WRITE IT TO THE QNA
MOV     @1,TIMER1         ; TIMER FOR INITIALIZE
10$:
TST     TIMER1             ; INIT DONE?
BNE     10$               ; BRANCH IF NOT
;
MOV     @MRESET,CSR(R2)    ; CLEAR THE RESET BIT
CLR     CSR(R2)           ; WRITE IT TO THE QNA
;
;LOAD AND RUN CITIZENSHIP TEST
;
TST     FLAG1              ; HAVE WE GONE THRU THE CITIZENSHIP
BNE     QNAINI            ; TEST BEFORE? IF YES, BRANCH.
MOV     @1,FLAG1
MOV     @2,R1
;BUILD TWO RECEIVE DESCRIPTORS OF
; 2K BYTES EACH
MOV     @RDESC,R4         ;GET POINTER TO DESCRIPTOR
MOV     @RBUFB,R3        ;GET POINTER TO 1ST BUFFER
40$:
MOV     @ 1,(R4).         ;FLAGWORD
MOV     @V,(R4).         ;DESCRIPTOR BITS

```

```

3283 071222 010324      MOV      R3,(R4).          ;POINTER TO BUFFER
3284
3285 071224 062703 004000  ADD      #2048.,R3        ;GET POINTER TO 2ND BUFFER
3286 071230 012724 176000  MOV      #-1024.,(R4).    ;SIZE IN WORDS
3287 071234 012724 177777  MOV      # 1,(R4).       ;STATUS WORD 1
3288 071240 012724 177777  MOV      #-1,(R4).       ;STATUS WORD 2
3289
3290 071244 005301      DEC      R1
3291 071246 003361      BGT     40$
3292 071250 005024      CLR     (R4).            ;FLAGWORD OF INVALID DESCRIPTOR
3293 071252 005024      CLR     (R4).            ;DESCRIPTOR BITS OF INVALID DESCRIPTOR
3294
3295 071254 012762 000002 000016  MOV     #MRESET,CSR(R2)   ; CLEAR THE RESET BIT
3296 071262 005062 000016      CLR     CSR(R2)          ; WRITE IT TO THE QNA
3297
3298 071266 012762 021200 000004  MOV     #RDESC,LORCV(R2)  ;SET POINTER TO DESCRIPTOR
3299 071274 012762 001010 000016  MOV     #BD!EL,CSR(R2)    ;SET LOAD ROM BIT
3300 071302 005062 000006      CLR     HIRCV(R2)        ;BEGIN THE LOAD
3301 071306 005004      CLR     R4              ;WAIT FOR BOOT PHASE 1
3302 071310
3303 071310 077401      SOB     R4,45$
3304 071312 042762 000010 000016  BIC     #BD,CSR(R2)       ;CLEAR BOOT BIT
3305 071320 005004      CLR     R4              ;WAIT FOR BOOT PHASE 2
3306 071322
3307 071322 077401      SOB     R4,47$
3308
3309 071324 012762 000002 000016  MOV     #MRESET,CSR(R2)   ; CLEAR THE RESET BIT
3310 071332 005062 000016      CLR     CSR(R2)          ; WRITE IT TO THE QNA
3311
3312 071336 012704 021200      MOV     #RDESC,R4
3313 071342 026427 000000 177777  CMP     FLAGWORD(R4),# 1  ;DID CITIZENSHIP TEST LOAD O.K.?
3314 071350 001016      BNE     50$              ;NO. ERROR
3315
3316 071352 026427 000010 177777  CMP     STAT3(R4),# 1    ;NO. ERROR
3317 071360 001012      BNE
3318
3319 071362 026427 000012 177777  CMP     STAT4(R4),#-1   ;NO. ERROR
3320 071370 001006      BNE
3321
3322 071372 012704 021214      MOV     #RDESC+RDESSZ,R4 ;GET POINTER TO 2ND DESCRIPTOR
3323 071376 026427 000000 177777  CMP     FLAGWORD(R4),# 1 ;DID CITIZENSHIP TEST LOAD O.K.?
3324 071404 001410      BEQ     70$              ;YES
3325
3326 071406
3327 071406      ERRHRD 65,MSG65        ;NO. ERROR.
3328 071406 104456
3329 071410 000101
3330 071412 066705
3331 071414 000000
3332 071416 012700 177777      MOV     # 1,R0
3333 071422 000137 072122      JMP     QNAEXI
3334
3331 071426
3332 071426 013701 047772      MOV     QNAADO,R1        ;GET BASE ADDRESS
3333
3334 071432 010246
3335 071434 010546      MOV     R2,-(SP)
3335 071434 010546      MOV     R5,(SP)         ;SAVE R5 STACKPOINTER

```

```

TRAP
.WORD 65
.WORD MSG65
.WORD 0

```

```

3336
3337 071436 012702 014100      MOV    #CITWORK,R2      ;GET WORK AREA
3338 071442 004737 004104      JSR    PC,RBUF#CITVECTOR ;GO TO THE CITIZENSHIP TEST VIA VECTOR
3339
3340 071446 012605              MOV    (SP),R5          ;RESTORE R5 STACKPOINTER
3341 071450 012602              MOV    (SP),R2
3342
3343 071452 005700              TST    R0               ;DID ERRORS OCCUR DURING CITIZENSHIP?
3344 071454 001413              BEQ    80$              ;NO
3345 071456 020027 100000      CMP    R0,#100000
3346 071462 001410              BEQ    80$              ;NO
3347 071464              ERRHRD 67,MSG67        ;NO, ERROR.
                                TRAP    C#ERRHRD
                                .WORD   67
                                .WORD   MSG67
                                .WORD   0
3348 071474 012700 177777      MOV    #-1,R0           ;SET FAILURE STATUS
3349 071500 000137 072122      JMP    QNAEXI           ;AND EXIT
3350
3351 071504 032762 010000 000016 80$:  BIT    #XC,CSR(R2)     ;IS TRANSCEIVER OK?
3352 071512 001410              BEQ    QNAIN1           ;YES
3353 071514              ERRHRD 66,MSG66        ;NO, ERROR.
                                TRAP    C#ERRHRD
                                .WORD   66
                                .WORD   MSG66
                                .WORD   0
3354 071524 012700 177777      MOV    #-1,R0           ;SET FAILURE STATUS
3355 071530 000137 072122      JMP    QNAEXI           ;AND EXIT
3356
3357 071534              QNAIN1:
3358 071534 013702 047772              MOV    QNAADO,R2        ; RESTORE DEVICE ADDRESS TO R2
3359 071540 052737 000002 050246      BIS    #MRESET,CSRBUF  ; SET SOFTWARE INIT BIT
3360 071546 013762 050246 000016      MOV    CSRBUF,CSR(R2)  ; WRITE IT TO THE QNA
3361 071554 012737 000001 003756      MOV    #1,TIMER1      ; TIMER FOR INITIALIZE
3362 071562              5$:
3363 071562 005737 003756              TST    TIMER1          ; INIT DONE?
3364 071566 001375              BNE    5$              ; BRANCH IF NOT
3365 071570 042737 000002 050246      BIC    #MRESET,CSRBUF  ; CLEAR THE RESET BIT
3366 071576 013762 050246 000016      MOV    CSRBUF,CSR(R2)  ; WRITE IT TO THE QNA
3367 071604 013762 047774 000014      MOV    QNAVCO,VECTOR(R2) ; SET THE INTERRUPT VECTOR
3368 071612 012737 024236 024220      MOV    #XRING,XRGCUR   ; SET XMIT POINTER TO BEGINNING OF RING
3369 071620 012737 024236 024224      MOV    #XRING,XRG NXT  ; AGAIN
3370 071626 012737 024260 024222      MOV    #RRING,RRG CUR  ; SET RCV POINTER TO BEGINNING OF RING
3371 071634 012737 024544 024234      MOV    #RRINGH,RRG PRV ; SET PREVIOUS POINTER TO LAST DESCRIPTOR
3372 071642 012737 024260 024226      MOV    #RRING,RRG NXT  ; SET NEXT DESCRIPTOR POINTER
3373 071650 012703 000001              MOV    #1,R3           ; RESET OWNERSHIP/STATUS OF XMIT RING
3374 071654              10$:
3375 071654 013701 024220              MOV    XRG CUR,R1      ; RING ENTRIES
3376 071660 005061 000000              CLR    FLAG(R1)        ; RESET FLAG
3377 071664 005061 000010              CLR    STAT1(R1)       ; RESET STATUS
3378 071670 005061 000012              CLR    STAT2(R1)       ; RESET STATUS 2
3379 071674 012761 024574 000004      MOV    #XRG001,LOADD(R1)
3380 071702              CALL   GETXNX #0,#XRG CUR
                                MOV    #XRG CUR,(R5)-
                                MOV    #0,(R5)-
                                JSR    R4,PREG14
                                .WORD  GETXNX ANCHOR

```

```

3381 071720 005303          DEC      R3
3382 071722 001354          BNE     10#
3383 071724 012703 000017  MOV     @15.,R3          ; ...AND RECEIVE RING
3384 071730                20#:
3385 071730 013701 024222  MOV     RRGCUR,R1
3386 071734 005061 000000  CLR     FLAG(R1)        ; CLEAR FLAG WORD
3387 071740 005061 000010  CLR     STAT1(R1)       ; CLEAR STATUS 1
3388 071744 005061 000012  CLR     STAT2(R1)       ; CLEAR STATUS 2
3389 071750 012761 100000 000002  MOV     @MVALID,DESC(R1) ; SET THE VALID BIT FOR THIS DESCRIPTOR
3390 071756                CALL    GETRNX @0,@RRGCUR ; UPDATE, BUT DON'T VALIDATE PREVIOUS
                                MOV     @RRGCUR,(R5)+
                                MOV     @0,(R5)+
                                JSR     R4,PREG14
                                .WORD  GETRNX ANCHOR
                                071756 012725 024222
                                071762 012725 000000
                                071766 004437 070656
                                071772 006450
3391 071774 005303          DEC      R3
3392 071776 001354          BNE     20#
3393 072000 012737 024236 024220  MOV     @XRING,XRGCUR   ; SET POINTERS TO BEGINING OF RING
3394 072006 012737 024260 024222  MOV     @RRING,RRGCUR
3395 072014 005037 024546          CLR     RRING+DESC      ; INVALIDATE LAST DESCRIPTOR
3396 072020 005037 050256          CLR     NIRCNT          ; CLEAR BUFFERS RECEIVED COUNTER
3397 072024 012762 024260 000004  MOV     @RRING,LORCV(R2) ; SET LOW RECEIVE LIST ADDRESS
3398 072032 005062 000006          CLR     HIRCV(R2)       ; SET HIGH ADDRESS OF RCV LIST
3399 072036 052737 000501 050246  BIS     @MILOOP!MINTEN!MRCVEN,CSRBUF ; SET BITS IN BUFFERED CSR
3400 072044 013762 050246 000016  MOV     CSRBUF,CSR(R2)  ; WRITE TO THE QNA
3401 072052 012703 004010          MOV     @PHYADR,R3     ; GET PHYSICAL NODE ADDRESS BUF
3402
3403 ; NOW SETUP ADDRESSES IN THE DEQNA
3404 ;
3405 072056 116223 000000          MOVB   NETADD(R2),(R3)+ ; GET BYTE OF NODE ADDRESS
3406 072062 116223 000002          MOVB   NETADD+2(R2),(R3)+ ; GET NEXT BYTE OF NODE ADDRESS
3407 072066 116223 000004          MOVB   NETADD+4(R2),(R3)+ ; GET NEXT BYTE OF NODE ADDRESS
3408 072072 116223 000006          MOVB   NETADD+6(R2),(R3)+ ; GET NEXT BYTE OF NODE ADDRESS
3409 072076 116223 000010          MOVB   NETADD+10(R2),(R3)+ ; GET NEXT BYTE OF NODE ADDRESS
3410 072102 116213 000012          MOVB   NETADD+12(R2),(R3)+ ; GET NEXT BYTE OF NODE ADDRESS
3411 072106                CALL    SETUP @INIADR   ; CALL SETUP WITH INIT ADRESS FUNC CODE
                                MOV     @INIADR,(R5)+
                                JSR     R4,PREG14
                                .WORD  SETUP-ANCHOR
                                072106 012725 000000
                                072112 004437 070656
                                072116 001232
3412 072120                P#POP  RO          ; GET RETURN STATUS
                                MOV     (R5),RO
3413 072122                QNAEXI:
3414 072122                RETURN RO         ; FINISHED, LEAVE, VAMOOSE, ETC.
                                MOV     RO,(R5)+
                                RTS     PC
072122 010025
072124 000207

```

3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472

```

.SBTTL SETUP - SETUP UP NI ADDRESSES IN QNA
: **
: FUNCTIONAL DESCRIPTION:
:
: THIS ROUTINE LETS THE QNA KNOW WHICH ADDRESSES IT IS SUPPOSED TO
: RECOGNIZE.
:
: BEGIN
: POINT TO TARGET ADDRESS TABLE
: GET SETUP TYPE FROM PARAMETER STACK
: CASE SETUP TYPE
: . INITIALIZE TABLE
: . . INITIALIZE ADDRESS GROUP POINTER
: . . DO WHILE ALL ADDRESS GROUPS HAVE NOT BEEN WRITTEN
: . . . INITIALIZE COLUMN POINTER
: . . . POINT TO BASIC ADDRESSES TABLE (MBZ, OUR OWN)
: . . . DO WHILE ALL ADDRESS COLUMNS FOR THIS GROUP ARE DONE
: . . . . IF BASIC ADDRESSES HAVE BEEN WRITTEN
: . . . . THEN
: . . . . POINT TO OUR OWN NI PHYSICAL ADDRESS
: . . . . ENDF
: . . . EXECUTE SUBROUTINE TO FILL IN THIS ADDRESS
: . . ENDDO
: . ENDDO
: . ADD DECNET CONSOLE ID MULTICAST ADDRESS
: . . POINT TO MULTICAST ADDRESS
: . . INIT GROUP POINTER
: . . INIT COLUMN POINTER TO MULTICAST ADDRESS COLUMN
: . . INIT ROW POINTER
: . . EXECUTE ROUTINE TO FILL IN THIS ADDRESS
: . KILL DECNET CONSOLE ID MULTICAST ADDRESS
: . . POINT TO MULTICAST ADDRESS
: . . INIT GROUP POINTER
: . . INIT COLUMN POINTER TO MULTICAST ADDRESS COLUMN
: . . INIT ROW POINTER
: . . EXECUTE ROUTINE TO FILL IN THIS ADDRESS
: ENDCASE
: VALIDATE THE XMIT DESCRIPTOR FOR A SETUP
: EXECUTE ROUTINE TO XMIT THE SETUP PACKET
: IF STATUS = SUCCESS
: . THEN
: . . SET STATUS = ERROR
: . . EXECUTE ROUTINE TO FIND A RECEIVED PACKET
: . . IF ERROR WAS DETECTED
: . . . THEN
: . . . . REPORT RECEIVE ERROR
: . ELSE
: . . IF NO PACKET HAS BEEN FOUND
: . . . THEN
: . . . . REPORT PACKET NOT RECEIVED
: . ELSE
: . . IF PACKET IS NOT A SETUP PACKET
: . . . THEN
: . . . . REPORT UNEXPECTED PACKET
: . . ELSE
: . . . IF THE XMITTED PACKET SIZE NOT = RECEIVED PACKET SIZE
: . . . . THEN

```

```

3473 ; . . . . . REPORT BYTE COUNT ERROR
3474 ; . . . . . ELSE
3475 ; . . . . . IF XMITTED SETUP DATA NOT = RECEIVED SETUP DATA
3476 ; . . . . . THEN
3477 ; . . . . . REPORT ADDRESS MISMATCH
3478 ; . . . . . ELSE
3479 ; . . . . . STATUS = SUCCESS
3480 ; . . . . . ENDFIF
3481 ; . . . . . ENDFIF
3482 ; . . . . . ENDFIF
3483 ; . . . . . ENDFIF
3484 ; . . . . . ENDFIF
3485 ; ENDFIF
3486 ; RETURN STATUS = STATUS
3487 ; RETURN WITH RETURN STATUS
3488 ; END
3489 ;
3490 ; CALLING SEQUENCE
3491 ;
3492 ; TBD
3493 ;
3494 ; INPUT
3495 ;
3496 ; NONE
3497 ;
3498 ; IMPLICIT INPUT
3499 ;
3500 ; NONE
3501 ;
3502 ; OUTPUT
3503 ;
3504 ; NONE
3505 ;
3506 ; IMPLICIT OUTPUT
3507 ;
3508 ; NONE
3509 ;
3510 ; COMPLETION CODES
3511 ;
3512 ; NONE
3513 ;
3514 ; SIDE EFFECTS
3515 ;
3516 ; NONE
3517 ;
3518 ; REGISTERS USED
3519 ;
3520 ; R0 - RETURN STATUS, SET BY CALLED ROUTINES
3521 ; R1 - CASE OFFSET REG, ADDRESS GROUP POINTER
3522 ; R2 - ADDRESS COLUMN POINTER
3523 ; R3 - POINTS TO TABLE OF NI ADDRESSES
3524 ;
3525 ; DEBUG
3526 ;
3527 ; NONE
3528 ;
3529 072126 SETUP:

```



```

3530 072126 012700 177777      MOV      # 1,R0      ; ASSUME ERROR
3531 072132 012704 023570      MOV      #TGTADR,R4 ; DEQNA ADDRESS TABLE (NOT NODE TABLE)
3532 072136                    P&POP      R1      ; GET PARAMETER, BUT DON'T LOSE IT
                                MOV      -(R5),R1
3533 072140 014501                    BPL      10$      ; BRANCH IF NOT UNDER RANGE
3534 072142                    ERRSF      50,EMSG50 ; REPORT RANGE UNDERFLOW
                                TRAP      C&ERSF
                                .WORD      50
                                .WORD      EMSG50
                                .WORD      0
                                072142 104454
                                072144 000062
                                072146 065544
                                072150 000000
3535 072152 000536                    BR      SETEXI
3536 072154                    10$:
3537 072154 022701 000002      CMP      #2,R1      ; CHECK UPPER LIMIT OF CASE OFFSET
3538 072160 002005                    BGE      20$      ; BRANCH IF WITHIN RANG
3539 072162                    ERRSF      51,EMSG51 ; REPORT RANGE OVERFLOW
                                TRAP      C&ERSF
                                .WORD      51
                                .WORD      EMSG51
                                .WORD      0
                                072162 104454
                                072164 000063
                                072166 065600
                                072170 000000
3540 072172 000526                    BR      SETEXI      ; LEAVE
3541 072174                    20$:
3542 072174 006301                    ASL      R1      ; TURN INTO WORD INDEX
3543 072176 062701 072204      ADD      #30$,R1    ; POINT TO WORD OFFSET
3544 072202 061107                    ADD      (R1),PC   ; GO DO IT
3545 072204                    30$:
3546 072204 000006                    .WORD   SETINI - 30$ ; OFFSET TO INITIALIZE
3547 072206 000142                    .WORD   SETADD - 30$ ; OFFSET TO SET DECNET MULTICAST
3548 072210 000204                    .WORD   SETKLD - 30$ ; OFFSET TO KILL DECNET MULTICAST
3549 072212                    SETINI:
3550 072212 012701 177700      MOV      #-100,R1   ; GROUP POINTER
3551 072216                    10$:
3552 072216 062701 000100      ADD      #100,R1   ; BUMP THE GROUP POINTER
3553 072222 001427                    BEQ      20$      ; IF ZERO, DO THE FIRST GROUP
3554 072224 022701 000200      CMP      #200,R1  ; HAVE THE TWO GROUPS BEEN WRITTEN?
3555 072230 001024                    BNE      20$      ; BRANCH IF NOT
3556
3557 072232 013701 024220      MOV      XRGCUR,R1 ; GET XMIT PACKET ADDRESS
3558 072236 012761 177700 000006  MOV      #-100,WRDCNT(R1) ; WRITE ALL ADDRESSES
3559 072244                    CALL     SETWRT    ; WRITE THEM
                                JSR      R4,PREG14
                                .WORD   SETWRT ANCHOR
3560 072252                    P&POP      R0      ; ANY ERROR
                                MOV      (R5),R0
3561 072254 001075                    BNE      SETEXI   ; IF YES, GO WRITE THE ADDRESSES
3562 072256 013701 024220      MOV      XRGCUR,R1 ; ELSE, GET NXT XMIT DESCRIPTOR
3563 072262 012761 177705 000006  MOV      # 73,WRDCNT(R1) ; WORD COUNT TO INIT THE MODE BITS
3564 072270                    CALL     SETWRT    ; WRITE THE PACKET
                                JSR      R4,PREG14
                                .WORD   SETWRT ANCHOR
3565 072276                    P&POP      R0      ; GET STATUS
                                MOV      (R5),R0
3566 072300 000463                    BR      SETEXIT   ; LEAVE
3567 072302                    20$:
3568 072302 012702 177777      MOV      # 1,R2    ; INIT THE COLUMN POINTER
3569 072306 012703 004002      MOV      #NOADR,R3 ; POINT TO MBZ AND PHYSICAL ADDRESSES
3570 072312                    30$:
3571 072312 005202                    INC      R2      ; BUMP THE COLUMN POINTER

```

```

3572 072314 022702 000010      CMP      #8.,R2      ; ADDRESS COLUMNS 0 7 INITIALIZED?
3573 072320 001736              BEQ      10$        ; DO NEXT GROUP IF YES
3574 072322 005701              TST     R1         ; IS THIS THE FIRST GROUP:
3575 072324 001003              BNE     40$        ; BRANCH IF NOT
3576 072326 022702 000002      CMP      #2,R2     ; HAVE BASIC ADDRESSES BEEN WRITTEN?
3577 072332 103002              BHIS   50$        ; BRANCH IF NOT
3578 072334              40$:
3579 072334 012703 004010      MOV     #PHYADR,R3 ; E SE. POINT TO OUR PHYSICAL ADDRESS
3580 072340              50$:
3581 072340 004137 073006      JSR     R1,SETFIL ; FILL IN THE ADDRESS
3582 072344 000762              BR      30$        ; GO DO NEXT COLUMN
3583 072346              SETADD:
3584 072346 012703 004032      MOV     #MCSTAD,R3 ; GET DECNET MULTICAST ADDRESS
3585 072352 005001              CLR     R1         ; INIT GROUP POINTER
3586 072354 012702 000004      MOV     #4,R2     ; INIT COLUMN TO CORRECT SLOT
3587 072360 004137 073006      JSR     R1,SETFIL ; FILL IN THIS ADDRESS
3588 072364 013701 024220      MOV     XRGCUR,R1 ; GET CURRENT XMIT DESCRIPTOR
3589 072370 012761 177706 000006  MOV     #72,WRDCNT(R1) ; WRITE ADDRESSES ONLY
3590 072376              CALL    SETWRT    ; WRITE THE NEW ADDRESS TABLE INTO QNA
      072376 004437 070656              JSR     R4,PREG14
      072402 001560              .WORD  SETWRT-ANCHOR
3591 072404              P$POP  R0         ; GET THE RETURNED STATUS
      072404 014500              MOV     -(R5),R0
3592 072406 000420              BR      SETEXI    ; LEAVE
3593 072410              SETKLD:
3594 072410 012703 004010      MOV     #PHYADR,R3 ; GET OUR PHYSICAL ADDRESS
3595 072414 005001              CLR     R1         ; INIT GROUP POINTER
3596 072416 012702 000004      MOV     #4,R2     ; INIT ADDRESS COLUMN POINTER
3597 072422 004137 073006      JSR     R1,SETFIL ; FILL IN THIS ADDRESS
3598 072426 013701 024220      MOV     XRGCUR,R1 ; GET CURRENT XMIT DESCRIPTOR
3599 072432 012761 177706 000006  MOV     #72,WRDCNT(R1) ; WRITE ADDRESSES ONLY
3600 072440              CALL    SETWRT    ; WRITE THE NEW ADDRESS TABLE
      072440 004437 070656              JSR     R4,PREG14
      072444 001560              .WORD  SETWRT-ANCHOR
3601 072446              P$POP  R0         ; GET THE RETURNED STATUS
      072446 014500              MOV     (R5),R0
3602 072450              SETEXI:
3603 072450              RETURN R0        ; RETURN
      072450 010025              MOV     R0,(R5)-
      072452 000207              RTS      PC

```

.SBTTL SETWRT WRITE A SETUP PACKET

3605  
3606  
3607  
3608  
3609  
3610  
3611  
3612  
3613  
3614  
3615  
3616  
3617  
3618  
3619  
3620  
3621  
3622  
3623  
3624  
3625  
3626  
3627  
3628  
3629  
3630  
3631  
3632  
3633  
3634  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651  
3652  
3653  
3654  
3655  
3656  
3657  
3658

072454  
072454 005200  
072456 013704 024224  
072462 012764 023570 000004  
072470  
072470 012725 000000  
072474 004437 070656  
072500 002326  
072502 005737 050252  
072506 001407  
072510

```

; **
; FUNCTIONAL DESCRIPTION:
;
; THIS ROUTINE TAKES CARE OF WRITING SETUP PACKETS AND VERIFYING THAT
; ALL OF IT OCCURRED OKAY
;
; CALLING SEQUENCE
;
; CALL SETWRT
;
; INPUT
;
; NONE
;
; IMPLICIT INPUT
;
; NONE
;
; OUTPUT
;
; NONE
;
; IMPLICIT OUTPUT
;
; NONE
;
; COMPLETION CODES
;
; NONE
;
; SIDE EFFECTS
;
; NONE
;
; REGISTERS USED
;
; R1 - HOLDS TRANSMITTED ADDRESS TABLE ADDRESS
; R2 - HOLDS ADDRESS OF PACKET
; R3 - AND ADDRESS OF RECEIVED ADDRESS TABLE
; R4 COUNT OF NUMBER OF BYTES TO CHECK
;
; DEBUG
;
; NONE
;
; SETWRT:
; WRITE NEW ADDRESS TABLE INTO THE QNA
; ASSUME ERROR
; GET DESCRIPTOR
;
; TRANSMIT THE PACKET
;
; MOV @XMTSET.(R5).
; JSR R4,PREG:4
; .WORD XMIT ANCHOR
;
; TST FATFLG ; NON EXISTENT MEMORY?
; BEQ 20$ ; BRANCH IF NOT
;
10$:
```

```

3659 072510 005037 050252          CLR  FATFLG          ; CLEAR FATAL ERROR FLAG
3660 072514          ERRHRD  43,MSG43      ; REPORT TIMEOUT
                                TRAP  C$ERRHRD
                                .WORD  43
                                .WORD  MSG43
                                .WORD  0
3661 072524 000504          BR    WRTEXI         ; LEAVE
3662 072526          20$: P$POP  R1          ; GET STATUS
3663 072526          014501          MOV    (R5),R1
3664 072530 001405          BEQ   30$           ; CONTINUE IF OKAY
3665 072532          ERRHRD  45,MSG45      ; REPORT XMIT ERROR
                                TRAP  C$ERRHRD
                                .WORD  45
                                .WORD  MSG45
                                .WORD  0
3666 072542 000475          BR    WRTEXI         ; EXIT SETUP, FAILURE OCCURRED
3667 072544          30$: CALL  RECEVE
3668 072544          004437 070656          JSR   R4,PREG14
                                .WORD  RECEVE ANCHOR
3669 072552 005737 050252          TST  FATFLG         ; NONE EXISTENT MEMORY ERROR?
3670 072556 001354          BNE  10$           ; BRANCH IF YES
3671 072560 005700          TST  R0            ; DID AN ERROR OCCUR?
3672 072562 001401          BEQ  40$           ; BRANCH IF NOT
3673 072564 000455          BR   SETVAL        ; EXIT SETUP, ERROR ALREADY REPORTED
3674 072566          40$: P$POP  R2          ; WERE ANY PACKETS RECEIVED
3675 072566          014502          MOV    (R5),R2
3676 072570 001005          BNE  50$           ; BRANCH IF YES
3677 072572          ERRHRD  46,MSG46      ; NO PACKET RECEIVED
                                TRAP  C$ERRHRD
                                .WORD  46
                                .WORD  MSG46
                                .WORD  0
3678 072602 000455          BR   WRTEXI
3679 072604          50$: MOV   RRGXNT,R2      ; GET ADDRESS OF THE PACKET
3680 072604 013702 024226          BIT  @MRSETP,STAT1(R2) ; SETUP PACKET?
3681 072610 032762 020000 000010    BNE  60$           ; BRANCH IF IT IS
3682 072616 001005          ERRHRD  47,MSG47      ; REPORT UNEXPECTED PACKET STATUS
3683 072620          104456          TRAP  C$ERRHRD
3684 072622          000057          .WORD  47
3685 072624          065366          .WORD  MSG47
3686 072626          000000          .WORD  0
3687 072630 000433          BR   SETVAL        ; EXIT
3688 072632          60$: MOV   LOADD(R4),R1    ; GET XMITTED ADDR SS TABLE
3689 072636 016401 000004          MOV   WRDCNT(R4),R4  ; GET XMIT SIZE
3690 072642 005404          NEG  R4            ; CONVERT FROM TWOS COMPLEMENT
3691 072644 006304          ASL  R4            ; TURN WORD COUNT TO BYTE COUNT
3692          ;; N.M. ADD  @12,R4      ; COMPENSATE FOR GARBAGE BYTES
3693 072646 016203 000004          MOV   LOADD(R2),R3  ; GET RECEIVED PACKET
3694 072652 120462 000012          CMPB R4,STAT2(R2)  ; IS THE SIZE THE SAME?
3695 072656 001405          BEQ  70$           ; BRANCH IF IT IS
3696 072660          ERRHRD  48,MSG48      ; BYTE COUNT ERROR
                                TRAP  C$ERRHRD
                                .WORD
                                .WORD
                                .WORD
072660 104456

```

```

072662 000060
072664 065446
072666 000000
3695 072670 000413
3696 072672
3697
3698 072672 006204
3699 072674
3700 072674 022123
3701 072676 001405
3702 072700
072700 104456
072702 000054
072704 065141
072706 000000
3703 072710 000403
3704 072712
3705 072712 005304
3706 072714 001367
3707 072716 005000
3708 072720
3709 072720
072720 012725 024226
072724 012725 000001
072730 004437 070656
072734 006450
3710 072736
3711 072736 013704 024224
3712 072742 012764 024574 000004
3713 072750 005064 000000
3714 072754 005064 000010
3715 072760 005064 000012
3716 072764
072764 012725 024224
072770 012725 000000
072774 004437 070656
073000 006534
3717 073002
073002 010025
073004 000207

                                .WORD 48
                                .WORD MSG48
                                .WORD 0

                                BR      SETVAL      ; EXIT
70$:
;; N.M. SUB      #12,R4      ; REMOVE GARBAGE BYTE COUNT
    ASR      R4      ; TURN BYTE COUNT INTO WORD COUNT
80$:
    CMP      (R1)+,(R3)+      ; DOES XMITTED = RECEIVED?
    BEQ      90$      ; BRANCH IF NOT
    ERRMRD   44,MSG44      ; COMPARE ERROR

                                TRAP   C$ERMPD
                                .WORD 44
                                .WORD MSG44
                                .WORD 0

                                BR      SETVAL      ; LEAVE
90$:
    DEC      R4      ; DECREMENT THE BYTE COUNT
    BNE      80$      ; CHECK NEXT ADDRESS
    CLR      R0      ; SET SUCCESS CODE IN R0
    SETVAL:  CALL   GETRNX #1,#RRGNXT ; REVALIDATE THE DESCRIPTOR
                                ; UPDATE NEXT AND PREVIOUS PACKET ADDRESS
                                MOV     #RRGNXT,(R5)+
                                MOV     #1,(R5)+
                                JSR     R4,PREG14
                                .WORD  GETRNX-ANCHOR

WRTEXI:
    MOV      XRG NXT,R4      ; GET TRANSMIT PACKET ADDRESS AGAIN
    MOV      #XRG001,LOADD(R4) ; RESET ADDRESS
    CLR      FLAG(R4)      ; REINIT THE DESCRIPTOR
    CLR      STAT1(R4)
    CLR      STAT2(R4)
    CALL     GETXNX #0,#XRG NXT ; GET NEXT XMIT BUFFER
                                MOV     #XRG NXT,(R5)+
                                MOV     #0,(R5)+
                                JSR     R4,PREG14
                                .WORD  GETXNX ANCHOR

                                RETURN R0      ; RETURN WITH STATUS
                                MOV     R0,(R5)+
                                RTS      PC

```

```

3719                                     .SBTTL SETFIL - FILL AN ADDRESS SLOT IN SETUP TABLE
3720                                     : **
3721                                     : FUNCTIONAL DESCRIPTION:
3722                                     :
3723                                     :     THIS
3724                                     :
3725                                     : BEGIN
3726                                     : INIT THE ROW POINTER
3727                                     : DO UNTIL 6 ADDRESS BYTES OF THE ADDRESS ARE WRITTEN
3728                                     :     ADD ROW NUMBER TO INDEX
3729                                     :     ADD COLUMN NUMBER TO INDEX
3730                                     :     ADD TABLE ADDRESS TO INDEX
3731                                     :     STUFF THE ADDRESS BYTE IN
3732                                     : ENDDO
3733                                     : BUMP ROW POINTER
3734                                     : STUFF A ZERO INTO THIS BYTE (MBZ)
3735                                     : BUMP ROW POINTER
3736                                     : STUFF A ZERO INTO THIS BYTE (MBZ)
3737                                     : RETURN FROM SUB
3738                                     : END
3739                                     :
3740                                     : CALLING SEQUENCE
3741                                     :
3742                                     :     JSR     PC,SETFIL
3743                                     :
3744                                     : INPUT
3745                                     :
3746                                     :     R1 = GROUP POINTER
3747                                     :     R2 = COLUMN POINTER
3748                                     :     R3 = POINTER TO NI ADDRESS
3749                                     :     R4 = ROW POINTER
3750                                     :
3751                                     : IMPLICIT INPUT
3752                                     :
3753                                     :     NONE
3754                                     :
3755                                     : OUTPUT
3756                                     :
3757                                     :     NONE
3758                                     :
3759                                     : IMPLICIT OUTPUT
3760                                     :
3761                                     :     NONE
3762                                     :
3763                                     : COMPLETION CODES
3764                                     :
3765                                     :     NONE
3766                                     :
3767                                     : SIDE EFFECTS
3768                                     :
3769                                     :     NONE
3770                                     :
3771                                     : REGISTERS USED
3772                                     :
3773                                     :     R1  INDEX
3774                                     :     R2  ADDRESS COLUMN POINTER
3775                                     :     R3  POINTS TO GROUPS OF ADDRESSES

```

```

3776          ; R4 ADDRESS ROW POINTER
3777          ;
3778          ; DEBUG
3779          ;
3780          ; NONE
3781          ;
3782          ; INIT THE ROW POINTER
3783          ; DO WHILE 6 ADDRESS BYTES OF THE ADDRESS ARE TO BE WRITTEN
3784 073006 SETFIL:
3785 073006 MOV    @TGTADR,R4      ; SET ROW POINTER TO DESTINATION TABLE
3786 073012 ADD    R2,R4        ; ADD COLUMN TO IT
3787 073014 ADD    (SP),R4     ; ADD SAVED GROUP TO IT
3788 073016 PUSH   R1          ; SAVE RETURN ADDRESS
3789          ;                               MOV    R1, -(SP)
3789 073020 MOV    R4,R1        ; INIT LIMIT TO CURRENT ADDRESS
3790 073022 ADD    @40.,R1     ; UPPER LIMIT (5 ADRS BYTES + 8 BYTE OFFSET TO
3791          ;                               ; NEXT ROW)
3792 073026          ;
3793 073026 10$: MOVB   (R3), (R4)    ; STUFF THE ADDRESS BYTE IN
3794 073030 ADD    @8.,R4      ; ELSE, POINT TO NEXT ROW IN THE COLUMN
3795 073034 CMP    R4,R1      ; HAVE ALL ADDRESSES BEEN WRITTEN
3796 073036 BGT   20$        ; BRANCH IF YES
3797 073040 BR    10$        ; AND STUFF IT IN
3798 073042          ;
3799 073042 20$: MOVB   (R3), (R4)    ; STASH LAST ADDRESS BYTE
3800 073044 MOVB   @0., (R4)    ; NEXT BYTE MBZ
3801 073050 ADD    @8.,R4      ; POINT TO NEXT BYTE
3802 073054 MOVB   @0., (R4)    ; NEXT BYTE MBZ
3803 073060 POP    R1          ; RESTORE RETURN ADDRESS
3804          ;                               MOV    (SP), R1
3804 073062 RTS    R1

```

3806  
 3807  
 3808  
 3809  
 3810  
 3811  
 3812  
 3813  
 3814  
 3815  
 3816  
 3817  
 3818  
 3819  
 3820  
 3821  
 3822  
 3823  
 3824  
 3825  
 3826  
 3827  
 3828  
 3829  
 3830  
 3831  
 3832  
 3833  
 3834  
 3835  
 3836  
 3837  
 3838  
 3839  
 3840  
 3841  
 3842  
 3843  
 3844  
 3845  
 3846  
 3847  
 3848 073064  
 3849 073064 010146  
 3850 073066 010346  
 3851 073070 013701 047772  
 3852 073074 016103 000016  
 3853 073100 032703 000004  
 3854 073104 001403  
 3855 073106 005237 050252  
 3856 073112 000436  
 3857 073114  
 3858 073114 032703 100000  
 3859 073120 001423  
 3860 073122 005237 050256  
 3861 073126 052737 100000 050246  
 3862 073134 013761 050246 000016

.SBTTL QNA INTERRUPT SERVICE ROUTINE

```

; **
; FUNCTIONAL DESCRIPTION:
;     THIS IS THE INTERRUPT SERVICE ROUTINE
; CALLING SEQUENCE
;     NONE
; INPUT
;     NONE
; IMPLICIT INPUT
;     NONE
; OUTPUT
;     NONE
; IMPLICIT OUTPUT
;     NONE
; COMPLETION CODES
;     NONE
; SIDE EFFECTS
;     NONE
; REGISTERS USED
;     NONE
; DEBUG
;     NONE
; --
QNAISR:
MOV     R1, -(SP)           ; SAVE R1
MOV     R3, -(SP)           ; ...
MOV     QNAADO, R1         ; GET QNA ADDRESS
MOV     CSR(R1), R3        ; AND ITS CSR CONTENTS
BIT     @MIXM, R3          ; INTERRUPT RESULT OF MEMORY TIMEOUT?
BEQ     10$,
INC     FATFLG             ; SET FLAG
BR      40$,              ; CONTINUE

10$:
BIT     @MRCVIN, R3        ; RECV INTERRUPT ??
BEQ     20$,              ; NO
INC     NIRCNT             ; YES, SET FLAG
BIS     @MRCVIN, CSRBUF    ; SET RECEIVER DONE BIT IN BUFFERED CSR
MOV     CSRBUF, CSR(R1)    ; WRITE IT TO REARM INTERRUPTS

```



3863	073142	042737	100000	050246	BIC	#MRCVIN,CSRBUF	; CLEAR THE BIT
3864	073150	032703	000040		BIT	#MINVRC,R3	; RECEIVE LIST INVALID?
3865	073154	001405			BEQ	20%	; NO
3866	073156	012761	024260	000004	MOV	#RRING,LORCV(R1)	; SET LOW ADDRESS OF RINGS
3867	073164	005061	000006		CLR	MIRCV(R1)	; CLEAR UPPER ADDRESS BITS
3868	073170			20%:			
3869	073170	032703	000200		BIT	#MXMTIN,R3	; TRANSMIT INTERRUPT ??
3870	073174	001403			BEQ	30%	; NO
3871	073176	005037	050260		CLR	XFLAG	; YES, INDICATE THAT XMIT HAS OCCURRED
3872	073202	000404			BR	50%	; LEAVE
3873	073204			30%:			
3874	073204	005237	050266		INC	BCOUNT	; ELSE, NONSENSE INTERRUPT
3875	073210			40%:			
3876	073210	005237	050270		INC	ERRFLG	
3877	073214			50%:			
3878	073214	012603			MOV	(SP),R3	; RESTORE REGISTERS
3879	073216	012601			MOV	(SP),R1	; RESTORE REGISTERS
3880	073220	000002			RTI		; AUF WIEDERSEHEN
3881							

3883  
3884  
3885  
3886  
3887  
3888  
3889  
3890  
3891  
3892  
3893  
3894  
3895  
3896  
3897  
3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905  
3906  
3907  
3908  
3909  
3910  
3911  
3912  
3913  
3914  
3915  
3916  
3917  
3918  
3919  
3920  
3921  
3922  
3923  
3924  
3925  
3926  
3927  
3928  
3929  
3930  
3931  
3932  
3933  
3934  
3935  
3936  
3937  
3938  
3939

073222  
073222 005037 050274  
073226  
073226 012737 000020 002402  
073234 012737 000020 002404  
073242  
073242 013704 024220  
073246 005714  
073250 001405  
073252 005764 000010  
073256 001002  
073260 000137 073650

```
.SBTTL XMIT TRANSMIT QNA PACKETS
***
; FUNCTIONAL DESCRIPTION:
;
; THIS ROUTINE SETS UP THE TRANSMIT DESCRIPTOR RING ENTRIES AND WRITES
; THE RING ADDRESS INTO THE TRANSMIT LOW AND HIGH ADDRESS REGISTERS TO
; START TRANSMISSION.
;
; CALLING SEQUENCE
;
; CALL XMIT #XMTSET OR #XMTDAT ; XMTSET = SETUP, XMTDAT = DATA
; P#POP P1
;
; INPUT
;
; NONE
;
; IMPLICIT INPUT
;
; NONE
;
; OUTPUT
;
; P1 = STATUS - 0 = SUCCESS, -1 FOR FAILURE
;
; IMPLICIT OUTPUT
;
; NONE
;
; COMPLETION CODES
;
; 0 = SUCCESS
; -1 = FAILURE
;
; SIDE EFFECTS
;
; NONE
;
; REGISTERS USED
;
;
; DEBUG
;
; NONE
;
;--
XMIT::
CLR RETRYS
10$:
MOV #16.,LSRTRY ; SET CARRIER LOSS RETRY COUNT
MOV #16.,NCRTRY ; SET NO CARRIER RETRY COUNT
20$:
MOV XRGCUR,R4 ; MOVE RING ENTRY LOCATION INTO R4
TST (R4) ; MAKE SURE WE OWN THIS
BEQ 30$ ; BRANCH IF WE DO
TST STAT1(R4) ; ELSE, SEE IF STATUS IS WRITTEN
BNE 30$ ; IF YES, CONTINUE
JMP 160$ ; ELSE REPORT THAT QNA STILLS OWNS IT
```

```

3940 073264          30$:
3941 073264          P$POP  R1          ; GET XMIT TYPE
          014501          MOV      -(R5),R1
3942 073266 001007          BNE     40$          ; BRANCH IF NOT A SETUP PACKET
3943 073270 012764 130000 000002  MOV     #MVALID!MEOM!MXSETP,DESC(R4) ; SET DESCRIPTOR BITS
3944 073276 012764 023570 000004  MOV     #TGTADR,LOADD(R4)          ; SET BUFFER ADDRESS
3945 073304 000431          BR      50$
3946 073306          40$:
3947 073306 005737 003202          TST     RSPFLG          ; ARE WE IN LISTEN/RESPOND MODE?
3948 073312 001020          BNE     45$          ; BRANCH IF YES, JUST VALIDATE AND XMIT
3949 073314 013764 050332 000006  MOV     BUFLN,WRDCNT(R4)          ; BUF LENGTH TO FIRST WORD OF NEXT RING
3950 073322 006264 000006          ASR     WRDCNT(R4)          ; TURN IT INTO A WORD COUNT
3951 073326 005464 000006          NEG     WRDCNT(R4)          ; TWO'S COMPLEMENT THE COUNT
3952 073332 016402 000004          MOV     LOADD(R4),R2          ; GET ADDRESS OF BUFFER
3953 073336 062702 000006          ADD     #6,R2             ; POINT TO SOURCE ADDRESS FIELD
3954 073342 012703 004010          MOV     #PHYADR,R3          ; POINT TO OUR PHYSICAL ADDRESS
3955 073346 012322          MOV     (R3)+,(R2)+          ; FILL IN FIRST TWO ADDRESS BYTES
3956 073350 012322          MOV     (R3)+,(R2)+          ; AND NEXT TWO ADDRESS BYTES
3957 073352 012322          MOV     (R3)+,(R2)+          ; AND NEXT TWO ADDRESS BYTES
3958 073354          45$:
3959 073354 012764 120000 000002  MOV     #MVALID!MEOM,DESC(R4)      ; SET VALID BIT
3960 073362 012764 024574 000004  MOV     #XRG001,LOADD(R4)          ; SET BUFFER ADDRESS
3961 073370          50$:
3962 073370 005064 000000          CLR     FLAG(R4)           ; CLEAR FLAG WORD
3963 073374 005064 000010          CLR     STAT1(R4)          ; CLEAR STATUS WORD 1
3964 073400 005064 000012          CLR     STAT2(R4)          ; CLEAR STATUS WORD 2
3965 073404 012737 000001 050260  MOV     #1,XFLAG           ; SET TRANSMIT FLAG
3966 073412 013703 047772          MOV     QNAADO,R3           ; GET DEVICE ADDRESS
3967 073416 012737 001130 003760  MOV     #1130,TIMER2        ; SET TIMER
3968 073424          60$:
3969 073424 032763 000020 000016  BIT     #MINVXM,CSR(R3)        ; IS XMIT LIST INVALID?
3970 073432 001004          BNE     70$           ; YES, CONTINUE
3971 073434 005737 003760          TST     TIMER2           ; TIMEOUT YET
3972 073440 001371          BNE     60$           ; NO
3973 073442 000514          BR      180$          ; TIMEOUT, REPORT ERROR AND LEAVE
3974 073444          70$:
3975 073444 010463 000010          MOV     R4,LOXMT(R3)        ; LOW ADDRESS OF XMIT BUFFER
3976 073450 005063 000012          CLR     HIXMT(R3)          ; START TRANSMISSION
3977 073454 012701 003760          MOV     #TIMER2,R1         ; SET UP TO WAIT FOR TRANSMIT TO COMPLETE
3978 073460 012711 001130          MOV     #1130,(R1)
3979 073464          80$:
3980 073464 005737 050260          TST     XFLAG             ; SEE IF TRANSMIT DONE BIT SET
3981 073470 001407          BEQ     100$            ; IF SET, SKIP WAIT LOOP
3982 073472 005737 050252          TST     FATFLG           ; NON EXISTENT MEMORY?
3983 073476 001401          BEQ     90$             ; BRANCH IF NOT
3984 073500 000503          BR      190$            ; LEAVE ROUTINE
3985 073502          90$:
3986 073502 005711          TST     (R1)             ; ELSE, SEE IF TIMEOUT YET
3987 073504 001367          BNE     80$             ; NO, WAIT
3988 073506 000472          BR      180$            ; YES, EXIT
3989 073510          100$:
3990 073510 052737 000200 050246  BIS     #MXMTIN,CSRBUF        ; SET XMIT DONE BIT IN BUFFERED CSR
3991 073516 013763 050246 000016  MOV     CSRBUF,CSR(R3)        ; REARM XMIT INTERRUPTS
3992 073524 042737 000200 050246  BIC     #MXMTIN,CSRBUF        ; CLEAR THE BIT IN BUFFERED CSR
3993 073532 005764 000010          TST     STAT1(R4)          ; SEE WHO OWNS THIS ENTRY
3994 073536 001444          BEQ     160$            ; IF QNA STILL OWNS THIS, SOMETHING IS WRONG
3995 073540 032764 010000 000002  BIT     #MXSETP,DESC(R4)      ; SETUP PACKET?

```

3996	073546	001006			BNE	105:			; BRANCH IF IT IS, WE DON'T COUNT SETUP STATISTICS
3997	073550				CALL		BMPCNT R4, @BMPXMT		; ELSE, GO UPDATE THE COUNTERS
	073550	012725	000001						MOV @BMPXMT, (R5).
	073554	010425							MOV R4, (R5).
	073556	004437	070656						JSR R4, PREG14
	073562	003736							.WORD BMPCNT-ANCHOR
3998	073564					105:			
3999	073564	032764	040000	000010	BIT		@MXERRS, STAT1(R4)		; SEE IF ANY ERRORS
4000	073572	001011			BNE		120:		; IF YES, BRANCH AND TAKE CARE OF THEM
4001	073574					110:			
4002	073574				CALL		GETXNX @0, @XRGCUR		; UPDATE 'TRANSMIT RING CURRENT' POINTER
	073574	012725	024220						MOV @XRGCUR, (R5).
	073600	012725	000000						MOV @C, (R5).
	073604	004437	070656						JSR R4, PREG14
	073610	006534							.WORD GETXNX-ANCHOR
4003	073612	005003			CLR		R3		; INDICATE SUCCESS
4004	073614	000437			BR		200:		; RETURN
4005	073616					120:			
4006	073616	032764	016000	000010	BIT		@MABORT!MLOSS!MNOCAR, STAT1(R4)		; WAS MESSAGE STILL SENT?
4007	073624	001763			BEQ		110:		; BRANCH IF YES, DO NEXT ONE
4008	073626					140:			
4009	073626	005237	050274		INC		RETRY5		; LET'S TRY IT AGAIN, KEEP COUNT OF THEM
4010	073632	022737	000003	050274	CMP		@3, RETRY5		; HAVE WE RETRIED THREE TIMES YET?
4011	073640	100001			BPL		150:		; IF NOT, THEN LET'S RETRANSMIT
4012	073642	000754			BR		110:		; ELSE, CLEAN UP AND LEAVE
4013	073644					150:			
4014	073644	000137	073370		JMP		50:		; TRY AGAIN
4015	073650					160:			
4016	073650				ERRDF		10, MSG10, ERR1		; TRANSMIT RING BOOKKEEPING ERROR
	073650	104455							TRAP C\$ERDF
	073652	000012							.WORD 10
	073654	063670							.WORD MSG10
	073656	070306							.WORD ERR1
4017	073660	000413			BR		190:		
4018	073662					170:			
4019	073662				ERRDF		12, MSG10, ERR1		; BOOKKEEPING ERROR, XRGXNT SHOULD = XRGCUR
	073662	104455							TRAP C\$ERDF
	073664	000014							.WORD 12
	073666	063670							.WORD MSG10
	073670	070306							.WORD ERR1
4020	073672	000406			BR		190:		
4021	073674					180:			
4022	073674	005237	050272		INC		TIMOUT		
4023	073700				ERRHRD		8, MSG08, ERR1		; REPORT ERROR
	073700	104456							TRAP C\$ERHRD
	073702	000010							.WORD 8
	073704	063605							.WORD MSG08
	073706	070306							.WORD ERR1

4024 073710  
 4025 073710 012703 177777  
 4026 073714  
 4027 073714 005064 000000  
 4028 073720 005064 000010  
 4029 073724 005064 000012  
 4030 073730 005064 000002  
 4031 073734  
       073734 010325  
       073736 000207

190\$:  
 200\$:  
 MOV # 1,R3  
 CLR FLAG(R4)  
 CLR STAT1(R4)  
 CLR STAT2(R4)  
 CLR DESC(R4)  
 RETURN R3

; ERROR INDICATOR  
 ; CLEAR FLAG WORD  
 ; CLEAR STATUS WORD 1  
 ; CLEAR STATUS WORD 2  
 ; INVALIDATE THE DESCRIPTOR  
 ; RETURN  
 MOV R3,(R5)+  
 RTS PC

4033  
4034  
4035  
4036  
4037  
4038  
4039  
4040  
4041  
4042  
4043  
4044  
4045  
4046  
4047  
4048  
4049  
4050  
4051  
4052  
4053  
4054  
4055  
4056  
4057  
4058  
4059  
4060  
4061  
4062  
4063  
4064  
4065  
4066  
4067  
4068  
4069  
4070  
4071  
4072  
4073  
4074  
4075  
4076  
4077  
4078 073740  
4079 073740 005000  
4080 073742 005002  
4081 073744  
4082 073744 005737 050256  
4083 073750 001002  
4084 073752 000137 074626  
4085 073756  
4086 073756 013704 024222  
4087 073762 016401 000010  
4088 073766 001002  
4089 073770 000137 074606

.SBTTL RECEVE RECEIVE QNA RING BUFFERS

```

; **
; FUNCTIONAL DESCRIPTION:
;
; THIS ROUTINE TAKES INCOMING DATA BUFFERS FROM THE QNA AND CHECK FOR
; ERRORS. THIS PROCESS CONTINUES FOR ALL COMPLETED BUFFERS
;
; CALLING SEQUENCE
;
; CALL    RECEVE
; P$POP  P1
;
; INPUT
;
; NONE
;
; IMPLICIT INPUT
;
; NONE
;
; OUTPUT
;
; NUMBER OF PACKETS PROCESSED BY THIS ROUTINE
;
; IMPLICIT OUTPUT
;
; NONE
;
; COMPLETION CODES
;
; NONE
;
; SIDE EFFECTS
;
; NONE
;
; REGISTERS USED
;
; NONE
;
; DEBUG
;
; NONE
;
RECEVE:
  CLR    R0                ; SET STATUS TO SUCCESS
  CLR    R2                ; CLEAR PACKETS HANDLED COUNTER
10$:   TST    NIRCNT        ; SEE IF ANY PACKETS TO RECEIVE
      BNE    20$           ; IF YES, CONTINUE
      JMP    180$          ; ELSE, EXIT
20$:   MOV    RRGCUR,R4    ; MOVE CURRENT RCV RING POINTER TO R4
      MOV    STAT1(R4),R1 ; MOVE STATUS OF PACKET TO R1
      BNE    30$           ; BRANCH IF WE OWN IT
      JMP    160$          ; STILL OWNED BY QNA, ERROR

```

```

4090 073774          30$:
4091 073774 032764 020000 000010 BIT    @MRSETP,STAT1(R4) ; WAS A SETUP PACKET RECEIVED?
4092 074002 001402      BEQ    32$ ; BRANCH IF NOT
4093 074004 000137 074560      JMP    150$ ; ELSE, LEAVE
4094 074010          32$:
4095 074010          CALL   BMCPCNT R4,@BMPRCV ; UPDATE COUNTERS FOR RECEIVE
      074010 012725 000000          MOV    @BMPRCV,(R5).
      074014 010425          MOV    R4,(R5).
      074016 004437 070656          JSR   R4,PREG14
      074022 003736          .WORD  BMCPCNT-ANCHOR
4096 074024 016403 000004      MOV    LOADD(R4),R3 ; MOVE BUFFER ADDRESS INTO R3
4097 074030 005737 002200      TST   BLDPLG ; ARE WE IN BUILD MODE?
4098 074034 001035      BNE   321$ ; BRANCH IF WE ARE
4099 074036 023713 004010      CMP   PHYADR,(R3) ; CHECK TO SEE IF THE DESTINATION ADDRESS IS OURS
4100 074042 001010      BNE   350$ ; BRANCH IF NOT
4101 074044 023763 004012 000002      CMP   PHYADR+2,2(R3) ; CHECK NEXT WORD OF ADDRESS
4102 074052 001004      BNE   350$ ; BRANCH IF NOT OURS
4103 074054 023763 004014 000004      CMP   PHYADR+4,4(R3) ; CHECK FINAL ADDRESS WORD
4104 074062 001411      BEQ   380$ ; BRANCH IF EVERYTHING IS OKAY
4105
4106 074064          350$:
4107 074064 032701 004000      BIT    @MRUNT,R1 ;: N.M.
4108 074070 001037      BNE   322$ ; RUNT PACKET
4109 074072          ERRHRD 61,EMSG61 ; IF YES, THROW OUT THE PACKET.
      074072 104456          TRAP   C$ERRHRD ; REPORT UNKNOWN RECEIVER ERROR
      074074 000075          .WORD  61
      074076 066412          .WORD  EMSG61
      074100 000000          .WORD  0
4110 074102 000137 074544      JMP    130$ ; CONTINUE
4111
4112 074106          380$:
4113 074106 032701 004000      BIT    @MRUNT,R1 ;: N.M.
4114 074112 001451      BEQ   38$ ; RUNT PACKET
4115 074114          ERRHRD 54,EMSG54 ; BRANCH IF NOT
      074114 104456          TRAP   C$ERRHRD ; REPORT THE ERROR
      074116 000066          .WORD  54
      074120 065773          .WORD  EMSG54
      074122 000000          .WORD  0
4116 074124 000137 074544      JMP    130$ ; CONTINUE
4117
4118 074130          321$:
4119 074130 023713 004032      CMP   MCSTAD,(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4120 074134 001011      BNE   35$ ; IF NOT, REPORT ERROR
4121 074136 023763 004034 000002      CMP   MCSTAD+2,2(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4122 074144 001005      BNE   35$ ; IF NOT, REPORT ERROR
4123 074146 023763 004036 000004      CMP   MCSTAD+4,4(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4124 074154 001001      BNE   35$ ; IF NOT, REPORT ERROR
4125 074156 000427      BR    38$ ; ITS A OKAY, MOVE ON
4126 074160          35$:
4127 074160          ERRHRD 68,EMSG68 ; ELSE, REPORT THE ERROR
      074160 104456          TRAP   C$ERRHRD
      074162 000104          .WORD  68
      074164 067121          .WORD  EMSG68
      074166 000000          .WORD  0
4128
4129 074170          322$:
4130 074170 010403      MOV   R4,R3 ; GET DESCRIPTOR ADDRESS

```

```

4131 074172 023703 024232      CMP      RRGLST,R3      ; IS THIS THE LAST DESCRIPTOR?
4132 074176 001003              BNE      36$           ; BRANCH IF NOT
4133 074200 013703 024216      MOV      RRGSR,R3      ; OTHERWISE, GET ADDRESS OF FIRST DESCRIPTOR
4134 074204 000402              BR       37$
4135 074206              36$:
4136 074206 062703 000014      ADD      @14,R3        ; POINT TO NEXT DESCRIPTOR
4137 074212              37$:
4138 074212 005763 000010      TST     STAT1(R3)     ; HAS THE NEXT DESCRIPTOR BEEN USED?
4139 074216 001406              BEQ     375$          ; BRANCH IF NOT
4140 074220 022737 000001 050256      CMP     @1,NIRCNT     ; IF IT IS USED, DO WE RECORD ONLY THE BOGUS RECEIVE
?
4141 074226 001002              BNE     375$          ; BRANCH IF THE GOOD AND THE BAD HAS BEEN RECORDED
4142 074230 005237 050256      INC     NIRCNT        ; ELSE, BUMP COUNT SO WE DON T MISS GOOD PACKET
4143 074234              375$:
4144 074234 000433              BR      50$          ; THROW THE PACKET OUT
4145 074236              38$:
4146 074236 026337 000014 050306      CMP     PROTOT(R3),PROTO0 ; SEE IF IT IS A LOOP SERVER TYPE CODE
4147 074244 001023              BNE     40$          ; IF NOT, CONT.
4148 074246 062703 000016      ADD     @FASKIP,R3    ; POINT TO SKIP COUNT
4149 074252 062303              ADD     (R3)+,R3      ; ADD SKIP COUNT TO DATA POINTER
4150 074254 022713 000002      CMP     @FORWARD,(R3) ; NEED TO FORWARD MESSAGE?
4151 074260 001004              BNE     45$          ; IF NOT, MOVE ON
4152 074262 005737 003202      TST     RSPFLG        ; ELSE, ARE WE ACTING AS A RESPONDER?
4153 074266 001416              BEQ     50$          ; IF NOT, JUST THROW IT OUT
4154 074270 000444              BR      60$          ; ELSE, GIVE IT TO THE CALLING ROUTINE
4155 074272              45$:
4156 074272 022713 000001      CMP     @REPLY,(R3)   ; ELSE, IS THIS A REPLY MESSAGE
4157 074276 001441              BEQ     60$          ; BRANCH IF YES
4158 074300              ERRHRD 52,MSG52    ; ELSE REPORT ERROR
      074300 104456              TRAP   C$ERRHRD
      074302 000064              .WORD 52
      074304 065633              .WORD MSG52
      074306 000000              .WORD 0
4159 074310              INC     R0            ; PUT FAILURE STATUS INTO R0
4160 074312 000545              BR      180$         ; LEAVE
4161 074314              40$:
4162 074314 026337 000014 050310      CMP     PROTOT(R3),PROTO2 ; ELSE CHECK FOR CONSOLE TYPE CODE
4163 074322 001427              BEQ     60$          ; IF OK, CONT.
4164 074324              50$:
4165 074324 012764 000001 000002      MOV     @1,DESC(R4)   ; MARK IT AS A THROW AWAY
4166 074332              CALL   GETRNX @0,@RRGCR ; AND THROW IT OUT
      074332 012725 024222              MOV     @RRGCR,(R5)+
      074336 012725 000000              MOV     @0,(R5)+
      074342 004437 070656              JSR     R4,PREG14
      074346 006450              .WORD  GETRNX-ANCHOR
4167 074350 020437 024226      CMP     R4,RRGNXT    ; IS THE CURRENT POINTER SAME AS NEXT?
4168 074354 001007              BNE     55$          ; BRANCH IF NOT
4169 074356              CALL   GETRNX @1,@RRGNXT ; ELSE, UPDATE NEXT POINTER AS WELL
      074356 012725 024226              MOV     @RRGNXT,(R5)+
      074362 012725 000001              MOV     @1,(R5)+
      074366 004437 070656              JSR     R4,PREG14
      074372 006450              .WORD  GETRNX ANCHOR
4170 074374              55$:
4171 074374 005337 050256      DEC     NIRCNT        ; DECREMENT RECEIVE COUNTER
4172 074400 000510              BR      170$         ; EXIT
4173 074402              60$:
4174 074402 032701 040000      BIT     @MRERRS,R1    ; SEE IF ANY ERRORS
4175 074406 001462              BEQ     140$         ; BRANCH IF NOT

```



```

4176 074410 032701 034007      BIT      #MRSETP!MDISC!MFRAM!MCRC!MOVF!MRUNT,R1 ; WAS THE PACKET TOO LARGE
4177 074414 001005      BNE      80$ ; BRANCH IF NOT
4178 074416      ERRHRD  53,MSG53 ; REPORT THE ERROR
      074416 104456      TRAP    C$ERHRD
      074420 000065      .WORD  53
      074422 065721      .WORD  MSG53
      074424 000000      .WORD  0
4179 074426 000446      BR       130$ ; CONTINUE
4180
4181 074430      80$:
4182 074430 032701 010000      BIT      #MDISC,R1 ; ARE THERE ANY OTHER VALID ERRORS?
4183 074434 001447      BEQ     140$ ; BRANCH IF NOT
4184 074436 032701 000001      BIT      #MOVF,R1 ; OVERFLOW?
4185 074442 001405      BEQ     90$ ; BRANCH IF NOT
4186 074444      ERRHRD  55,MSG55 ; REPORT OVERFLOW ERROR
      074444 104456      TRAP    C$ERHRD
      074446 000067      .WORD  55
      074450 066063      .WORD  MSG55
      074452 000000      .WORD  0
4187 074454 000433      BR       130$ ; BRANCH
4188 074456      90$:
4189 074456 032701 000002      BIT      #MCRC,R1 ; CRC ERROR?
4190 074462 001414      BEQ     110$ ; BRANCH IF NOT
4191 074464      ERRHRD  56,MSG56 ; ELSE REPORT IT
      074464 104456      TRAP    C$ERHRD
      074466 000070      .WORD  56
      074470 066116      .WORD  MSG56
      074472 000000      .WORD  0
4192 074474      100$:
4193 074474 032701 000004      BIT      #MFRAM,R1 ; WAS FRAMING ERROR ALSO SET?
4194 074500 001405      BEQ     110$ ; BRANCH IF NOT
4195 074502      ERRHRD  57,MSG57 ; REPORT THE ERROR
      074502 104456      TRAP    C$ERHRD
      074504 000071      .WORD  57
      074506 066152      .WORD  MSG57
      074510 000000      .WORD  0
4196 074512 000414      BR       130$ ; CONTINUE
4197 074514      110$:
4198 074514 032701 000010      BIT      #MSHORT,R1 ; PACKET LESS THAN 10 BYTES?
4199 074520 001405      BEQ     120$ ; BRANCH IF NOT
4200 074522      ERRHRD  60,MSG60 ; REPORT IT
      074522 104456      TRAP    C$ERHRD
      074524 000074      .WORD  60
      074526 066344      .WORD  MSG60
      074530 000000      .WORD  0
4201 074532 000404      BR       130$
4202 074534      120$:
4203 074534      ERRHRD  61,MSG61 ; REPORT UNKNOWN RECEIVER ERROR
      074534 104456      TRAP    C$ERHRD
      074536 000075      .WORD  61
      074540 066412      .WORD  MSG61
      074542 000000      .WORD  0
4204 074544      130$:
4205 074544 005237 050276      INC     RCVERR ; ELSE, INCREMENT RECEIVE ERROR COUNTER
4206 074550 005200      INC     RO ; SET STATUS TO FAILURE
4207 074552 000664      BR      50$ ; UPDATE POINTERS AND RETURN
4208 074554      140$:

```

```

4209 074554 005237 050300          INC    RCVBUF          ; INCREMENT GOOD BUFFERS RECEIVED COUNTER
4210 074560          150$:          INC    R2              ; KEEP COUNT OF HOW MANY BUFFERS RECEIVED
4211 074560 005202          DEC    NIRCNT          ; KEEP BOOKKEEPING IN ORDER
4212 074562 005337 050256          CALL   GETRX  #0,#RRGCR ; UPDATE "RECEIVE RING CURRENT" POINTER
4213 074566          MOV    #RRGCR,(R5)+
      074566 012725 024222          MOV    #0,(R5)+
      074572 012725 000000          JSR   R4,PREG14
      074576 004437 070656          .WORD GETRX-ANCHOR
      074602 006450
4214 074604 000406          BR    170$
4215 074606          160$:
4216 074606 012700 000002          MOV   #2,R0           ; SET FAILURE STATUS IN R0
4217 074612          ERRDF 11,EMSG11,ERR1 ; RECEIVE RING BOOKKEEPING ERROR
      074612 104455          TRAP C$ERDF
      074614 000013          .WORD 11
      074616 063735          .WORD EMSG11
      074620 070306          .WORD ERR1
4218 074622          170$:
4219 074622 005064 000002          CLR   DESC(R4)       ; INVALIDATE CURRENT DESCRIPTOR
4220 074626          180$:
4221 074626          RETURN R2           ; RETURN WITH NUMBER OF ENTRYS HANDLED
      074626 010225          MOV   R2,(R5)+
      074630 000207          RTS    PC
4222

```

4224  
4225  
4226  
4227  
4228  
4229  
4230  
4231  
4232  
4233  
4234  
4235  
4236  
4237  
4238  
4239  
4240  
4241  
4242  
4243  
4244  
4245  
4246  
4247  
4248  
4249  
4250  
4251  
4252  
4253  
4254  
4255  
4256  
4257  
4258  
4259  
4260  
4261  
4262  
4263  
4264  
4265  
4266  
4267  
4268  
4269  
4270  
4271  
4272  
4273  
4274  
4275  
4276  
4277  
4278

074632  
074632 012701 050144  
074636  
074636 014502  
074640  
074640 014503  
074642 001504  
074644 016204 000010  
074650 032704 000400

```

.SBTTL  BMPCNT  UPDATE STATISTICS
; **
; FUNCTIONAL DESCRIPTION:
;
; THIS ROUTINE IS USED TO UPDATE THE STATISTICS COUNTERS FROM THE
; INFO IN STATUS WORDS 1 AND 2 IN BUFFER DESCRIP.OR ENTRIES
;
; CALLING SEQUENCE
;
; CALL  BMPCNT  XFER TYPE,DESCRIPTOR ADDRESS
;
; INPUT
;
; DESCRIPTOR ADDRESS  ADDRESS OF THE RING DESCRIPTOR ENTRY
; XFER TYPE - 0 = RING DESCRIPTOR FROM RCV LIST
;           - 1 = RING DESCRIPTOR FROM XMIT LIST
;
; IMPLICIT INPUT
;
; NONE
;
; OUTPUT
;
; COUNTER ARE UPDATED
;
; IMPLICIT OUTPUT
;
; NONE
;
; COMPLETION CODES
;
; NONE
;
; SIDE EFFECTS
;
; NONE
;
; REGISTERS USED
;
; R1 - HOLDS ADDRESS OF COUNTER TABLE
; R2 - HOLDS ADDRESS OF RING DESCRIPTOR
; R3 - SCRATCH PAD
; R4 - SCRATCH PAD
;
; DEBUG
;
; NONE
;
BMPCNT:  MOV     #CNTRS,R1      ; GET ADDRESS OF COUNTER TABLE
        P$POP  R2          ; GET ADDRESS OF DESCRIPTOR
        MOV     -(R5),R2   ;
        P$POP  R3          ; XMIT OR RECEIVE?
        MOV     -(R5),R3   ;
        BEQ    BMPREC      ; BRANCH IF RECEIVE PACKET
        MOV    STAT1(R2),R4 ; GET STATUS WORD
        BIT    #MFAIL,R4   ; DID WE GET HEARTBEAT FAILURE?

```

```

4279 074654 001402          BEQ      3#          ; BRANCH IF NOT
4280 074656 005261 000074    INC      C.COLL(R1) ; ELSE, INCREMENT NUMBER OF COLLISION TEST FAILURES
4281                               ;
4282                               ; SEE IF XMISSION WAS ABORTED, IF IT WAS, SET REASON MASK, NOTE THAT 16
4283                               ; RETRIES WILL HAVE BEEN ATTEMPTED FOR CARRIER LOSS AND NO CARRIER
4284                               ;
4285 074662          3#:
4286 074662 032704 016000    BIT      #MABORT!MLOSS!MNOCAR,R4 ; ABORT ?
4287 074666 001453          BEQ      50#          ; BRANCH IF NOT
4288 074670 005261 000072    INC      C.XABT(R1)   ; BUMP ABORT COUNT
4289 074674 032704 004000    BIT      #MABORT,R4  ; IS ABORT SET?
4290 074700 001410          BEQ      10#         ; BRANCH IF NOT
4291 074702          PUSH     R4          ; SAVE R4
                                MOV      R4,(SP)
4292 074704 042704 177417    BIC      #177417,R4  ; CLEAR EVERYTHING BUT RETRY COUNT
4293 074710 001003          BNE      5#          ; BRANCH IF THE COUNT IS NOT ZERO
4294 074712 052761 000001 000070  BIS      #C.RTRY,C.XABB(R1) ; ELSE, SET RETRY FAILURE BIT IN BIT MASK
4295 074720          5#:
4296 074720          POP      R4          ; RESTORE CONTENTS OF R4
                                MOV      (SP),R4
4297 074722          10#:
4298 074722 032704 010000    BIT      #MLOSS,R4   ; WAS LOST CARRIER REASON FOR ABORT?
4299 074726 001406          BEQ      20#         ; BRANCH IF NOT
4300 074730 032704 002000    BIT      #MNOCAR,R4  ; TEST FOR NO CARRIER SENSED
4301 074734 001003          BNE      20#         ; BRANCH IF ZERO
4302 074736 052761 000002 000070  BIS      #C.CAR,C.XABB(R1) ; SET CARRIER LOST BIT
4303 074744          20#:
4304 074744 032704 010000    BIT      #MLOSS,R4   ; WAS LOST CARRIER REASON FOR ABORT?
4305 074750 001006          BNE      30#         ; BRANCH IF NOT
4306 074752 032704 002000    BIT      #MNOCAR,R4  ; TEST FOR NO CARRIER SENSED
4307 074756 001403          BEQ      30#         ; BRANCH IF ZERO
4308 074760 032761 000004 000070  BIT      #C.SHRT,C.XABB(R1) ; YES, IT'S SET, SET SHORTED BIT
4309 074766          30#:
4310 074766 032761 000001 000070  BIT      #C.RTRY,C.XABB(R1) ; RETRY ABORT SET?
4311 074774 001407          BEQ      40#         ; BRANCH IF NOT
4312 074776 032761 000004 000070  BIT      #C.SHRT,C.XABB(R1) ; ELSE, IS THE SHORT BIT SET?
4313 075004 001003          BNE      40#         ; BRANCH IF NOT
4314 075006 052761 000010 000070  BIS      #C.OPEN,C.XABB(R1) ; ELSE, RETRY ABORT AND NOT A SHORT, SET OPEN BIT
4315 075014          40#:
4316 075014 000560          BR       BMPEXI      ; NO XMISSION OCCURRED, IGNORE BYTCNT
4317 075016          50#:
4318 075016 042704 177417    BIC      #177417,R4  ; CLEAR ALL BUT RETRY COUNT
4319 075022 001460          BEQ      BMPMUL      ; COUNTER IS ZERO, NO RETRIES OCCURRED
4320 075024 162704 000020    SUB      #20,R4      ; DECREMENT COUNT
4321 075030 001455          BEQ      BMPMUL      ; BRANCH IF ONLY ONE RETRY
4322 075032 162704 000020    SUB      #20,R4      ; ELSE, DECREMENT COUNT AGAIN
4323 075036 001003          BNE      60#         ; BRANCH IF MORE THAN TWO
4324 075040 005261 000050    INC      C.PXM2(R1)  ; ELSE, BUMP 2 RETRIES COUNT
4325 075044 000447          BR       BMPMUL      ; FINISH COUNTERS
4326 075046          60#:
4327 075046 005261 000044    INC      C.PXM3(R1)  ; INCREMENT 3 OR MORE RETRIES COUNTER
4328 075052 000444          BR       BMPMUL      ; FINISH COUNTERS
4329 075054          BMPREC:
4330 075054 016204 000010    MOV      STAT1(R2),R4 ; GET STATUS WORD 1
4331 075060 032704 040000    BIT      #MREERRS,R4 ; TEST ERROR BIT
4332 075064 001001          BNE      5#          ; BRANCH IF ERROR BIT IS SET
4333 075066 000436          BR       BMPMUL
    
```

BMPCNT UPDATE STATISTICS

```

4334 075070          5%:
4335 075070 032704 010000      BIT    #MDISC,R4      ; IS RUNT OR DISCARD SET?
4336 075074 001424          BEQ    40%           ; BRANCH IF NOT
4337 075076          10%:
4338 075076 005261 000016      INC    C.RERR(R1)    ; INCREMENT RECEIVE ERROR COUNT
4339 075102 032704 000001      BIT    #MOVF,R4      ; OVERFLOW?
4340 075106 001402          BEQ    20%           ; BRANCH IF NOT
4341 075110 005261 000030      INC    C.RLIN(R1)    ; SET INTERNAL BUFFER ERROR
4342 075114          20%:
4343 075114 032704 000002      BIT    #MCRC,R4      ; CRC ERROR SET?
4344 075120 001411          BEQ    35%           ; BRANCH IF NOT
4345 075122 052761 000001 000014  BIS    #C.CRC,C.RERB(R1) ; SET CRC ERROR BIT
4346 075130          30%:
4347 075130 032704 000004      BIT    #MFRAM,R4     ; FRAMING ERROR?
4348 075134 001403          BEQ    35%           ; BRANCH IF NOT
4349 075136 052761 000002 000014  BIS    #C.FRAME,C.RERB(R1) ; ELSE, SET FRAMING ERROR BIT
4350 075144          35%:
4351 075144 000504          BR     BMPEXI        ; EXIT
4352 075146          40%:
4353 075146 032704 004000      BIT    #MRUNT,R4     ; RUNT PACKET?
4354 075152 001404          BEQ    BMPMUL        ; BRANCH IF NOT
4355 075154 052761 000010 000014  BIS    #C.RUNT,C.RERB(R1) ; SET RUNT PACKET BIT
4356 075162 000475          BR     BMPEXI        ; EXIT
4357 075164          BMPMUL:
4358 075164 016225 000004      P#PUSH LOADD(R2)    ; SAVE ADDRESS OF DATA BUFFER
4359 075170 005703          TST    R3            ; XMIT OR RECEIVE?
4360 075172 001405          BEQ    5%           ; BRANCH IF RECEIVE
4361 075174 016204 000006      MOV    WRDCNT(R2),R4 ; GET XMITTED WORD COUNT
4362 075200 005404          NEG    R4            ; COMPLEMENT IT
4363 075202 006304          ASL   R4            ; TURN INTO BYTE COUNT
4364 075204 000413          BR     15%         ; CONTINUE
4365 075206          5%:
4366 075206 042704 174377      BIC    #174377,R4    ; EXTRACT RCV BYTE LENGTH 10:8
4367 075212          P#PUSH R2            ; SAVE THE PACKET ADDRESS
4368 075214 016202 000012      MOV    STAT2(R2),R2 ; GET RCV BYTE LENGTH 7:0
4369 075220 042702 177400      BIC    #1C377,R2    ; CLEAR COPY OF THE COUNT IN HIGH BYTE
4370 075224 050204          BIS    R2,R4        ; MERGE THE TWO BYTE LENGTHS
4371 075226 062704 000074      ADD    #60.,R4      ; MAKE UP FOR BYTES LOST DURING ADDRESS CHECK
4372 075232          P#POP  R2            ; RESTORE THE PACKET ADDRESS
4373 075234          15%:
4374 075234 162704 000016      SUB    #16,R4       ; SUBTRACT PROTOCOL BYTES, COUNT ONLY DATA BYTES
4375 075240 005703          TST    R3            ; SEE IF XMIT OR RECEIVE
4376 075242 001430          BEQ    50%         ; BRANCH IF RECEIVE
4377 075244 005261 000034      INC    C.PXMT(R1)    ; INCREMENT NUMBER OF PACKETS XMITTED
4378 075250 032762 000100 000002  BIT    #MHIBYT,DESC(R2) ; DID DATA START ON BYTE BOUNDARY?
4379 075256 001401          BEQ    10%         ; BRANCH IF NOT
4380 075260 005304          DEC    R4            ; DECREMENT COUNT BY ONE
4381 075262          10%:
4382 075262 032762 000200 000002  BIT    #MLOBYT,DESC(R2) ; DID DATA START ON BYTE BOUNDARY?
4383 075270 001401          BEQ    20%         ; BRANCH IF NOT
4384 075272 005304          DEC    R4            ; DECREMENT COUNT BY ONE
4385 075274          20%:
4386 075274          P#POP  R2            ; GET DATA BUFFER ADDRESS
4386 075274 014502          MOV    (R5),R2

```

BMP CNT UPDATE STATISTICS

4387	075276	032762	000001	000000	BIT	#BIT00,DESTIN(R2)	:	MULTICAST ADDRESS?
4388	075304	001404			BEQ	40:	:	NO IF ZERO, MOVE ON
4389	075306	005261	000040		INC	C.MXMT(R1)	:	BUMP MULTICAST XMIT COUNT
4390	075312	060461	000064		ADD	R4,C.XMDB(R1)	:	ADD BYTE COUNT TO MULTICAST BYTE CNT
4391	075316							
4392	075316	060461	000060		ADD	R4,C.XDAT(R1)	:	ADD BYTE COUNT TO TOTAL DATA
4393	075322	000415			BR	BMPEXI	:	EXIT
4394	075324							
4395	075324				P#POP	R2	:	GET DATA BUFFER ADDRESS
	075324	014502						MOV -(R5),R2
4396	075326	005261	000004		INC	C.PREC(R1)	:	BUMP PACKETS RECEIVED
4397	075332	032762	000001	000000	BIT	#BIT00,DESTIN(R2)	:	MULTICAST ADDRESS?
4398	075340	001404			BEQ	60:	:	BRANCH NOT SET
4399	075342	005261	000010		INC	C.MREC(R1)	:	BUMP MULTICAST RECEIVE PACKET COUNT
4400	075346	060461	000024		ADD	R4,C.RMDB(R1)	:	BUMP MULTICAST RECEIVE BYTE COUNT
4401	075352							
4402	075352	060461	000020		ADD	R4,C.RDAT(R1)	:	BUMP RECEIVED DATA COUNT
4403	075356				BMPEXI:			
4404	075356				RETURN		:	ADIOS, AVOIR, AUMEIDERSEN
	075356	000207						RTS PC

.SBTTL EDPACK ETHERNET DATA PACKING ROUTINE

4406  
4407  
4408  
4409  
4410  
4411  
4412  
4413  
4414  
4415  
4416  
4417  
4418  
4419  
4420  
4421  
4422  
4423  
4424  
4425  
4426  
4427  
4428  
4429  
4430  
4431  
4432  
4433  
4434  
4435  
4436  
4437  
4438  
4439  
4440  
4441

\*\*\*  
: FUNCTIONAL DESCRIPTION:

: EDPACK ETHERNET DATA PACK

: THIS ROUTINE WILL CONVERT A STRING OF HEX CHARACTERS INTO A RIGHT  
: JUSTIFIED BINARY STREAM (WITH LEADING ZEROS), COMPATIBLE WITH  
: ETHERNET CONVENTIONS. THE SOURCE STRING MUST BE FOPMATTED USING  
: EITHER A WORD BY WORD HEX DESCRIPTION OR A BYTE BY BYTE HEX  
: DESCRIPTION. THE RETURNED STRING WILL BE BYTE ORIENTED AS  
: REQUIRED BY THE ETHERNET:  
: LO-BYTE-WORD0 HI-BYTE-WORD0 LO-BYTE-WORD1 HI-BYTE-WORD1, ETC.

: INPUT ARGUMENTS P1 - ADDRESS OF THE SOURCE (HEX) STRING  
: TO BE CONVERTED TO A BINARY STREAM.  
: P2 ADDRESS OF THE DESIRED DESTINATION  
: BUFFER WHICH WILL ACCEPT BINARY DATA  
: P3 LENGTH (IN BYTES) OF THE DESTINATION  
: BUFFER  
: EXPLICIT OUTPUTS P4 - ZERO IF SUCCESSFUL, -1 IF BUFFER TOO  
: LONG OR ODD NUMBER OF HEX CHARACTERS  
: IMPLIED OUTPUTS THE BUFFER AT P2 WILL CONTAIN A RIGHT  
: JUSTIFIED BINARY STREAM W/ LEADING ZEROS  
: AND CORRESPONDING TO HEX STRING AT R5.  
: SUBORDINATE PROCEDURES HXFORM. (STRIP NONHEX CHARACTERS)  
: HEXBIN. (HEX TO BINARY CONVERSION)  
: CALLING PROCEDURE CALL EDPACK P1,P2,P3 ;INPUT P1-P3 PARAMETERS  
: P;POP P4 ;OUTPUT P4 PARAMETER

4442 075360  
4443 075472 000000  
4444 075474  
075474 014537 075472  
075500 014504  
075502 014503  
4445  
4446 075504 005002  
4447 075506 006303  
4448 075510  
075510 010325  
075512 012725 075360  
075516 013725 075472  
075522 004437 070656  
075526 004672  
4449 075530  
075530 014501  
075532 014502  
4450  
4451 075534 005702  
4452 075536 001010

:  
: -  
: LOCDST: .BLKB 74. ;MAX NUMBER OF CHARACTERS THAT MAY BE ENTERED  
: SOURCE: .WORD ;SOURCE ADDRESS  
: EDPACK: P;POP SOURCE,R4,R3 ;R4-DESTINATION, R3-NUMBER OF CHARS REQ  
: MOV (R5),SOURCE  
: MOV -(R5),R4  
: MOV -(R5),R3  
: ;SOURCE-SRC ADDRESS, ORIENT-WORD/BYTE?  
: ;ASSUME NO ERRORS, VALUE RETURNED  
: ;NUMBER OF CHARACTERS REQUIRED W/ '0'S  
: CLR R2  
: ASL R3  
: CALL HXFORM SOURCE,#LOCDST,R3  
: MOV R3,(R5)+  
: MOV #LOCDST,(R5)+  
: MOV SOURCE,(R5)+  
: JSR R4,PREG14  
: .WORD HXFORM ANCHOR  
: P;POP R1,R2 ;R1-ADDRESS OF LAST CHAR  
: MOV -(R5),R1  
: MOV -(R5),R2  
: ;R2=SUCCESS/FAIL CODE (0/ 1)  
: ;R1 WILL POINT TO RIGHTMOST CHARACTER  
: ;RIGHT JUSTIFY BUFFER  
: TST R2  
: BNE 10;

```

4453
4454 075540 006203          ASR      R3
4455 075542          CALL     HEXBIN @LOCST,R4,R3
      075542 010325
      075544 010425
      075546 012725 075360
      075552 004437 070656
      075556 005030
4456
4457 075560          10$:    RETURN R2
      075560 010225
      075562 000207
4458
;CONVERT HEX AT LOCST TO BINARY
;R3 BYTES IN OUTPUT BIT STREAM
      MOV     R3,(R5)+
      MOV     R4,(R5)+
      MOV     @LOCST,(R5)+
      JSR     R4,PREG14
      .WORD   HEXBIN ANCHOR
;RETURN WITH SUCCESS/FAILURE INDICATION
      MOV     R2,(R5)+
      RTS     PC

```



```

4460 .SBTTL HXFORM - HEX FORMAT ROUTINE
4461 ;**
4462 ; FUNCTIONAL DESCRIPTION:
4463 ;
4464 ; HXFORM HEX FORMAT ROUTINE
4465 ;
4466 ; THIS ROUTINE WILL ACCEPT A HEX STRING, AND STRIP OUT THE NON-HEX
4467 ; CHARACTERS (WITHOUT FLAGGING AN ERROR FOR THE NON-HEX CHARACTERS).
4468 ; A NULL WILL BE LEFT AS THE END OF STRING DELIMITER.
4469 ;
4470 ; INPUT ARGUMENTS P1 ADDRESS OF THE SOURCE ASCII STRING (NULL
4471 ; DELIMITER AT END OF STRING)
4472 ; P2 ADDRESS OF THE DESTINATION ASCII HEX STRING
4473 ; (STRIPPED OF NON-HEX AND RIGHT JUSTIFIED)
4474 ; P3- THE NUMBER OF HEX CHARACTERS DESIRED @R4
4475 ; EXPLICIT OUPUTS P4- WILL CONTAIN THE ADDRESS OF THE LAST
4476 ; CHARACTER IN THE OUTPUT BUFFER.
4477 ; P5- THE SUCCESS/FAILURE (0/-1) INDICATOR
4478 ;
4479 ; IMPLICIT OUTPUTS THE BUFFER AT R4 WILL CONTAIN FORMATTED
4480 ; HEX (ASCII) STRING.
4481 ; CALLING PROCEDURE CALL HXFORM P1,P2,P3
4482 ; P$POP P4,P5
4483 ;
4484 075564 000000 HXN: .WORD
4485 075566 014503 HXFORM: P$POP R3,R2,HXN ;ADDRESS OF SOURCE STRING
075570 014502 MOV -(R5),R3
075572 014537 075564 MOV (R5),P2
MOV (R5),HXN
4486 ;ADDRESS OF DESTINATION STRING
4487 ;NUMBER OF HEX CHARACTERS DESIRED
4488
4489 075576 010204 MOV R2,R4 ;DESTINATION ADDRESS, R2: DESTINATION POINTER
4490 075600 063704 075564 ADD HXN,R4 ;POINT TO END OF OUTPUT BUFFER (DESTINATION)
4491 ;DO WHILE NO NULL FOUND IN SOURCE STRING
4492 075604 112301 10$: MOVB (R3),R1 ;GET BYTE IN QUESTION (& POINT TO NEXT BYTE)
4493 075606 142701 000200 20$: BICB @200,R1 ;ENSURE HI BIT IS LO (SEVEN BIT ASCII)
4494 ;CHECK FOR VALID HEX CHARACTER
4495 075612 001414 BEQ 30$ ;THE NULL IS A VALID CHARACTER (BUT THE LAST)
4496 075614 120127 000060 CMPB R1,#60 ;IF GREATER THAN "0" THEN RANGE OK
4497 075620 100413 BMI 40$ ;40$-OUT OF RANGE, BYTE VALUE TOO SMALL
4498 075622 120127 000072 CMPB R1,#72 ;IF BYTE IS LESS THAN 72 AND >=60 THEN RANGE OK
4499 075626 100406 BMI 30$ ;RANGE OK IF >=60 AND <72 ELSE, CONTINUE CHECK
4500 075630 120127 000101 CMPB R1,#101 ;BYTE MUST BE >=101 TO CONTINUE CHECK
4501 075634 100405 BMI 40$ ;IF BYTE >71 AND <101 THEN BYTE OUT OF RANGE.
4502 075636 122701 000106 CMPB @106,R1 ;BYTE MUST BE <=106 TO BE OK, ELSE: NOT OK.
4503 075642 100402 BMI 40$ ;BYTE NOT OK, DON'T PLACE THIS BYTE IN OUTPUT.
4504 ;PLACE THE HEX BYTE IN THE OUTPUT BUFFER.
4505 075644 110122 30$: MOVB R1,(R2) ;BYTE IN RANGE, CONFIRMED. BYTE OK. POINT TO
4506 075646 001405 BEQ 60$ ;NEXT BYTE DEST ADDRESS. IF NULL, THEN EXIT.
4507 ;(NO ERRORS)
4508 ;IF NOT NULL, THEN CHECK FOR BUFFER OVERFLOW.
4509 ;R4 POINTS TO LAST CHARACTER POSITION (OUTPUT)
4510 075650 020402 40$: CMP R4,R2 ;R2 - PRESENT WRITE ADDRESS
4511 075652 100354 BPL 10$ ;(SHOULD BE POSITIVE RESULT OR 0) (MORE TO DO)
4512 075654 012704 177777 50$: MOV @-1,R4 ;SET ERROR CONDITION (EXIT WITH ERROR)
4513 075660 000404 BR 70$ ;ERROR DETECTED EXIT PATH -> (TOO MANY CHARS)

```

4514  
4515  
4516 075662 005302  
4517 075664 020402  
4518 075666 001372  
4519 075670 005004  
4520 075672  
075672 010425  
075674 010225  
075676 000207  
4521

60\$: DEC R2  
CMP R4,R2  
BNE 50\$  
CLR R4  
70\$: RETURN R2,R4

;SUCCESSFUL EXIT PATH  
;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  
MOV R4,(R5).  
MOV R2,(R5).  
RTS PC  
;ERROR INDICATOR (R4) IS P5

```

4523 .SBTTL HEXBIN HEX TO BINARY CONVERSION
4524 :
4525 : HEXBIN HEX TO BINARY CONVERSION PROCEDURE
4526 :
4527 : THIS PROCEDURE WILL CONVERT A STRING OF HEX (ASCII) CHARACTERS
4528 : DIRECTLY TO A BINARY STREAM. THE DESTINATION BINARY STREAM WILL
4529 : REQUIRE ONLY HALF AS MANY BYTES AS THE HEX STRING BECAUSE TWO
4530 : HEX CHARACTERS ARE REQUIRED TO REPRESENT A SINGLE BINARY BYTE.
4531 :
4532 : INPUTS P1 SOURCE STRING ADDRESS (DELIMITED BY A NULL)
4533 : P2 DESTINATION ADDRESS FOR THE BINARY DATA.
4534 : P3 THE NUMBER OF BINARY BYTES REQUIRED (HALF THE
4535 : NUMBER OF CHARACTERS AT P1.
4536 : OUTPUTS NO EXPLICIT OUTPUTS
4537 : IMPLIED OUTPUTS THE BUFFER AT P2 WILL CONTAIN THE BINARY
4538 : STREAM, CONVERTED DIRECTLY FROM THE BUFFER
4539 : AT P1.
4540 : SUBORDINATE PROCEDURES NONE
4541 : CALLING PROCEDURE CALL HEXBIN P1,P2,P3
4542 :--
4543 075700 000000 MN: .WORD
4544 075702 060 061 062 CMPSTR: .ASCIZ /0123456789ABCDEF/
075705 063 064 065
075710 066 067 070
075713 071 101 102
075716 103 104 105
075721 106 000

4545 .EVEN
4546 075724 HEXBIN: P$POP R1,R2,MN ;R1=SOURCE STRING ADDRESS
075724 014501 MOV (R5),R1
075726 014502 MOV (R5),R2
075730 014537 075700 MOV -(R5),MN
;R2=DESTINATION STRING ADDRESS
;MN=NUMBER OF BYTES REQUIRED
;MN NOW POINTS TO THE LAST BYTE POSITION+1
;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

4547
4548
4549 075734 060237 075700 ADD R2,MN
4550 075740 012704 075702 10$: MOV @CMPSTR,R4
4551 075744 121124 20$: CMPB (R1),(R4)
4552 075746 001376 BNE 20$
4553 075750 005201 INC R1
4554 075752 162704 075703 SUB @CMPSTR+1,R4
4555
4556
4557 075756 006304 ASL R4
4558 075760 006304 ASL R4
4559 075762 006304 ASL R4
4560 075764 006304 ASL R4
4561 075766 010403 MOV R4,R3
4562 075770 012704 075702 30$: MOV @CMPSTR,R4
4563 075774 121124 CMPB (R1),(R4)
4564 075776 001376 BNE 30$
4565 076000 005201 INC R1
4566 076002 162704 075703 SUB @CMPSTR+1,R4
4567
4568

```

4569	076006	050403	
4570			
4571	076010	110322	
4572	076012	020237	075700
4573	076016	100750	
4574	076020		
	076020	000207	

```

BIS      R4,R3
MOVB    R3,(R2)+
CMP     R2,HN
BMI     10$
RETURN

```

```

;NOW THE TWO CHARACTERS HAVE MADE A SINGLE BYTE
;NOW PLACE THE COMPLETE BYTE IN THE DESTINATION
;AND POINT TO THE NEXT DESTINATION BYTE
;IF THE DESTINATION POINTER (R2) REACHES THE
;LAST CHARACTER POSITION+1 (HN) THEN DONE.
;RETURN TO CALLER

```

RTS PC

```

4576 .SBTTL          BINHEX - BINARY TO HEX CONVERSION PROCEDURE
4577          :---*
4578          :      BINHEX          BINARY TO HEX CONVERSION PROCEDURE
4579          :
4580          :      THIS PROCEDURE WILL CONVERT A BINARY DATA STREAM INTO A HEX STRING.
4581          :
4582          :      INPUTS  - P1- BINARY DATA BUFFER ADDRESS
4583          :                P2- NUMBER OF BYTES IN THE BUFFER
4584          :                P3- ADDRESS OF OUTPUT BUFFER FOR HEX STRING.
4585          :                HEX CHARACTER PAIRS SEPERATED BY "-"S
4586          :                (NOTE: THIS BUFFER MUST BE AT LEAST 3*P2 BYTES LONG)
4587          :      OUTPUTS - NONE
4588          :      IMPLICIT OUTPUTS  THE BUFFER AT P3 WILL CONTAIN THE HEX STRING
4589          :                        FOLLOWED BY A NULL CHARACTER.
4590          :      SUBORDINATE ROUTINES  NONE
4591          :      CALLING PROCEDURE     CALL BINHEX P1,P2,P3
4592          :---*
4593          HEXC:  .ASCII /0123456789ABCDEF/
4594          076022      060      061      062
4595          076025      063      064      065
4596          076030      066      067      070
4597          076033      071      101      102
4598          076036      103      104      105
4599          076041      106
4599 076042 000000          LST:  .WORD
4599 076044          BINHEX: P$POP  R1,LST,R2          ;R1 HAS THE INPUT BUFFER ADDRESS
4599 076044 014501          MOV          -(R5),R1
4599 076046 014537 076042  MOV          -(R5),LST
4599 076052 014502          MOV          -(R5),R2
4599          ;LST: HAS THE NUMBER OF BYTES IN INPUT BUFFER
4599          ;R2 HAS THE OUTPUT BUFFER ADDRESS
4599          ;LST IS NOW ADDRESS OF LAST SOURCE BYTE + 1
4599 10$:  ADD          R1,LST          ;GET THE CURRENT BYTE AND POINT TO NEXT BYTE
4599  MOVB          (R1)+,R3          ;SEPARATE NIBBLES AND GET CHARACTERS SEPARATELY
4599  MOVB          R3,R4          ;ONLY RIGHT BINARY NIBBLE REMAINS IN R3
4599  BIC          #177760,R3          ;SHIFT OVER FOR LEFT BINARY NIBBLE IN R4
4599  ASR          R4
4599  ASR          R4
4599  ASR          R4
4599  ASR          R4
4599  BIC          #177760,R4          ;ONLY LEFT BINARY NIBBLE REMAINS IN R4
4599          ;R4 IS THE MOST SIGNIFICANT NIBBLE (FIRST)
4599          ;R3 IS THE LEAST SIGNIFICANT NIBBLE (SECOND)
4599  MOVB          HEXC(R4),(R2)+          ;PUT THE ASCII BYTE INTO THE BUFFER HI POSITION
4599  MOVB          HEXC(R3),(R2)+          ;PUT THE ASCII BYTE INTO THE BUFFER LO POSITION
4599  MOVB          #'',(R2)+          ;PUT - BETWEEN HEX PAIRS
4599  CMP          R1,LST          ;RESULT IS NEGATIVE UNTIL R1=LST
4599  BLO          10$          ;UNTIL R1=LST. (TRANSFER ALL SOURCE BYTES)
4599  CLRB          (R2)          ;TERMINATE OUTPUT BUFFER WITH A NULL
4599  RETURN
4599          RTS          FC
4600 076104 116422 076022
4601 076110 116322 076022
4602 076114 112722 000055
4603 076120 020137 076042
4604 076124 103755
4605 076126 105042
4606 076130
4607 076130 000207

```

```

4617 .SBTTL BLDLD BUILD LOOP DIRECT DATA BUFFERS FOR TRANSMIT.
4618
4619 ; *
4620 ; FUNCTIONAL DESCRIPTION:
4621 ; THIS SUBROUTINE BUILDS LOOP DIRECT PACKETS FOR TRANSMISSION
4622 ; FROM THE QNA. SOURCE ADDRESS, DESTINATION ADDRESS,
4623 ; PROT. TYPE, AND LOOP DIRECT HEADER INFO ARE ADDED
4624 ; TO THE MESSAGE BUFFER. THE MESSAGE BUFFER IS BUILT
4625 ; BY A CALL TO BLDBUF.
4626 ;
4627 ; INPUTS - P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
4628 ; IMPLICIT - P#SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
4629 ; XRGNXT POINTS TO THE NEXT AVAILABLE RING ENTRY
4630 ; PHYADR HOLDS THE CURRENT LOCAL QNA PHYSICAL ADDRESS
4631 ;
4632 ; OUTPUTS - BUILT MESSAGE PACKET.
4633 ;
4634 ; CALLING PROCEDURE - CALL BLDLD P1
4635 ;
4636 ; SIDE EFFECTS - THE MESSAGE PACKET IS BUILT AND CONTAINED IN THE
4637 ; BUFFER POINTED TO BY XRGNXT WHEN THE ROUTINE WAS ENTERED.
4638 ; XRGNXT IS UPDATED TO POINT TO THE NEXT RING ENTRY
4639 ;
4640 ; REGISTER USAGE - R1 HOLDS ADDRESS OF DESTINATION ADDRESS
4641 ; R2 IS A POINTER FOR THE LOOP DIRECT HEADER INFO
4642 ; R3 HOLDS THE PACKET LENGTH
4643 ; R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
4644 ;
4645 ;---+
4645 076132 BLDLD::
4646 076132 P#POP R1 ; PUT ADDRESS OF DEST. ADDRESS IN R1
4647 076132 014501 MOV XRGNXT,R4 ; MOVE NEXT PACKET ADDRESS TO R4
4648 076134 013704 024224 MOV LOADD(R4),R4 ; POINT R4 TO DATA BLOCK
4649 076140 016404 000004 MOV P#SIZE,R3 ; PUT MESSAGE SIZE INTO R3
4650 076150 062703 000040 ADD #40,R3 ; ADD HEADER AND LOOP DIRECT INFO TO LENGTH
4651 076154 010337 050324 MOV R3,XFER ; PUT BYTES TRANSFERED INTO WORD
4652 076160 162737 000016 050324 SUB #16,XFER ; AND CORRECT FOR HEADER
4653 076166 022703 001100 CMP #XPKLEN,R3 ; SEE IF LONGER THAN ONE PACKET
4654 076172 002474 BLT 10# ; IF YES, ERROR
4655 076174 010337 050332 MOV R3,BUFLEN ; PUT PACKET LENGTH IN BUFLEN
4656 076200 012164 000000 MOV (R1)+,DESTIN(R4) ; MOVE FIRST TWO BYTES OF ADDRESS
4657 076204 012164 000002 MOV (R1)+,DESTIN+2(R4) ; MOVE BYTES THREE AND FOUR
4658 076210 011164 000004 MOV (R1),DESTIN+4(R4) ; MOVE BYTES FIVE AND SIX
4659 076214 005064 000006 CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
4660 076220 005064 000010 CLR SOURCC+2(R4) ; SIX BYTES WORTH
4661 076224 005064 000012 CLR SOURCC+4(R4)
4662 076230 013764 050306 000014 MOV PROTO,PROTOT(R4) ; MOVE PROTOCALL TYPE INTO HEADER
4663 076236 012702 050346 MOV #LOPDIR,R2 ; MOVE LOOPDIRECT FORMAT HEADER LOC. TO R2
4664 076242 012264 000016 MOV (R2)+,LDSKIP(R4) ; SKIP COUNT
4665 076246 011264 000020 MOV (R2),LDFCT1(R4) ; FUNCTION CODE (FORWARD)
4666 076252 013764 004010 000022 MOV PHYADR,LDADR1(R4) ; LOCAL NODE ADDRESS
4667 076260 013764 004012 000024 MOV PHYADR+2,LDADR1+2(R4) ; SIX BYTES
4668 076266 013764 004014 000026 MOV PHYADR+4,LDADR1+4(R4)
4669 076274 016264 000010 000030 MOV 10(R2),LDFCT2(R4) ; FUNCTION CODE (REPLY)
4670 076302 013764 004010 000032 MOV PHYADR,LDADR2(R4) ; LOCAL NODE ADDRESS
4671 076310 013764 004012 000034 MOV PHYADR+2,LDADR2+2(R4) ; SIX BYTES
4672 076316 013764 004014 000036 MOV PHYADR+4,LDADR2+4(R4) ;

```

4673	076324	062704	000040	ADD	@LDADR2+6,R4	;	POINT R4 TO FIRST DATA BYTE	
4674	076330	010437	050334	MOV	R4,CMPBUF	;	STORE BUFFER LOCATION FOR DATA COMPARE	
4675	076334			CALL	BLDBUF R4	;	BUILD DATA BUFFER	
	076334	010425						MOV R4,(R5)+
	076336	004437	070656					JSR R4,PREG14
	076342	006756						.WORD BLDBUF-ANCHOR
4676	076344			CALL	GETXNX #0,#XRGNXT	;	UPDATE POINTER TO NEXT RING ENTRY	
	076344	012725	024224					MOV #XRGNXT,(R5)+
	076350	012725	000000					MOV #0,(R5)+
	076354	004437	070656					JSR R4,PREG14
	076360	006534						.WORD GETXNX-ANCHOR
4677	076362	000404		BR	20\$	;	EXIT	
4678	076364			ERRDF	14,MSG14,ERR1	;	MESSAGE SIZE TOO BIG	
	076364	104455						TRAP C\$ERDF
	076366	000016						.WORD 14
	076370	063774						.WORD MSG14
	076372	070306						.WORD ERR1
4679	076374			20\$:	RETURN			
	076374	000207						RTS PC

```

4681 .SBTTL BLDFA$ - BUILD PACKET FOR FULL ASSIST TRANSMISSION
4682 ;
4683 ; FUNCTIONAL DESCRIPTION:
4684 ; THIS SUBROUTINE BUILDS FULL ASSIST PACKETS FOR TRANSMISSION
4685 ; FROM THE QNA. SOURCE ADDRESS, DESTINATION ADDRESS, PROT. TYPE
4686 ; AND FULL ASSIST HEADER INFO ARE ADDED TO THE MESSAGE BUFFER.
4687 ; THE MESSAGE BUFFER IS BUILT BY A CALL TO BLDBUF.
4688 ;
4689 ; INPUTS - P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
4690 ; IMPLICIT - P$SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
4691 ; XRGXNT POINTS TO THE NEXT AVAILABLE RING ENTRY
4692 ; PHYADR HOLDS THE CURRENT LOCAL NODE ADDRESS
4693 ;
4694 ; OUTPUTS THE BUILT BUFFER
4695 ;
4696 ; CALLING PROCEDURE - CALL BLDFA$ P1
4697 ;
4698 ; SIDE EFFECTS XRGXNT SI UPDATED TO POINT TO THE NEXT RING ENTRY
4699 ;
4700 ; REGISTER USAGE - R1 HOLDS ADDRESS OF TARGET NODE ADDRESS
4701 ; R2 HOLDS ADDRESS OF ASSIST NODE ADDRESS
4702 ; R3 HOLDS THE PACKET LENGTH
4703 ; R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
4704 ;
4705 ; -+
4706 BLDFA$: P$POP R1,R2 ; PUT ADDRESS OF TARGET ADDRESS INTO R1
4707 076376 MOV -(R5),R1
076376 014501 MOV -(R5),R2
076400 014502
4708 ; AND ADDRESS OF ASSIST ADDRESS INTO R2
4709 076402 013704 024224 MOV XRGXNT,R4 ; MOVE NEXT PACKET ADDRESS TO R4
4710 076406 016404 000004 MOVL0ADD(R4),R4 ; POINT R4 TO DATA BLOCK
4711 076412 013703 002374 MOV P$SIZE,R3 ; PUT MESSAGE SIZE INTO R3
4712 076416 062703 000060 ADD #60,R3 ; ADD HEADER INFO TO LENGTH
4713 076422 010337 050324 MOV R3,XFER ; PUT 'BYTES TRANSFERED' INTO WORD
4714 076426 162737 000016 050324 SUB #16,XFER ; AND CORRECT FOR HEADER
4715 076434 022703 001100 CMP #XPKLEN,R3 ; SEE IF LONGER THAN ONE PACKET
4716 076440 002524 BLT 10$ ; IF YES, ERROR
4717 076442 010337 050332 MOV R3,BUFLEN ; PUT PACKET LENGTH IN BUFLLEN
4718 076446 011264 000000 MOV (R2),DESTIN(R4) ; MOVE FIRST TWO BYTES OF ADDRESS
4719 076452 016264 000002 000002 MOV 2(R2),DESTIN+2(R4) ; MOVE BYTES THREE AND FOUR
4720 076460 016264 000004 000004 MOV 4(R2),DESTIN+4(R4) ; MOVE BYTES FIVE AND SIX
4721 076466 005064 000006 CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
4722 076472 005064 000010 CLR SOURCC+2(R4) ; SIX BYTES WORTH
4723 076476 005064 000012 CLR SOURCC+4(R4)
4724 076502 013764 050306 000014 MOV PROTOO,PROTOT(R4) ; MOVE PROTOCALL TYPE INTO HEADER
4725 076510 012764 000000 000016 MOV #0,FASKIP(R4) ; SKIP COUNT
4726 076516 012764 000002 000020 MOV #2,FAFCT1(R4) ; FUNCTION CODE (FORWARD)
4727 076524 011164 000022 MOV (R1),FAADR1(R4) ; TARGET NODE ADDRESS
4728 076530 016164 000002 000024 MOV 2(R1),FAADR1+2(R4) ; SIX BYTES
4729 076536 016164 000004 000026 MOV 4(R1),FAADR1+4(R4)
4730 076544 012764 000002 000030 MOV #2,FAFCT2(R4) ; FUNCTION CODE (FORWARD)
4731 076552 011264 000032 MOV (R2),FAADR2(R4) ; ASSIST NODE ADDRESS
4732 076556 016264 000002 000034 MOV 2(R2),FAADR2+2(R4) ; SIX BYTES
4733 076564 016264 000004 000036 MOV 4(R2),FAADR2+4(R4)
4734 076572 012764 000002 000040 MOV #2,FAFCT3(R4) ; FUNCTION CODE (FORWARD)
4735 076600 013764 004010 000042 MOV PHYADR,FAADR3(R4) ; LOCAL NODE ADDRESS
    
```



4736	076606	013764	004012	000044	MOV	PHYADR+2,FAADR3+2(R4)	:	SIX BYTES
4737	076614	013764	004014	000046	MOV	PHYADR+4,FAADR3+4(R4)	:	
4738	076622	012764	000001	000050	MOV	#1,FAFCT4(R4)	:	FUNCTION CODE (REPLY)
4739	076630	013764	004010	000052	MOV	PHYADR,FAADR4(R4)	:	LOCAL NODE ADDRESS
4740	076636	013764	004012	000054	MOV	PHYADR+2,FAADR4+2(R4)	:	SIX BYTES
4741	076644	013764	004014	000056	MOV	PHYADR+4,FAADR4+4(R4)	:	
4742	076652	062704	000060		ADD	#FAADR4+6,R4	:	POINT R4 TO FIRST DATA BYTE
4743	076656	010437	050334		MOV	R4,CMPBUF	:	STORE BUFFER LOCATION FOR DATA COMPARE
4744	076662				CALL	BLDBUF R4	:	BUILD DATA BUFFER
	076662	010435						MOV R4,(R5)
	076664	004437	070656					JSR R4,PREG14
	076670	006756						.WORD BLDBUF ANCHOR
4745	076672				CALL	GETXNX #0,#XRGXNT	:	UPDATE POINTER TO NEXT RING ENTRY
	076672	012725	024224					MOV #XRGXNT,(R5)
	076676	012725	000000					MOV #0,(R5)
	076702	004437	070656					JSR R4,PREG14
	076706	006534						.WORD GETXNX ANCHOR
4746	076710	000404			BR	20\$	:	EXIT
4747	076712			10\$:	ERRDF	29,EMSG14,ERR1	:	MESSAGE SIZE TOO BIG
	076712	104455						TRAP C1ERRDF
	076714	000035						.WORD 29
	076716	063774						.WORD EMSG14
	076720	070306						.WORD ERR1
4748	076722			20\$:	RETURN			
	076722	000207						RTS PC

```

      .SBTTL          BLDAST - BUILD TRANSMIT AND RECEIVE ASSIST PACKETS
4750
4751
4752      ; FUNCTIONAL DESCRIPTION:
4753
4754      ; THIS ROUTINE BUILDS RECEIVE ASSIST PACKETS
4755
4756      BLDAST::
4757      P#POP      R1,R2      ; PUT DESTINATION ADDRESS INTO R1
      MOV          -(R5),R1
      MOV          -(R5),R2
4758      ; ASSIST ADDRESS INTO R2
4759      MOV          XRG NXT,R4      ; MOVE NEXT PACKET ADDRESS TO R4
4760      MOV          LOADD(R4),R4      ; POINT R4 TO DATA BLOCK
4761      MOV          P#SIZE,R3      ; PUT MESSAGE SIZE INTO R3
4762      ADD          #50,R3      ; ADD HEADER INFO INTO LENGTH
4763      MOV          R3,XFER      ; PUT 'BYTES TRANSFERED INTO WORD
4764      SUB          #16,XFER      ; AND CORRECT FOR HEADER
4765      CMP          #XPKLEN,R3      ; SEE IF LONGER THAN ONE PACKET
4766      BLT          10$      ; IF YES, ERROR
4767      MOV          R3,BUFLEN      ; PUT PACKET LENGTH INTO BUFLen.
4768      MOV          (R1),DESTIN(R4)      ; MOVE DESTINATION ADDRESS INTO HEADER
4769      MOV          2(R1),DESTIN+2(R4)      ; SIX BYTES WORTH
4770      MOV          4(R1),DESTIN+4(R4)
4771      CLR          SOURCC(R4)      ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
4772      CLR          SOURCC+2(R4)      ; SIX BYTES WORTH
4773      CLR          SOURCC+4(R4)
4774      MOV          PROT00,PROT0T(R4)      ; MOVE PROTOCOL TYPE INTO HEADER
4775      MOV          #0,FASKIP(R4)      ; SKIP COUNT
4776      MOV          #2,FAFCT1(R4)      ; FUNCTION CODE (FORWARD)
4777      MOV          (R2),FAADR1(R4)      ; TARGET NODE ADDRESS
4778      MOV          2(R2),FAADR1+2(R4)      ; SIX BYTES
4779      MOV          4(R2),FAADR1+4(R4)
4780      MOV          #2,FAFCT2(R4)      ; FUNCTION CODE (FORWARD)
4781      MOV          PHYADR,FAADR2(R4)      ; LOCAL NODE ADDRESS
4782      MOV          PHYADR+2,FAADR2+2(R4)      ; SIX BYTES WORTH
4783      MOV          PHYADR+4,FAADR2+4(R4)
4784      MOV          #1,FAFCT3(R4)      ; FUNCTION CODE (REPLY)
4785      MOV          PHYADR,FAADR3(R4)      ; LOCAL NODE ADDRESS
4786      MOV          PHYADR+2,FAADR3+2(R4)
4787      MOV          PHYADR+4,FAADR3+4(R4)
4788      ADD          #FAADR3+6,R4      ; POINT R4 TO FIRST DATA BYTE
4789      MOV          R4,CMPBUF      ; STORE BUFFER LOCATION FOR DATA COMPARE
4790      CALL         BLDBUF R4      ; BUILD DATA BUFFER
      MOV          R4,(R5)+
      JSR          R4,PREG14
      .WORD       BLDBUF ANCHOR
477162      010425
477164      004437      070656
477170      006756

```

4791	077172		CALL	GETXNX	#0,#XRGXNT	; UPDATE RING POINTER		
	077172	012725					MOV	#XRGXNT,(R5);
	077176	012725					MOV	#0,(R5);
	077202	004437					JSR	R4,PREG14
	077206	006534					.WORD	GETXNX-ANCHOR
4792	077210	000404		BR	20;			
4793	077212		10;	ERRDF	36,MSG14,ERR1	; MESSAGE SIZE TOO BIG ERROR		
	077212	104455					TRAP	CERRDF
	077214	000044					.WORD	36
	077216	063774					.WORD	MSG14
	077220	070306					.WORD	ERR1
4794	077222		20;	RETURN				
	077222	000207					RTS	PC

4796  
4797  
4798  
4799  
4800  
4801  
4802  
4803  
4804  
4805  
4806  
4807  
4808  
4809  
4810  
4811  
4812  
4813  
4814  
4815  
4816  
4817  
4818  
4819  
4820  
4821  
4822  
4823  
4824  
4825  
4826  
4827  
4828  
4829  
4830  
4831  
4832  
4833  
4834  
4835  
4836

077224  
077224  
077230  
077234  
077242  
077246  
077252  
077256  
077262  
077266  
077272  
077276  
077304  
077310  
077314  
077320  
077324  
077324  
077330  
077334  
077340  
077342  
077342  
000207

```
.SBTTL          BLDREQ - BUILD REQUEST ID PACKETS FOR TRANSMIT
:--*
: FUNCTIONAL DESCRIPTION:
: THIS SUBROUTINE BUILDS REQUEST ID PACKETS FOR TRANSMISSION
: FROM THE QNA. 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
LOADD(R4),R4      ; POINT R4 TO DATA BLOCK
MOV             #100,BUFLEN       ; MOVE BUFFER SIZE TO BUFLEN
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             PROTO2,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 #0,@XRGXNT  ; UPDATE POINTER TO NEXT RING ENTRY
MOV             #XRGXNT,(R5)
MOV             #0,(R5)
JSR             R4,PREG14
WORD           GETXNX ANCHOR
RETURN
RTS             PC
```

```

4838 .SBTTL GET?NX - GET NEXT TRANSMIT OR RECEIVE RING ENTRY
4839
4840 ;
4841 ; FUNCTIONAL DESCRIPTION
4842 ; THIS SUBROUTINE GETS THE NEXT TRANSMIT OR RECEIVE RING
4843 ; ENTRY. IT IS ENTERED AT SEPERATE POINTS DEPENDING ON
4844 ; WHICH RING IS BEING USED.
4845 ;
4846 ; INPUTS P1 FLAG INDICATING IF THE PREVIOUS DESCRIPTOR NEEDS REVALIDATING
4847 ; P2 THE ADDRESS OF THE RING POINTER TO BE UPDATED.
4848 ;
4849 ; OUTPUTS - THE RING POINTER IS UPDATED TO POINT TO THE NEXT AVAILABLE
4850 ; ENTRY.
4851 ;
4852 ; CALLING PROCEDURE - CALL GETXNX @P1,@P2 ; FOR TRANSMIT UPDATES
4853 ; CALL GETRXN @P1,@P2 ; FOR RECEIVE UPDATES
4854 ;
4855 ; SIDE EFFECTS - NONE
4856 ;
4857 ; REGISTER USAGE - R1 POINTS TO THE FIRST ENTRY IN THE RING
4858 ; R2 POINTS TO THE LAST ENTRY IN THE RING
4859 ; R3 IS THE ADDRESS OF THE RING POINTER TO BE UPDATED
4860 ;
4861 ;---
4861 077344 GETRXN::
4862 077344 013701 024216 MOV RRGSR,R1 ; MOVE FIRST RING ENTRY TO R1
4863 077350 013702 024232 MOV RRGSR,R2 ; MOVE LAST RING ENTRY TO R2
4864 077354 P#POP R3 ; SHOULD WE VALIDATE PREVIOUS DESCRIPTOR
4865 077356 014503 MOV -(R5),R3
4866 077360 001431 BEQ GETCOM ; BRANCH IF NOT
4867 077360 013703 024234 RNXPV::
4868 077364 005063 000000 MOV RRGPRV,R3 ; ELSE, GET PREVIOUS DESCRIPTOR
4869 077370 005063 000010 CLR FLAG(R3) ; REINIT THE FLAG
4870 077374 005063 000012 CLR STAT1(R3) ; REINIT THE STATUS
4871 077400 012763 100000 000002 CLR STAT2(R3) ; REINIT STATUS 2
4872 077406 020203 MOV @MVALID,DESC(R3) ; VALIDATE IT
4873 077410 001404 CMP R2,R3 ; IS THIS THE LAST DATA DESCRIPTOR?
4874 077412 062737 000014 024234 BEQ 10$ ; BRANCH IF YES
4875 077420 000410 ADD @12.,RRGPRV ; ELSE UPDATE PREVIOUS POINTER
4876 077422 10$: BR GETCOM ; AND DO COMMON STUFF
4877 077422 010137 024234 MOV R1,RRGPRV ; LAST ENTRY, POINT TO START OF LIST
4878 077426 000405 BR GETCOM ; GO TO COMMON MODE
4879 077430 GETXNX::
4880 077430 P#POP R3 ; THROW AWAY FIRST PARAMETER
4881 077432 014503 MOV -(R5),R3
4882 077436 013701 024214 MOV XRGSR,R1 ; MOVE FIRST RING ENTRY TO R1
4883 077442 013702 024230 MOV XRGSR,R2 ; MOVE LAST RING ENTRY TO R2
4884 077442 P#POP R3 ; GET ADDRESS OF RING POINTER IN R3
4885 077444 014503 MOV -(R5),R3
4886 077446 021302 CMP (R3),R2 ; SEE IF POINTER POINTS TO LAST RING
4887 077450 062713 000014 BEQ 10$ ; IF YES, BRANCH
4888 077454 020327 024226 ADD @12.,(R3) ; ELSE, ADD ENTRY LENGTH TO POINTER
4889 077460 001073 CMP R3,@RGNXT ; IS THIS THE NEXT RECEIVE POINTER?
4890 077462 012704 024226 BNE 20$ ; BRANCH IF NOT
4891 077466 062704 000014 MOV @RGNXT,R4 ; ELSE GET NEXT RING DESCRIPTOR
; POINT TO NEXT DESCRIPTOR

```



```

4930 .SBTTL          BLDBUF  BUILD MESSAGE BUFFERS
4931 :
4932 : *
4933 : FUNCTIONAL DESCRIPTION
4934 : THIS SUBROUTINE CREATES A MESSAGE BUFFER TO BE USED
4935 : FOR TRANSMISSION.
4936 :
4937 : INPUTS          P1  BUFFER LOCATION TO PUT BUILT MESSAGE
4938 : IMPLICIT        P&SIZE CONTAINS THE SIZE THE BUFFER IS TO BE
4939 : P&TYPE          P&TYPE CONTAINS THE MESSAGE TYPE
4940 :
4941 : OUTPUTS -      IMPLICIT - BUFFER STARTING AT LOCATION P1 CONTAINS A
4942 : MESSAGE P&SIZE BYTES LONG USING THE MESSAGE
4943 : TYPE SPECIFIED BY P&TYPE.
4944 :
4945 : CALLING PROCEDURE  CALL BLDBUF P1
4946 :
4947 : SIDE EFFECTS      NONE
4948 :
4949 : REGISTER USAGE - R1 POINTS TO THE NEXT BYTE OF STORED MESSAGE TO BE BUILT
4950 : IN THE MESSAGE BUFFER.
4951 : R2 = (MESSAGE TYPE X 2), USED AS OFFSET FOR POINTERS
4952 : R3 POINTS TO THE NEXT BYTE OF THE BUFFER UNDER CONSTRUCTION
4953 : R4 POINTS TO THE LAST BYTE OF THE BUFFER UNDER CONSTRUCTION
4954 :
4955 : - *
4956 : BLDBUF::
4957 : P&POP          R3
4958 :                : PUT BUFFER ADDRESS INTO R3
4959 :                : MOV      -(R5),R3
4960 : MOV            P&TYPE,R2
4961 :                : PUT MESSAGE TYPE INTO R2
4962 : ASL            R2
4963 :                : MULTIPLY BY 2
4964 : MOV            P&SIZE,R4
4965 :                : PUT SIZE INTO R4
4966 : ADD            R3,R4
4967 :                : MAKE R4 = LAST BYTE OF BUFFER
4968 : MOV            MSGAD(R2),R1
4969 :                : POINT R1 TO FIRST BYTE OF STORED MESSAGE
4970 : CLR            COUNT
4971 :                : CLEAR BYTE COUNTER
4972 : 10$: INC        COUNT
4973 :                : COUNT NO. OF BYTES COPIED
4974 : MOV            (R1)+,(R3)+
4975 :                : PUT BYTE IN BUFFER
4976 : CMP            MSGCNT(R2),COUNT
4977 :                : ARE WE AT END OF STORED MESSAGE
4978 : BNE            20$
4979 :                : IF NO, CHECK IF DONE
4980 : MOV            MSGAD(R2),R1
4981 :                : ELSE, POINT R1 TO BEGINING
4982 : CLR            COUNT
4983 :                : AND CLEAR COUNTER
4984 : 20$: CMP        R3,R4
4985 :                : IS BUFFER FILLED?
4986 : BNE            10$
4987 :                : IF NO, LOOP
4988 : RETURN
4989 :                : ELSE, RETURN
4990 :                : RTS      PC
    
```

```

          .SBTTL          DATCMP - COMPARE DATA BUFFERS
4972      ;
4973      ; FUNCTIONAL DESCRIPTION
4974      ;
4975      ; THIS SUBROUTINE COMPARES TWO DATA BUFFERS BYTE BY BYTE.
4976      ; IF COMPARISON ERRORS OCCURED, LOCATION, EXPECTED DATA
4977      ; AND RECEIVED DATA ARE PRINTED OUT FOR THE FIRST FIVE
4978      ; ERRORS. THE TOTAL NUMBER OF ERRORS IS ALSO PRINTED.
4979      ;
4980      ; INPUTS          P1 - THE SIZE (IN BYTES) OF THE BUFFER TO BE COMPARED.
4981      ;                P2 - THE ADDRESS OF BUFFER TO COMPARE OTHER BUFFER AGAINST.
4982      ;                P3  THE ADDRESS OF THE SECOND BUFFER.
4983      ;
4984      ; OUTPUTS        P4  THE NUMBER OF COMPARISON ERRORS.
4985      ;
4986      ; CALLING PROCEDURE - CALL DATCMP P1,P2,P3
4987      ;                   P$POP P4
4988      ;
4989      ; SIDE EFFECTS   NONE.
4990      ;
4991      ; REGISTER USAGE - R1 CONTAINS THE COMPARE SIZE
4992      ;                 R2 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 1
4993      ;                 R3 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 2
4994      ;                 R4 CONTAINS THE BYTE OFFSET (BYTES FROM BEGINING OF BUFFER)
4995      ;
4996      ; **
4997 077734 DATCMP::
4998 077734 P$POP R1,R2,R3 ; PUT COMPARE SIZE IN R1
          MOV      -(R5),R1
          MOV      (R5),R2
          MOV      (R5),R3
4999      ; BUFFER 1 ADDRESS IN R2 AND
5000      ; BUFFER 2 ADDRESS IN R3
5001 077742 005004 CLR R4 ; INITIALIZE BYTE OFFSET
5002 077744 005037 050314 CLR TEMP ; AND ERROR COUNTER
5003 077750 005204 10$: INC R4 ; INCREMENT OFFSET COUNTER
5004 077752 121213 CMPB (R2),(R3) ; COMPARE BUFFERS
5005 077754 001421 BEQ 20$ ; IF SAME, BRANCH
5006 077756 005237 050314 INC TEMP ; INCREMENT ERROR COUNTER
5007 077762 022737 000005 050314 CMP #5,TEMP ; IF MORE THAN 5 ERRORS,
5008 077770 002413 BLT 20$ ; DON'T PRINT MESSAGE
5009 077772 PRINTX @CMPE1,R4,(R2),(R3) ; PRINT ERROR MESSAGE
          MOV      (R3),(SP)
          MOV      (R2),(SP)
          MOV      R4,(SP)
          MOV      @CMPE1,(SP)
          MOV      #4,(SP)
          MOV      SP,R0
          TRAP    C$PNTX
          ADD     @12,SP
5010 100020 005202 20$: INC R2 ; INCREMENT BUFFER 1 POINTER
5011 100022 005203 INC R3 ; INCREMENT BUFFER 2 POINTER
5012 100024 005301 DEC R1 ; DECREMENT COMPARE SIZE
5013 100026 001350 BNE 10$ ; IF NOT FINISHED, GO BACK FOR MORE
5014 100030 022737 000000 050314 CMP #0,TEMP ; WERE THERE ANY ERRORS?
5015 100036 001412 BEQ 30$ ; IF NO, EXIT
5016 100040 PRINTX @CMPE2,TEMP
          MOV      TEMP,(SP)

```



	100044	012746	067677	
	100050	012746	000002	
	100054	010600		
	100056	104415		
	100060	062706	000006	
5017	100064			301: RETURN TEMP
	100064	013725	050314	
	100070	000207		

	MOV	#CMPER2, (SP)
	MOV	#2, (SP)
	MOV	SP, R0
	TRAP	C\$PNTX
	ADD	#6, SP
		; RETURN WITH ERROR COUNT ON STACK
	MOV	TEMP, (R5).
	RTS	PC

<9

```

5019 .SBTTL WRITES - WRITE DATA ONTO SUMMARY TABLE
5020 ;
5021 ; FUNCTIONAL DESCRIPTION:
5022 ; THIS SUBROUTINE UPDATES THE SUMMARY TABLE DATA FOR
5023 ; THE NODES SPECIFIED IN THE CALL STATEMENT. EITHER ONE
5024 ; OR TWO NODES CAN BE UPDATED PER CALL. AFTER THE CALL,
5025 ; THE SUMMARY DATA COUNTERS ARE CLEARED. THE SUMMARY TABLE
5026 ; IS CHECKED FOR A MATCHING NODE ADDRESS AND UPDATES THE
5027 ; DATE FOR THAT NODE, OR ADDS THE NODE TO THE TABLE IF IT
5028 ; DOESN'T EXIST. AN ERROR IS REPORTED IF THE END OF THE TABLE
5029 ; IS REACHED.
5030 ;
5031 ; INPUTS P1 THE NUMBER OF NODES TO UPDATE (1 OR 2).
5032 ; P2 - THE ADDRESS OF THE FIRST NODE ADDRESS.
5033 ; P3 THE ADDRESS OF THE SECOND NODE ADDRESS IF P1 = 2 OR
5034 ; BLANK IF P1 = 1.
5035 ;
5036 ; OUTPUTS - THE SUMMARY TABLE IS UPDATED.
5037 ;
5038 ; CALLING PROCEDURE - CALL WRITES P1,P2(,P3)
5039 ;
5040 ; SIDE EFFECTS - THE SUMMARY COUNTERS ARE CLEARED.
5041 ;
5042 ; REGISTER USAGE - R1 POINTS TO THE CURRENT LOCATION IN THE SUMMARY TABLE.
5043 ; R2 POINTS TO THE NODE TO BE UPDATED'S ADDRESS.
5044 ; R3 IS SCRATCH
5045 ; R4 HOLDS THE SECOND NODE TO BE UPDATED ADDRESS.
5046 ;
5047 ;---+
5048 100072 WRITES: P$POP TEMP ; SEE HOW MANY NODES TO WRITE
5049 100072 014537 050314 000001 CMP TEMP,#1 ; IF ONLY ONE, GET ADDRESS
5050 100104 001002 BNE 10$
5051 100106 P$POP R2 MOV (R5),R2
5052 100110 000402 BR 20$
5053 100112 10$: P$POP R2,R4 ; IF TWO, GET BOTH ADDRESSES
5054 100112 014502 MOV #STATBL,R1 ; MOVE STATISTICAL TABLE ADDRESS INTO R1
5055 100116 012701 002714 20$: TST (R1) ; SEE IF SLOT IS EMPTY
5056 100122 005711 30$: BEQ 40$ ; IF YES, BR
5057 100124 001415 BEQ 80$ ; IF YES, BR
5058 100126 021127 177777 CMP (R1),#-1 ; SEE IF TABLE FULL
5059 100132 001454 BEQ 80$ ; IF YES, ERROR
5059 100134 CALL CMPADR R1,R2 ; LOOK FOR MATCHING ADDRESS
5059 100134 010225 MOV R2,(R5)+
5059 100136 010125 MOV R1,(R5)+
5059 100140 004437 070656 JSR R4,PREG14
5059 100144 026004 .WORD CMPADR-ANCHOR
5060 100146 P$POP R3 MOV -(R5),R3
5061 100150 014503 BEQ 50$ ; IF YES, BR
5062 100152 062701 000026 ADD #26,R1 ; ELSE, POINT R1 TO NEXT ENTRY
5063 100156 000761 BR 30$ ; AND CHECK AGAIN
5064 100160 011211 40$: MOV (R2),(R1) ; ADD NODE ADDRESS TO TABLE
5065 100162 016261 000002 000002 MOV (R2),2(R1) ; SIX BYTES WORTH
5066 100170 016261 000004 000004 MOV 4(R2),4(R1)
    
```

```

5067 100176 062701 000006      50$:  ADD      #6,R1      ; POINT R1 TO DATA
5068 100202 063721 050032      ADD      S.NREC,(R1)+ ; UPDATE SUMMARY DATA, RECEIVES NOT COMPLETE
5069 100206 063721 050030      ADD      S.REC,(R1)+  ; RECEIVES COMPLETE
5070 100212 063721 050034      ADD      S.LEN,(R1)+  ; LENGTH ERRORS
5071 100216 063721 050036      ADD      S.COMP,(R1)+ ; COMPARE ERRORS
5072 100222 063721 050040      ADD      S.BYTE,(R1)+ ; BYTES COMPARED
5073 100226 103001              BCC      60$          ; IF OVERFLOW, INCREMENT NEXT WORD
5074 100230 005511              ADC      (R1)
5075 100232 062701 000002      60$:  ADD      #2,R1      ; POINT R1 TO NEXT DATA
5076 100236 063721 050042      ADD      S.XFER,(R1)+ ; BYTES TRANSFERED
5077 100242 103001              BCC      70$          ; IF OVERFLOW, INCREMENT NEXT WORD
5078 100244 005511              ADC      (R1)
5079 100246 062701 000002      70$:  ADD      #2,R1      ; POINT R1 TO NEXT DATA
5080 100252 005337 050314      DEC      TEMP        ; DECR NO OF NODES COUNTER
5081 100256 001414              BEQ      90$          ; IF NO MORE, EXIT
5082 100260 010402              MOV      R4,R2       ; POINT R2 TO NEXT NODE
5083 100262 000715              BR       20$         ; AND UPDATE SUMMARY DATA
5084 100264              PRINTF  #TABFUL,#SUMM ; PRINT TABLE FULL MESSAGE
      100264 012746 053422              MOV      #SUMM,(SP)
      100270 012746 053274              MOV      #TABFUL,(SP)
      100274 012746 000002              MOV      #2,(SP)
      100300 010600              MOV      SP,R0
      100302 104417              TRAP    C$PNTF
      100304 062706 000006              ADD     #6,SP
5085 100310 005037 050032      90$:  CLR      S.NREC      ; CLEAR SUMMARY DATA COUNTERS
5086 100314 005037 050030      CLR      S.REC
5087 100320 005037 050034      CLR      S.LEN
5088 100324 005037 050036      CLR      S.COMP
5089 100330 005037 050040      CLR      S.BYTE
5090 100334 005037 050042      CLR      S.XFER
5091 100340 000207      RETURN

```

RTS PC

```

5093 .SBTTL          BINDEC CONVERT A 32 BIT BINARY NUMBER TO DECIMAL
5094 :---+
5095 : FUNCTIONAL DESCRIPTION:
5096 : THIS SUBROUTINE CONVERTS A 32 BIT BINARY NUMBER TO
5097 : A DECIMAL NUMBER REPRESENTED AS AN ASCIZ STRING.
5098 :
5099 : INPUTS -      P1 - THE ADDRESS OF THE FIRST WORD OF BINARY DATA
5100 :              BITS 0-15. THE SECOND WORD, BITS 16-31, IS
5101 :              EXPECTED TO IMMEDIATELY FOLLOW THE FIRST WORD.
5102 :
5103 : OUTPUTS -     THE ASCII STRING WILL BE LOCATED STARTING AT DECSTR
5104 :
5105 : SIDE EFFECTS - NONE
5106 :
5107 : REGISTER USAGE - R1 POINTS TO BITS 0-15 OF BINARY DATA
5108 :                  R2 POINTS TO BITS 16-31 OF BINARY DATA
5109 :                  R3 POINTS TO THE OUTPUT STRING
5110 :                  R4 POINTS TO THE POWERS OF 10 TABLE
5111 :
5112 :---+
5113 BINDEC::
5114 P$POP      R1          ; PUT ADDRESS OF BINARY WORD INTO R1
                    MOV      -(R5),R1
5115 100342    014501
5116 100344    010546      MOV      R5,-(SP)
5117 100346    010102      MOV      R1,R2          ; PUT ADDRESS OF SECOND WORD INTO R2
5118 100350    062702    000002      ADD      #2,R2
5119 100354    012703    100526      MOV      #DECSTR,R3      ; PUT ADDRESS OF OUTPUT STRING INTO R3
5120 100360    012704    100456      MOV      #TENPWR,R4      ; ADDRESS OF TEN POWER TABLE
5121 100370    012705    100460      MOV      #TENPWR+2,R5
5122 100376    005037    100542      MOV      #10,40$
5123 100402    161411      10$: CLR      PART          ; CLEAR PARTIAL COUNTER
5124 100404    005612      20$: SUB      (R4),(R1)      ; SUBTRACT 10 POWER
5125 100406    161512      SBC      (R2)
5126 100410    002403      SUB      (R5),(R2)
5127 100412    005237    100542      BLT      30$          ; BRANCH IF 10 POWER TOO LARGE
5128 100416    000771      INC      PART          ; ELSE ADD 1 TO PARTIAL
5129 100420    062411      BR      20$          ; LOOP
5130 100422    005512      30$: ADD      (R4)+,(R1)      ; RESTORE BINARY WORDS
5131 100424    062412      ADC      (R2)          ; AND POINT R4 TO NEXT TABLE ENTRIES
5132 100426    022525      ADD      (R4)+,(R2)
5133 100430    052737    000060    100542      CMP      (R5)+,(R5)+
5134 100436    113723    100542      BIS      #'0,PART      ; CHANGE PARTIAL TO ASCII
5135 100442    005327      MOV      PART,(R3)+    ; AND PUT INTO OUTPUT STRING
5136 100444    000000      DEC      (PC)+        ; HAVE WE DONE ALL 10 DIGITS
5137 100446    001353      40$: .WORD 0          ; IF NO, BRANCH
5138 100450    105023      BNE     10$          ; IF YES, TERMINATE WITH ZERO
5139 100452    012605      CLR     (R3)+
5140 100454    000207      MOV     (SP)+,R5
                    RETURN
                    RTS      PC
5141 100456
5142 100456    145000      TENPWR: 145000          ; 1.0 E09
5143 100460    035632      35632
5144 100462    160400      160400          ; 1.0 E08
5145 100464    002765      2765
5146 100466    113200      113200          ; 1.0 E07
5147 100470    000230      230
    
```

5148	100472	041100			: 1.0 E06
5149	100474	000017			
5150	100476	103240			: 1.0 E05
5151	100500	000001			
5152	100502	023420			: 1.0 E04
5153	100504	000000			
5154	100506	001750			: 1.0 E03
5155	100510	000000			
5156	100512	000144			: 1.0 E02
5157	100514	000000			
5158	100516	000012			: 1.0 E01
5159	100520	000000			
5160	100522	000001			: 1.0 E00
5161	100524	000000			

5162					
5163	100526		DECSTR: .BLKB	12.	: 12 BYTES FOR ASCIZ OUTPUT STRING
5164	100542	000000	PART: .WORD	0	: PARTIAL COUNTER

```

5166 .SBTTL          COMMAND LINE TRAVERSE ROUTINES
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184 100544
5185 100544 013704 003204
5186 100550 013703 003206
5187 100554 121327 000003
5188 100560 003405
5189 100562 105714
5190 100564 001441
5191 100566 121327 000013
5192 100572 003023
5193 100574 111301
5194 100576 006301
5195 100600 016101 100614
5196 100604 062701 100614
5197 100610 004711
5198 100612 000760
5199
5200
5201 100614 000114
5202 100616 000134
5203 100620 000152
5204 100622 000162
5205 100624 000204
5206 100626 000270
5207 100630 000612
5208 100632 000656
5209 100634 000270
5210 100636 000256
5211 100640 000744
5212
5213
5214
5215 100642 121314
5216 100644 001403
5217 100646 004737 100712
5218 100652 000740
5219 100654 005204
5220 100656 004737 100672
5221 100662 062703 000004
5222

```

```

;***
; P$TRV SUBROUTINE
; PARSE THE COMMAND LINE SUBROUTINE
; TAKE ACTIONS (VIA ACTION TREE) AS PARSING LINE
; PARSING DIRECTIONS FROM "CLI PARSING NODES"
; REGS USED:
;
; R1,R5=SCRATCH          P$NUM=NUMERIC CODE FROM DATA
; 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,R5
P$TRS:  CMPB (R3),#3          ;SEE IF ONE OF FIRST THREE SPECIAL CODES
BLE 10$          ;IF YES, DON'T CHECK INPUT STRING
TSTB (R4)          ;SEE IF ANY CHARS LEFT IN INPUT STRING
BEQ P$EXIT          ;BR IF NO
CMPB (R3),#11.          ;SEE IF SPECIAL CLI CHAR CODE OR ASCII
BGT 30$          ;BR IF REGULAR ASCII CHAR.
10$:  MOVB (R3),R1          ;GET SPECIAL CHAR CODE INTO R5
ASL R1
MOV 20$(R1),R1          ;BUILD TRAVERSE ROUTINE ADDRESS
ADD #20$,R1
JSR PC,(R1)          ;JSR TO SPECIAL CLI TRAVERSE ROUTINE
BR P$TRS          ;GO SEE IF MORE OF STRING LEFT

20$:  .WORD TRVERR 20$          ;TRAVERSE TABLE FOR "CLI FUNCTIONS"
      .WORD TRVEXI-20$          ;1
      .WORD TRVBR-20$          ;2
      .WORD TRVPF 20$          ;3
      .WORD TRVSPA-20$          ;4
      .WORD TRVNUM 20$          ;5
      .WORD TRVALP-20$          ;6
      .WORD TRVALN-20$          ;7
      .WORD TRVOCT 20$          ;8
      .WORD TRVDEC 20$          ;9
      .WORD TRVSTR 20$          ;10

;NOT A SPECIAL CODE
30$:  CMPB (R3),(R4)          ;SEE IF FIRST CHAR OF STRING IS A MATCH
BEQ 40$          ;BR IF A MATCH
JSR PC,TRVBRC          ;IF NOT A MATCH, GO TAKE MISS BRANCH
BR P$TRS          ;THEN GO BACK PT'G TO MISS NODE
40$:  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
; ADJUST PTR TO NEXT CLI NODE

```

```

COMMAND [THE TRAVERSE ROUTINES]
5223 100666 000732          BR      P%TR5
5224
5225 100670 000207          P%EXIT: RTS      PC          ;RETURN FROM PARSER
5226
5227
5228
5229          ;GOTO USER ACTION ROUTINE
5230 100672 116302 000001    TRVACT: MOVB     1(R3),R2      ;GET ACTION CODE FROM CLI MODE
5231 100676 042702 177400    BIC      #177400,R2          ;CLEAR ANY SIGN EXTENSION
5232 100702 013701 003210    MOV      P%ACT,R1           ;GET ADDRESS OF CLI ACTION ROUTINE
5233 100706 004711          JSR      PC,(R1)            ;GO DO ACTION DEFINED BY CODE
5234 100710 000207          RTS      PC                ;RETURN TO CALLING CODE
5235
5236          ;TAKE BRANCH IN TREE
5237 100712 016301 000002    TRVBRC: MOV     2(R3),R1      ;GET BRANCH DISPLACEMENT FROM TREE
5238 100716 060103          ADD     R1,R3              ; AND POINT R3 TO THE "MISS" NODE
5239 100720 000207          RTS      PC                ; RETURN TO P%TRV
5240
5241          ;NO BRANCH TAKEN
5242 100722 062703 000004    TRVNOB: ADD     #4,R3        ;THINGS OK, UPDATE R3 TO POINT TO NEXT
5243 100726 000207          RTS      PC                ; NODE AND RETURN TO P%TRV
5244
5245          ;-----
5246          ;ERROR HANDLING
5247 100730 004737 100672    TRVERR: JSR     PC,TRVACT    ;TAKE ERROR ACTION
5248 100734 112737 177777    MOVB     #-1,P%GDBD        ;SET ERROR RETURN FLAG
5249 100742 005726          TST     (SP)              ;GET RID OF "JSR PUSH TO TRVERR"
5250 100744 000137 100670    JMP      P%EXIT           ;RETURN DIRECT TO EXIT OF P%TRV ROUTINE
5251
5252          ;EXIT ACTION CODE
5253 100750 004737 100672    TRVEXI: JSR     PC,TRVACT    ;TAKE EXIT ACTION
5254 100754 105037 003221    CLRB     P%GDBD           ;SET GOOD/BAD FLAG TO "SUCCESS (0)"
5255 100760 005726          TST     (SP)              ;GET RID OF "JSR PUSH TO TRVEXI"
5256 100762 000137 100670    JMP      P%EXIT           ;RETURN DIRECT TO EXIT OF P%TRV ROUTINE
5257
5258          ;BRANCH ACTION CODE
5259 100766 004737 100672    TRVBR:  JSR     PC,TRVACT    ;GO TAKE BRANCH ACTION
5260 100772 000137 100712    JMP      TRVBRC
5261
5262          ;BRANCH-IF ACTION CODE
5263 100776 004737 100672    TRVBIF: JSR     PC,TRVACT
5264 101002 105737 003221    TSTB     P%GDBD
5265 101006 001402          BEQ     10%
5266 101010 000137 100712    JMP      TRVBRC
5267 101014 000137 100722    10%:    JMP      TRVNOB
5268
5269          ;SPACE ACTION CODE
5270 101020 005001          TRVSPA: CLR     R1
5271 101022 121427 000011    10%:    CMPB     (R4),#11
5272 101026 001003          BNE     20%
5273 101030 005204          INC     R4
5274 101032 005201          INC     R1
5275 101034 000772          BR      10%
5276
5277 101036 121427 000040    20%:    CMPB     (R4),#40
5278 101042 001003          BNE     30%
5279 101044 005204          INC     R4

```

```

COMMAND LINE TRAVERSE ROUTINES
5280 101046 005201      INC      R1
5281 101050 000764      BR       10%
5282 101052 005701      30%:    TST      R1
5283 101054 001404      BEQ      40%
5284 101056 004737 100672  JSR      PC,TRVACT
5285 101062 000137 100722  JMP      TRVNOB
5286 101066 000137 100712  40%:    JMP      TRVBRC
5287
5288
5289 101072 012737 000012 003216 TRVDEC: MOV      #10.,P#RADX      ;USE DECIMAL AS RADIX AND ASSUME .
5290 101100 000137 101112      JMP      TRVNMA
5291 101104      TRVOCT: ;(SAME AS TRVNUM SINCE DEFAULT RADIX IS OCTAL)
5292 101104 012737 000010 003216 TRVNUM: MOV      #8.,P#RADX      ;USE OCTAL AS RADIX AND ASSUME .
5293 101112      TRVNMA: PUSH     R5
5294 101114 010546      MOV      R5,-(SP)
5295 101116 005001      CLR      R1      ;CLEAR DIGIT COUNTER
5296 101122 121427 000053      CMPB    (R4),#0   ;SEE IF THERE'S A + SIGN THERE
5297 101124 001001      BNE     10%      ; BR IF NO
5298 101126 121427 000055      BR      20%      ; ELSE P#RADX ALREADY SAYS +, JUST BR
5299 101132 001004      CMPB    (R4),#-   ;SEE IF THERE'S A - SIGN THERE
5300 101134 112737 177777 003217 30%:    BNE     30%      ; BR IF NO
5301 101142 005204      MOVB    #-1,P#RADX+1 ;SET "MINUS FLAG" (HI BYTE OF P#RADX)
5302      INC      R4      ;BUMP R4 TO POINT TO FIRST CHAR
5303 101144 121427 000060      30%:    CMPB    (R4),#0   ;SEE IF CHAR. LESS THAN A "0"
5304 101150 002434      BLT     60%      ;BR IF YES (NOT NUMERIC)
5305 101152 121427 000067      CMPB    (R4),#7   ;SEE IF CHAR. GREATER THAN A "/"
5306 101156 003426      BLE     50%      ; BR IF YES
5307 101160 123727 003216 000012      CMPB    P#RADX,#10. ;SEE IF IN DECIMAL MODE
5308 101166 001417      BEQ     40%      ; BR IF YES (CAN USE HIGHER LIMIT)
5309 101170 121427 000071      CMPB    (R4),#71  ;SEE IF DIGIT WAS A 8 OR 9
5310 101174 003022      BGT     60%      ;BR IF NON-NUMERIC
5311 101176      PRINTF  #CLIBRX      ;ELSE WAS A 8 OR 9 WHEN IN OCTAL RADIX
5312 101176 012746 052222      MOV      #CLIBRX,-(SP)
5313 101202 012746 000001      MOV      #1,-(SP)
5314 101206 010600      MOV      SP,R0
5315 101210 104417      TRAP    C#PNTF
5316 101212 062706 000004      ADD     #4,SP
5317 101216 112737 177777 003221      MOVB    #-1,P#GDBD ;SET ERROR RETURN FLAG
5318 101224 000475      BR      110%     ; PRINT ERROR AND TAKE MISS
5319
5320
5321 101226 121427 000071      40%:    CMPB    (R4),#71  ;SEE IF CHAR. GREATER THAN A 9
5322 101232 003003      BGT     60%      ;BR IF YES (NOT NUMERIC)
5323 101234 005204      50%:    INC      R4      ;UPDATE CMD LINE PTR TO NEXT CHAR.
5324 101236 005201      INC      R1      ;INDICATE A NUMERIC FOUND
5325 101240 000741      BR      30%      ;GO LOOK AT NEXT CHAR.
5326
5327 101242 005701      60%:    TST      R1      ;SEE IF FOUND ANY NUMERIC
5328 101244 001465      BEQ     110%     ;BR IF NO, TAKE "MISS" BRANCH
5329 101246 010405      MOV     R4,R5    ;GET POINTER TO START OF NUMERIC STRING
5330 101250 160105      SUB     R1,R5
5331 101252 005037 003214      CLR     P#NUM    ;CLEAR LOC. WHERE VALUE WILL BE STORED
5332 101256 112502      70%:    MOVB    (R5),R2  ;GET ASCII CHAR AND CONVERT IT TO A #
5333 101260 162702 060060      SUB     #60,R2
5334 101264 006337 003214      ASL     P#NUM    ;SHIFT CURRENT VALUE TO MAKE ROOM
5335 101270 103440      BCS     100%     ;ERROR IF NUMBER TOO BIG
5336 101272 013737 003214 003212      MOV     P#NUM,P#CNT ;SAVE FOR LATER IN CASE DECIMAL RADIX

```



```

5331 101300 006337 003214          ASL    P#NUM
5332 101304 103432                   BCS    100$      ;ERROR IF NUMBER TOO BIG
5333 101306 006337 003214          ASL    P#NUM
5334 101312 103427                   BCS    100$      ;ERROR IF NUMBER TOO BIG
5335 101314 123727 003216 000012    CMPB   P#RADX,#10. ;SEE IF DECIMAL RADIX
5336 101322 001004                   BNE    80$      ;BR IF NOT EQUAL
5337 101324 063737 003212 003214    ADD    P#CNT,P#NUM
5338 101332 103417                   BCS    100$      ;ERROR IF NUMBER TOO BIG
5339 101334 060237 003214          80$:   ADD    R2,P#NUM
5340 101340 103414                   BCS    100$      ;ERROR IF NUMBER TOO BIG
5341 101342 005301                   DEC    R1
5342 101344 001344                   BNE    70$
5343 101346 105737 003217          TSTB   P#RADX*1    ;SEE IF NUM WAS PRECEDED BY A SIGN
5344 101352 001402                   BEQ    90$      ; BR IF NO
5345 101354 005437 003214          NEG    P#NUM      ; ELSE NEGATE THE NUMBER BEFORE LEAVING
5346 101360          90$:   POP    R5      ;RESTORE R5
5347 101362 004737 100672          JSR    PC,TRVACT  ;SINCE NUMERIC FOUND, GO TAKE ACTION
5348 101366 000137 100722          JMP    TRVNOB     ;GO POINT R3 TO NEXT NODE
5349
5350 101372          100$:  PRINTF  @CLINBG      ;PRINT NUMBER TOO BIG ERROR
101372 012746 052175                   MOV    @CLINBG, (SP)
101376 012746 000001                   MOV    #1, (SP)
101402 010600                   MOV    SP,R0
101404 104417                   TRAP   C#PNTF
101406 062706 000004                   ADD    #4,SP
5351 101412 112737 177777 003221    MOVB   #-1,P#GDBD ;SET ERROR RETURN FLAG
5352 101420          110$:  POP    R5      ;RESTORE R5
101420 012605                   MOV    (SP),R5
5353 101422 000137 100712          JMP    TRVBRC     ;TAKE "MISS" BRANCH
5354
5355
5356 101426 005001          TRVALP: CLR    R1      ;CLEAR ALPHA FOUND FLAG
5357 101430 121427 000101    10$:   CMPB   (R4),#101 ;SEE IF CHAR. LESS THAN A "A"
5358 101434 002406                   BLT    20$      ;BR IF YES (NOT ALPHA)
5359 101436 121427 000132    CMPB   (R4),#132 ;SEE IF CHAR. GREATER THAN A "Z"
5360 101442 003003                   BGT    20$      ;BR IF YES (NOT ALPHA)
5361 101444 005204                   INC    R4      ;UPDATE CMD LINE PTR TO NEXT CHAR
5362 101446 005201                   INC    R1      ;INDICATE AN ALPHA WAS FOUND
5363 101450 000767                   BR     10$      ;GO LOOK AT NEXT CHAR.
5364 101452 005701          20$:   TST    R1      ;SEE IF ANY ALPHA'S WERE FOUND
5365 101454 001404                   BEQ    30$      ;BR IF NO
5366 101456 004737 100672          JSR    PC,TRVACT  ;IF ANY FOUND TAKE ACTION
5367 101462 000137 100722          JMP    TRVNOB     ;THEN UPDATE R3 TO NEXT NODE NO BRANCH
5368 101466 000137 100712          30$:   JMP    TRVBRC     ;NONE FOUND, TAKE MISS BRANCH
5369
5370 101472 005001          TRVALN: CLR    R1      ;CLEAR ALPHANUM FOUND FLAG
5371 101474 121427 000060    10$:   CMPB   (R4),#60  ;SEE IF CHAR. LESS THAN A "0"
5372 101500 002417                   BLT    30$      ;BR IF YES (NOT NUMERIC OR ALPHA)
5373 101502 121427 000072    CMPB   (R4),#72  ;SEE IF CHAR. GREATER THAN A "9"
5374 101504 003003                   BGT    20$      ;BR IF YES (NOT NUMERIC)
5375 101510 005204                   INC    R4      ;UPDATE CMD LINE PTR TO NEXT CHAR.
5376 101512 005201                   INC    R1      ;INDICATE A NUMERIC FOUND
5377 101514 000767                   BR     10$      ;GO LOOK AT NEXT CHAR.
5378 101516 121427 000101          20$:   CMPB   (R4),#101 ;SEE IF CHAR. LESS THAN A "A"
5379 101522 002406                   BLT    30$      ;BR IF YES (NOT ALPHA)
5380 101524 121427 000132    CMPB   (R4),#132 ;SEE IF CHAR. GREATER THAN A "Z"

```

5381	101530	003003		BGT	30:			;BR IF YES (NOT ALPHA)
5382	101532	005204		INC		R4		;UPDATE CMD LINE PTR TO NEXT CHAR
5383	101534	005201		INC		R1		;INDICATE AN ALPHA FOUND
5384	101536	000756		BR		10:		;GO LOOK AT NEXT CHAR.
5385	101540	005701		TST	30:	R1		;SEE IF ANY ALPHANUM'S WERE FOUND
5386	101542	001404		BEQ		40:		;BR IF NO
5387	101544	004737	100672	JSR		PC,TRVACT		;IF ANY FOUND TAKE ACTION
5388	101550	000137	100722	JMP		TRVNOB		;THEN UPDATE R3 TO NEXT NODE -NO BRANCH
5389	101554	000137	100712	JMP	40:	TRVBRC		;NONE FOUND, TAKE MISS BRANCH
5390	101560			TRVSTR:		PUSH		;SAVE R5
	101560	010546						MOV R5,-(SP)
5391	101562	010401		MOV		R4,R1		;POINT R1 TO CMD STRING
5392	101564	010305		MOV		R3,R5		
5393	101566	062705	000006	ADD		#6,R5		;POINT R5 TO MATCH STRING FROM CLI NODE
5394	101572	005037	003212	CLR		P%CNT		;CLEAR CHAR MATCH COUNT
5395	101576	105715		TSTB	10:	(R5)		;SEE IF END OF MATCH STRING YET
5396	101600	001411		BEQ		20:		;BR IF YES
5397	101602	105711		TSTB		(R1)		;SEE IF END OF CMD LINE YET
5398	101604	001407		BEQ		20:		;BR IF YES
5399	101606	121115		CMPB		(R1),(R5)		;SEE IF CHARACTERS MATCH
5400	101610	001005		BNE		20:		;BR IF NO
5401	101612	005237	003212	INC		P%CNT		;MATCH -INCREMENT MATCH COUNT
5402	101616	005201		INC		R1		;UPDATE STRING POINTERS
5403	101620	005205		INC		R5		
5404	101622	000765		BR		10:		;BR TO CONTINUE CHECKING CHARS.
5405								
5406	101624	005737	003212	TST	20:	P%CNT		;WHEN DONE SEE IF ANY MATCHES FOUND
5407	101630	001407		BEQ		30:		;BR IF NO, GO TAKE THE MISS BRANCH
5408	101632	010104		MOV		R1,R4		;POINT CMD POINTER TO END OF STRING &
5409	101634			POP		R5		;RESTORE R5
	101634	012605						MOV (SP),R5
5410	101636	004737	100672	JSR		PC,TRVACT		;IF A MATCH FOUND, GO DO MATCH ACTION
5411	101642	066303	000004	ADD		4(R3),R3		;UPDATE R3 TO NEXT NODE (NO BRANCH)
5412	101646	000207		RTS		PC		; (NO RETURN THRU TRVNOB SINCE DIFFERENT
5413								; DISPLACEMENT DUE TO MATCH STRING)
5414	101650			POP	30:	R5		;RESTORE R5
	101650	012605						MOV (SP),R5
5415	101652	000137	100712	JMP		TRVBRC		; GO TAKE BRANCH
5416								; (PARSED OK), 1 IF ILL CMD.....

5418  
5419  
5420  
5421  
5422  
5423  
5424  
5425  
5426  
5427  
5428  
5429  
5430  
5431  
5432  
5433  
5434  
5435  
5436  
5437  
5438  
5439  
5440  
5441  
5442  
5443  
5444  
5445  
5446  
5447  
5448  
5449  
5450  
5451  
5452  
5453  
5454  
5455  
5456  
5457  
5458  
5459  
5460  
5461  
5462  
5463  
5464  
5465  
5466  
5467  
5468  
5469  
5470  
5471  
5472  
5473  
5474

101656 005001  
101660 121427 000000  
101664 001423  
101666 121427 000040  
101672 002414  
101674 121427 000042  
101700 001454  
101702 121427 000057  
101706 001420  
101710 121427 000132  
101714 003003  
101716 005204  
101720 005201  
101722 000756  
101724 112737 177777 003221 20%  
101732 000464  
101734 005701 30%

```

.SBTTL TRVADR TRAVERSE COMMAND LINE INPUT ADDRESS

TRVADR TRAVERSE COMMAND LINE INPUT ADDRESS

THIS ROUTINE IS CALLED BY TWO DIFFERENT ACTION ROUTINES. THE
NODE ACTION ROUTINE CALLS IT TO PARSE THROUGH THE NODE
ADDRESS INPUT BY THE OPERATOR. THE OPRSEL ACTION ROUTINE
CALLS TRVADR TO PARSE THROUGH THE "OPERATOR SELECTED" MESSAGE
WHICH HAS BEEN INPUT IN THE COMMAND LINE. FOR A NODE ADDRESS,
THE ROUTINE LOOKS FOR A '/' AS A DELIMITER FOR THE ADDRESS,
AND REPLACES THE / WITH A NULL BYTE FOR USE BY THE ADDRESS
PACKING ROUTINE. WHEN CALLED BY THE OPRSEL ROUTINE, A ''
IS EXPECTED AS THE DELIMITER FOR THE OPERATOR SELECTED MESSAGE.
IF A NULL STRING IS ENTERED, AN ERROR MESSAGE IS PRINTED.

INPUTS - R4 - POINTS TO THE BEGINING OF THE ADDRESS
OR MESSAGE IN THE COMMAND LINE

OUTPUTS - SUMMARIZED IN TABLE BELOW

COMMAND LINE INPUT CONDITION : P%GDBD : R4 POINTS TO : CFLAG CONTAINS : P%MERR
-----
ILLEGAL CHAR. : -1 : ILL. CHAR. : : N/A
ADR./ASSIST : 0 : END OF LINE : CASIST : N/A
ADR./TARGET : 0 : END OF LINE : CTARGET : N/A
ADR. : : : : :
ADR./CHAR. OR "OPR SEL/CHAR. : : : : :
OTHER THAN "A" : -1 : / : CTARGET : N/A
"T" OR BLANK : : : : :
"" : 0 : CHAR. AFTER " : : 1
"OPR SEL" : 0 : CHAR. AFTER " : OPRSEL : 0

CALLING PROCEDURE JSR PC,TRVADR
REGISTER USAGE 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

TRVADR: CLR R1 ;CLEAR HEX DIGIT FOUND FLAG
10%: CMPB (R4),#0 ;SEE IF NULL CHAR.
BEQ 30% ; IF YES, RETURN
CMPB (R4),#40 ;SEE IF ILLEGAL CHARACTER
BLT 20% ;IF YES, BRANCH TO ERROR ROUTINE
CMPB (R4),#42 ;SEE IF CHAR. IS A '/'
BEQ 70% ;IF YES, BRANCH TO 70%
CMPB (R4),#57 ;SEE IF CHAR. IS A ''
BEQ 50% ;BRANCH IF YES
CMPB (R4),#132 ;SEE IF CHAR. GREATER THAN 'F
BGT 20% ; IF YES, ILLEGAL CHAR.
INC R4 ;UPDATE CMD LINE POINTER TO NEXT CHAR.
INC R1 ;INCIDATE A VALID CHAR. FOUND
BR 10% ;LOOK AT NEXT CHAR.
MOVB #1,P%GDBD ;SET ERROR FLAG
BR 90% ;RETURN
TST R1 ;SEE IF VALID CHARACTERS FOUND
    
```

5475	101736	001772				BEQ	20\$		; IF NO, ILLEGAL CHAR.
5476	101740	012737	000000	003734	40\$:	MOV	@CTARGET,CFLAG		;SET TARGET FLAG
5477	101746	000456				BR	90\$		;RETURN
5478	101750	005701			50\$:	TST	R1		;SEE IF VALID CHARACTERS FOUND
5479	101752	001764				BEQ	20\$		; IF NO, ILLEGAL CHAR.
5480	101754	112714	000000			MOVB	@0,(R4)		; IF YES, REPLACE "/" WITH NULL CHAR.
5481	101760	005204				INC	R4		;UPDATE CMD. LINE POINTER TO NEXT CHAR.
5482	101762	121427	000000			CMPB	(R4),@0		;IS NEXT CHAR. NULL
5483	101766	001764				BEQ	40\$		; IF YES, TAKE DEFAULT OF TARGET
5484	101770	121427	000101			CMPB	(R4),@'A		;IS NEXT CHAR. "A"
5485	101774	001412				BEQ	60\$		; IF YES, BR 60\$
5486	101776	121427	000124			CMPB	(R4),@'T		;IS NEXT CHAR. 'T'
5487	102002	001756				BEQ	40\$		; IF YES, SET TARGET FLAG
5488	102004	112737	177777	003221		MOVB	@ 1,P\$GDBD		; ELSE, SET ERROR FLAG.
5489	102012	005304				DEC	R4		; READJUST COMMAND LINE POINTER
5490	102014	112714	000057			MOVB	@'/(R4)		; AND REPLACE / IN CMD LINE TO FIX ERROR
5491	102020	000747				BR	40\$		; SET TARGET FLAG AND RETURN
5492	102022	012737	000001	003734	60\$:	MOV	@CASIST,CFLAG		;SET ASSIST FLAG
5493	102030	000425				BR	90\$		
5494	102032	005701			70\$:	TST	R1		;SEE IF ANY CHARACTERS TYPED
5495	102034	001407				BEQ	80\$		;IF NO, BRANCH TO 80\$
5496	102036	112714	000000			MOVB	@0,(R4)		;ELSE, REPLACE ' ' WITH NULL
5497	102042	012737	000006	003734		MOV	@OPRSEL,CFLAG		;SET OPERATOR SELECTED FLAG
5498	102050	005204				INC	R4		
5499	102052	000414				BR	90\$		;RETURN
5500	102054				80\$:	PRINTF	@NULSTR		;PRINT NULL STRING ERROR MESSAGE
	102054	012746	053076					MOV	@NULSTR,-(SP)
	102060	012746	000001					MOV	@1,(SP)
	102064	010600						MOV	SP,R0
	102066	104417						TRAP	C\$PNTF
	102070	062706	000004					ADD	@4,SP
5501	102074	112737	177777	003223		MOVB	@ 1,P\$MERR		;SET OPER. SELECTED MSG. ERROR FLAG
5502	102102	005204				INC	R4		;MOVE CMD. LINE POINTER TO NEXT CHAR.
5503	102104	000207			90\$:	RTS	PC		;RETURN

5505  
5506  
5507  
5508  
5509  
5510  
5511  
5512  
5513 102106  
102106

.SBTTL REPORT CODING SECTION

\*\*\*  
: THE REPORT CODING SECTION CONTAINS THE  
: "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.  
:

BGNRPT

L\$RPT::

5514  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5526

\*\*\*\*\*  
: THIS SECTION, WHICH IS OPTIONAL, CONTAINS THE CODE FOR PRINTING  
: STATISTICAL INFORMATION GATHERED BY THE DIAGNOSTIC. IT IS  
: EXECUTED BY THE OPERATOR COMMAND "PRINT" OR BY THE MACRO CALL  
: "DORPT". USE THE PRINTS MACRO TO PRINT THE INFORMATION.  
: USE FORMAT STATEMENTS AS IN THE PRINTB/PRINTX MACROS. IT IS  
: THE PROGRAMMER'S RESPONSIBILITY TO DEVISE AND IMPLEMENT THE  
: FORM AND CONTENT OF THE STATISTICS.  
\*\*\*\*\*

5527 102106 004737 105170  
5528 102112  
102112 000167  
102114 000000

JSR PC.ACTSUM  
EXIT RPT

.WORD J\$JMP  
.WORD L10006 2 .

5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5540

\*\*\*\*\*  
: INSERT LOCAL STORAGE THAT IS USED ONLY  
: DURING THE REPORT SECTION.  
\*\*\*\*\*  
\*\*\*\*\*  
: INSERT MESSAGES THAT ARE USED ONLY  
: DURING THE REPORT SECTION.  
\*\*\*\*\*

5541  
5542  
5543 102116  
102116  
102116 104425

.EVEN  
ENDRPT

L10006: TRAP C\$RPT

5545  
5546  
5547  
5548  
5549  
5550 102120  
102120  
5551 102120 177777  
5552 102122 177777  
5553 102124 177777  
5554 102126  
5556  
5557  
5558  
5559  
5560  
5561  
5562  
5563  
5564  
5565  
5566

```

.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 P-TABLE
; 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.
;*****

```

INITIALIZE SECTION

5569
5570
5571
5572
5573
5574 102126
102126
5575
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618 102126
5619 102126
102126 010446
102130 010346
102132 010246
5620 102134 022737 000020 003734
5621 102142 001004
5622 102144 005037 003734
5623 102150 000137 103344

```
.SBTTL INITIALIZE SECTION
***
; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
; AT THE BEGINNING OF EACH PASS.
;
BGNINIT
L$INIT::
*****
THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS. THERE
ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
PLACE. THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
CORRESPONDING EVENT FLAGS ARE:
START COMMAND EF.START
RESTART COMMAND EF.RESTART
CONTINUE COMMAND EF.CONTINUE
POWERDOWN/POWERUP EF.PWR
NEW PASS EF.NEW
EXAMPLE OF EVENT FLAG USE:
READEF @EF.START
BCOMplete STARTCODE
DURING THE INIT CODE, USE THE "GPHARD" MACRO TO OBTAIN P TABLE
INFORMATION FOR DEVICE TESTING. GET ONE UNIT'S INFORMATION IF
THIS IS A SEQUENTIAL DIAGNOSTIC. GET INFORMATION ON ALL
UNITS AVAILABLE FOR TESTING IF THIS IS AN EXERCISER. THE NUMBER
OF UNITS AVAILABLE IS IN A HEADER LOCATION: "L$UNIT".
*****
ROUTINE STEPS
CHECK ENTRY FLAGS AND TAKE CORRECT ACTION
INIT THE STACKS
LOCATE FREE MEMORY
VERIFY THAT A CLOCK OF SOME SORT EXISTS
GET THE QNA ADDRESS FROM THE PTABLE
SET UP THE QNA INTERRUPT VECTOR
CLEAR NODE TABLE
CLEAR SUMMARY TABLE
CLEAR COUNTER DATA BASE
SET UP FOR CLOCK INTERRUPTS
CALL THE QNA INIT ROUTINE
GET THE DEFAULT PHYSICAL ADDRESS AND PRINT IT
RESET THE CLOCK
EXIT
INIT:
PUSH R2,R3,R4 ; SAVE THE REGISTERS
MOV R4,(SP)
MOV R3,(SP)
MOV R2,(SP)
CMP @CEXIT,CFLAG ;SEE IF EXIT COMMAND TYPED
BNE INIT1 ; IF NO, DO INIT CODE
CLR CFLAG ; ELSE, CLEAR EXIT FLAG
JMP INICLN ; AND DO CLEANUP
```

```

5624 102154          INIT1:
5625 102154          READEF  #EF.START          ; IF HERE BECAUSE OF "START", DO INIT
      102154 012700 000040          MOV      #EF.START,RO
      102160 104447          TRAP     C#REFG
5626 102162          BCOMPLETE      START
      102162 103424          ; IF HERE BECAUSE OF "RESTART", DO SOME INIT
5627 102164          READEF  #EF.RESTART      ; IF HERE BECAUSE OF "RESTART", DO SOME INIT
      102164 012700 000037          MOV      #EF.RESTART,RC
      102170 104447          TRAP     C#REFG
5628 102172          BNCOMPLETE      10$
      102172 103002          BCC      10$
5629 102174 000137 103234          JMP      RESTRT
5630 102200          10$:
5631 102200          READEF  #EF.CONTINUE      ; IF HERE BECAUSE OF "CONTINUE", EXIT
      102200 012700 000036          MOV      #EF.CONTINUE,RO
      102204 104447          TRAP     C#REFG
5632 102206          BNCOMPLETE      20$
      102206 103002          BCC      20$
5633 102210 000137 103234          JMP      RESTRT
5634 102214          20$:
5635 102214          READEF  #EF.NEW          ; IF HERE ON NEW PASS, SKIP SOME INIT
      102214 012700 000035          MOV      #EF.NEW,RO
      102220 104447          TRAP     C#REFG
5636 102222          BNCOMPLETE      30$
      102222 103002          BCC      30$
5637 102224 000137 103334          JMP      NEW
5638 102230          30$:
5639 102230 000137 103354          JMP      INIEXI          ; IF DON'T KNOW WHY WE'RE HERE, EXIT
5640 102234          START:
5641 102234          I$STACK #1000,SP          ; SET PARAMETER STACK POINTER
      102234 012705 001000          MOV      #1000,R5          ; INITIALIZE THE PARAMETER STACK POINTER.
      102240 010606          MOV      SP,SP          ; INITIALIZE THE HARDWARE STACK POINTER.
5642 102242          MEMORY FRESIZ          ; GET FREE MEMORY INFO
      102242 104431          TRAP     C#MEM
      102244 010037 050022          MOV      RO,FRESIZ
5643 102250 013737 050022 050024          MOV      FRESIZ,FREMEM          ; SIZE OF FREE MEMORY IN FRESIZ
5644 102256 062737 000002 050024          ADD      #2,FREMEM          ; START OF FREE MEMORY IN FREMEM
5645 102264 012702 003736          MOV      #CLKCSR,R2          ; SETUP R2 AS A PRT. TO CLOCK INFO. BLOCK
5646 102270          CLOCK L,R1          ; GET LINE CLOCK INFO
      102270 012700 000114          MOV      #L,RO
      102274 104462          TRAP     C#CLK
      102276 010001          MOV      RO,R1
5647 102300          BNCOMPLETE      10$          ; IF NONE, SEE IF P CLOCK PRESENT
      102300 103006          BCC      10$
5648 102302 004737 070504          JSR      PC,CLKSET          ; SET UP CLOCK INFO TABLE AND VECTOR
5649 102306 012737 000100 003746          MOV      #LCLKEN,CLKEN      ; SET UP THE ENABLE LINE CLOCK DATA
5650 102314 000514          BR      40$
5651 102316          10$:
5652 102316          CLOCK P,R1          ; GET P CLOCK INFO
      102316 012700 000120          MOV      #P,RO
      102322 104462          TRAP     C#CLK
      102324 010001          MOV      RO,R1
5653 102326          BNCOMPLETE      20$          ; IF NO CLOCK, ERROR
      102326 103017          BCC      20$
5654 102330 004737 070504          JSR      PC,CLKSET          ; ELSE SET UP CLOCK INFO AND VECTOR
5655 102334 062737 000002 003736          ADD      #2,CLKCSR          ; POINT CLKCSR TO P CLK COUNT SET REG.
5656 102342 012777 001f00 101366          MOV      #PCLKCT,#CLKCSR      ; LOAD CLK SET REG. WITH COUNT VALUE

```



```

5657 102350 162737 000002 003736      SUB      #2,CLKCSR      ;POINT CLKCSR BACK TO P-CLK CSR
5658 102356 012737 000111 003746      MOV      #PCLKEN,CLKEN ;SETUP TO ENABLE P-CLK DATA
5659 102364 000470                      BR       40$
5660 102366                      20$:
5661 102366 012737 000100 003742      MOV      #100,CLKVEC   ; SET UP INTERRUPT VECTOR
5662 102374 012737 003746 003736      MOV      #CLKEN,CLKCSR ; FAKE A CSR LOCATION
5663 102402 012737 000001 003756      MOV      #1,TIMER1    ; SET TIMER 1 VALUE
5664 102410                      SETVEC   CLKVEC,#CLKINT,#0 ; ENABLE CLOCK INTERRUPTS
      02410 012746 000000                      MOV      #0,-(SP)
      02414 012746 070530                      MOV      #CLKINT,-(SP)
      102420 013746 003742                      MOV      CLKVEC,-(SP)
      102424 012746 000003                      MOV      #3,-(SP)
      102430 104437                      TRAP     C$SVEC
      102432 062706 000010                      ADD      #10,SP
5665 102436                      DELAY    50              ; WAIT SOME TIME FOR CLOCK INTERRUPTS
      102436 012727 000050                      MOV      #50,(PC)
      102442 000000                      .WORD   0
      102444 013727 002116                      MOV      L$DLY,(PC)
      102450 000000                      .WORD   0
      102452 005367 177772                      DEC      6(PC)
      102456 001375                      BNE     . 4
      102460 005367 177756                      DEC     -22(PC)
      102464 001367                      BNE     . 20
5666 102466 005737 003756      TST      TIMER1        ; DID TIMER1 DECREMENT TO ZERO?
5667 102472 001013      BNE     30$            ; NO, NO CLOCK EXISTS. REPORT ERROR
5668 102474                      GMANID   L5060,CLKHZ,D,377,50.,60.,YES ; ASK IF 50 OR 60 HZ
      102474 104443                      TRAP     C$GMAN
      102476 000406                      BR       10000$
      102500 003744                      .WORD   CLKHZ
      102502 000052                      .WORD   T$CODE
      102504 052242                      .WORD   L5060
      102506 000377                      .WORD   377
      102510 000062                      .WORD   T$LOLIM
      102512 000074                      .WORD   T$HILIM
      102514                      10000$:
5669 102514 004737 070504      JSR      PC,CLKSET
5670 102520 000412      BR       40$          ; CONTINUE
5671 102522                      30$:
5672 102522                      PRINTF   #NOCLK        ;ERROR MESSAGE
      102522 012746 061766                      MOV      #NOCLK,(SP)
      102526 012746 000001                      MOV      #1,-(SP)
      102532 010600                      MOV      SP,RO
      102534 104417                      TRAP     C$PNTF
      102536 062706 000004                      ADD      #4,SP
5673 102542 000137 103344      JMP      INICLN        ;CANNOT CONTINUE, DO CLEANUP
5674 102546                      40$:
5675 102546                      GPHARD   #0,R1         ;GET P-TAB POINTER FOR THIS UNIT
      102546 012700 000000                      MOV      #0,RO
      102552 104442                      TRAP     C$GPHRD
      102554 010001                      MOV      RO,R1
5676 102556                      BCOMPLETE 50$         ;THIS ONE IS NOT AVAILABLE
      102556 103402                      BCS     50$
5677 102560 000137 103344      JMP      INICLN
5678 102564                      50$:
5679 102564 012137 047772      MOV      (R1)+,QNAADO  ;SAVE DEVICE ADDRESS
5680 102570 012137 047774      MOV      (R1)+,QNAVCO  ;SAVE VECTOR
5681 102574 012137 047776      MOV      (R1)+,QNAAPRO ;SAVE PRIORITY

```

INITIALIZE SECTION

```

5682 102600          SETVEC  QNAVCO,#QNAISR,QNAPRO ;SETUP QNA INTERRUPT VECTOR
      102600 013746 047776          MOV      QNAPRO,-(SP)
      102604 012746 073064          MOV      #QNAISR,-(SP)
      102610 013746 047774          MOV      QNAVCO,-(SP)
      102614 012746 000003          MOV      #3,-(SP)
      102620 104437          TRAP    C$SVEC
      102622 062706 000010          ADD      #10,SP
5683 102626 012703 000050          MOV      #TBLEN,R3          ;CLEAR NODE TABLE
5684 102632 012702 002412          MOV      #NODTBL,R2
5685 102636          60$:
5686 102636 005022          CLR      (R2)+
5687 102640 005303          DEC      R3
5688 102642 001375          BNE     60$
5689 102644 012703 000132          MOV      #STBL,R3          ;CLEAR SUMMARY TABLE
5690 102650 012702 002714          MOV      #STATBL,R2
5691 102654          70$:
5692 102654 005022          CLR      (R2)+
5693 102656 005303          DEC      R3
5694 102660 001375          BNE     70$
5695 102662 005037 050266          CLR      BCOUNT          ; CLEAR UNEXPLAINED ERROR COUNTER
5696 102666 005037 050270          CLR      ERRFLG          ; CLEAR ERROR FLAG
5697 102672 005037 050252          CLR      FATFLG          ; CLEAR FATAL FLAG ERROR
5698 102676 005037 050032          CLR      S.NREC          ; CLEAR SUMMARY DATA COUNTERS
5699 102702 005037 050030          CLR      S.REC
5700 102706 005037 050034          CLR      S.LEN
5701 102712 005037 050036          CLR      S.COMP
5702 102716 005037 050040          CLR      S.BYTE
5703 102722 005037 050042          CLR      S.XFER
5704 102726 012702 050144          MOV      #CNTRS,R2          ; GET BASE ADDRESS OF COUNTER DATA BASE
5705 102732 012203          MOV      (R2)+,R3          ; GET SIZE OF COUNT DATA BASE
5706 102734 006203          ASR     R3          ; TURN BYTE COUNT INTO WORD COUNT
5707 102736          80$:
5708 102736 005022          CLR      (R2)+          ; CLEAR THIS COUNT
5709 102740 005303          DEC      R3          ; KNOCK DOWN COUNT BY ONE
5710 102742 001375          BNE     80$          ; BRANCH IF MORE TO DO
5711 102744 005037 003750          CLR      TIMMIN          ;CLEAR TIME SINCE-START-LOCATIONS
5712 102750 005037 003752          CLR      TIMSEC
5713 102754 013737 003744 003754          MOV      CLKHZ,TIMTCK          ; LOAD TICKS/SEC
5714 102762          SETVEC  CLKVEC,#CLKINT,CLKBR ;SETUP CLOCK INTERRUPT VECTOR
      102762 013746 003740          MOV      CLKBR,-(SP)
      102766 012746 070530          MOV      #CLKINT,-(SP)
      102772 013746 003742          MOV      CLKVEC,-(SP)
      102776 012746 000003          MOV      #3,-(SP)
      103002 104437          TRAP    C$SVEC
      103004 062706 000010          ADD      #10,SP
5715 103010 013777 003746 100720          MOV      CLKEN,#CLKCSR          ;SET ENABLE BITS IN THE CLOCK TO START
5716 103016          SETPRI  #PRI00          ;SET PRIORITY=0 SO CLOCK CAN INTERRUPT
      103016 012700 000000          MOV      #PRI00,R0
      103022 104441          TRAP    C$SPRI
5717 103024          CALL   QNAINI          ;INITIALIZE THE QNA
      103024 004437 070656          JSR     R4,PREG14          .WORD QNAINI-ANCHOR
5718 103032          P$POP  R0          ; GET RETURN STATUS
      103032 014500          MOV      -(R5),R0
5719 103034 001401          BEQ    90$          ; BRANCH IF NO ERRORS
5720 103036 000542          BR     INICLN
5721 103040          90$:

```

```

5722 103040 012702 004010      MOV      @PHYADR,R2      ; GET CSR ADDRESS
5723 103044 012703 003774      MOV      @DEPADR,R3      ; GET DEFAULT PHYSICAL ADDRESS
5724 103050 012223              MOV      (R2),.(R3).      ; GET FIRST WORD OF THE ADDRESS
5725 103052 012223              MOV      (R2),.(R3).      ; GET SECOND WORD OF THE ADDRESS
5726 103054 011213              MOV      (R2),.(R3).      ; GET FINAL WORD OF THE ADDRESS
5727 103056 012703 047674      MOV      @IDTADR,R3      ; POINT TO SYSTEM ID MESSAGE
5728 103062 012702 004010      MOV      @PHYADR,R2      ; POINT TO PHYSICAL ADDRESS
5729 103066 011263 000006      MOV      (R2),SOURCC(R3) ; OUR ADDRESS IS CONSOLE ID SOURCE
5730 103072 016263 000002 000010  MOV      2(R2),SOURCC+2(R3) ; AGAIN
5731 103100 016263 000004 000012  MOV      4(R2),SOURCC+4(R3) ; AND AGAIN
5732 103106 011263 000042      MOV      (R2),SIADDR(R3) ; OUR ADDRESS IS CONSOLE ID DEVICE ADDRESS
5733 103112 016263 000002 000044  MOV      2(R2),SIADDR+2(R3) ; AGAIN
5734 103120 016263 000004 000046  MOV      4(R2),SIADDR+4(R3) ; AND AGAIN
5735 103126
5736 103126 1001:      CALL     BINHEX @DEPADR,@6,@STRBUF ; PUT ADDRESS INTO HEX FORMAT
103126 012725 002324      MOV      @STRBUF,(R5).
103132 012725 000006      MOV      @6,(R5).
103136 012725 003774      MOV      @DEPADR,(R5).
103142 004437 070656      JSR      R4,PREG14
103146 005150      .WORD   BINHEX ANCHOR
5737 103150 PRINTS @MMSG1,@STRBUF ; PRINT ADDRESS
103150 012746 002324      MOV      @STRBUF,(SP)
103154 012746 055046      MOV      @MMSG1,(SP)
103160 012746 000002      MOV      @2,(SP)
103164 010600      MOV      SP,R0
103166 104416      TRAP    C1PNTS
103170 062706 000006      ADD     @6,SP
5738 103174 012737 000000 002372  MOV      @ALPHA,P#TYPE ; SET MESSAGE DEFAULT VALUES
5739 103202 012737 001000 002374  MOV      @512,P#SIZE
5740 103210 012737 000001 002376  MOV      @1,P#CPYS
5741 103216 005037 003202      CLR     RSPFLG ; CLEAR RESPONDER MODE FLAG
5742 103222 005037 003750      CLR     TIMMIN ; CLEAR TIME SINCE START LOCATIONS
5743 103226 005037 050146      CLR     CNTRS.C.SECONDS ; CLEAR QNA TIME SINCE START
5744 103232 000450      BR      INIEXI
5745 103234
5746 103234 005037 002200      CLR     BLDFLG ; INDICATE THAT WE ARE NO LONGER BUILDING
5747 103240 005037 003202      CLR     RSPFLG ; CLEAR RESPONDER MODE FLAG
5748 103244 013737 003744 003754  MOV      CLKHZ,TIMTCK ; LOAD TICKS/SEC
5749 103252 SETVEC CLKVEC,@CLKINT,CLKBR ; SETUP CLOCK INTERRUPT VECTOR
103252 013746 003740      MOV      CLKBR,(SP)
103256 012746 070530      MOV      @CLKINT,(SP)
103262 013746 003742      MOV      CLKVEC,(SP)
103266 012746 000003      MOV      @3,(SP)
103272 104437      TRAP    C1SVEC
103274 062706 000010      ADD     @10,SP
5750 103300 013777 003746 100430  MOV      CLKEN,@CLKCSR ; SET ENABLE BITS IN THE CLOCK TO START
5751 103306 SETPRI @PRI00 ; SET PRIORITY=0 SO CLOCK CAN INTERRUPT
103306 012700 000000      MOV      @PRI00,R0
103312 104441      TRAP    C1SPRI
5752 103314 CALL     QNAINI ; INITIALIZE THE QNA
103314 004437 070656      JSR      R4,PREG14
103320 000220      .WORD   QNAINI ANCHOR
5753 103322 P#POP R0 ; GET RETURN STATUS
103322 014500      MOV      (R5),R0
5754 103324 001401      BEQ    101 ; BRANCH IF NO ERRORS
5755 103326 000406      BR      INICLN
5756 103330 101:

```

INITIALIZE SECTION

```

5757 103330 005037 003752          CLR      TIMSEC
5758 103334          NEW:          MOV      CLKEN, @CLKCSR      ;SET ENABLE BITS IN THE CLOCK TO START
5759 103334 013777 003746 100374    BR       INIEXI          ;EXIT
5760 103342 000404          INICLN:    POP      R2,R3,R4      ; RESTORE THE REGISTERS
5761 103344          MOV      (SP),R2
5762 103344          MOV      (SP),R3
103344 012602          MOV      (SP),R4
103346 012603          ; ABORT PASS
103350 012604          DOCLN          TRAP      C#DCLN
5763 103352          INIEXI:    POP      R2,R3,R4      ; RESTORE THE REGISTERS
103352 104444          MOV      (SP),R2
103354          MOV      (SP),R3
103356 012603          MOV      (SP),R4
103360 012604          EXIT      INIT          ; EXIT INIT SECTION
5766 103362          TRAP      C#EXIT
103362 104432          .WORD    L10010
103364 000002

5767          ; *****
5769          ;   INSERT LOCAL STORAGE THAT IS USED ONLY
5770          ;   DURING THE INITIALIZE SECTION.
5771          ; *****
5772          ; *****
5773          ;   INSERT MESSAGES THAT ARE USED ONLY
5774          ;   DURING THE INITIALIZE SECTION.
5775          ; *****
5776          ; *****
5777          ; *****
5779          .EVEN
5780          ENDINIT
5781
5782 103366          L10010:  TRAP      C#INIT
103366
103366 104411

```

AUTODROP SECTION

5784  
5785  
5786  
5787  
5788  
5789  
5790  
5791  
5792  
5793  
5794  
5796  
5797  
5798  
5799  
5801  
5802

103370  
103370  
  
103370  
103370  
103370 104461

.SBTTL AUTODROP SECTION

\*\*\*  
; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF  
; THE "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO  
; SEE IF THEY WILL RESPOND. THOSE THAT DON'T ARE IMMEDIATELY  
; DROPPED FROM TESTING.  
;

BGNAUTO

L\$AUTO::

;\*\*\*\*\*  
; INSERT CODE HERE TO CHECK DEVICE(S) TO SEE IF THEY RESPOND.  
; ISSUE A "DODU" FOR THOSE THAT DON'T.  
;\*\*\*\*\*

ENDAUTO

L10011: TRAP C\$AUTC

5804  
5805  
5806  
5807  
5808  
5809  
5810  
5811  
5812  
5813  
5814  
5815  
5816  
5817  
5818  
5819

.SBTTL CLEANUP CODING SECTION

\*\*\*  
; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED  
; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.  
; ROUTINE STEPS:  
; GET CSR ADDRESS  
; CLEAN UP ANY RECEIVE BUFFERS HANGING AROUND  
; CLEAR INTERRUPT AND RECEIVE ENABLE BITS  
; DISABLE CLOCKS  
; EXIT  
;--

5820 103372  
103372

BGNCLN

L\$CLEAN::

5821  
5823  
5824  
5825  
5826  
5827  
5828

\*\*\*\*\*  
; INSERT YOUR CLEANUP CODING. THIS CODING SHOULD  
; RESTORE YOUR TEST-DEVICE TO A NEUTRAL STATE.  
; THIS CODE WILL BE EXECUTED AFTER EACH PASS AND AFTER THE  
; PROGRAM IS INTERRUPTED BY "PC".  
\*\*\*\*\*

5830 103372  
103372 010225

P\$PUSH R2 ; PUT R2 ON THE STACK  
MOV R2,(R5).

5831 103374  
5832 103374

10\$:

SETPRI PRI00 ; MAKE SURE CLOCKS CAN INTERRUPT  
MOV PRI00,R0  
TRAP C\$SPRI

103374 013700 000000  
103400 104441

5833 103402 005737 050256  
5834 103406 001417

TST NIRCNT ; SEE IF RECEVE COUNT ZERO  
BEQ 20\$ ; IF YES, EXIT

5835 103410  
103410 004437 070656  
103414 003044

CALL RECEVE ; ELSE, CLEAR OUT UNWANTED ENTRIES  
JSR R4,PREG14  
.WORD RECEVE ANCHOR

5836 103416  
103416 014502

P\$POP R2  
MOV -(R5) R2

5837 103420 001765  
5838 103422 013702 024226

BEQ 10\$ ; IF ANY WERE FOUND, UPDATE NEXT POINTER  
MOV RRGNXT,R2 ; GET THE DESCRIPTOR  
CALL GETRNX #1,#RRGNXT ; UPDATE NEXT AND PREVIOUS DESCRIPTOR POINTERS

103426 012725 024226  
103432 012725 000001

MOV #RRGNXT,(R5).  
MOV #1,(R5).

103436 004437 070656  
103442 006450

JSR R4,PREG14  
.WORD GETRNX ANCHOR

5840 103444 000753

BR 10\$ ;SEE IF ANY MORE ENTRIES

5841 103446  
5842 103446 013702 047772

20\$:

MOV QNAADO,R2 ; GET DEVICE ADDRESS  
MOV #1,TIMER1 ; TIMER FOR INITIALIZE

5843 103452 012737 000001 003756  
5844 103460 012762 000002 000016

MOV #MRESET,CSR(R2) ; RESET THE DEVICE  
MOV #1,TIMER1 ; TIMER FOR INITIALIZE

5845 103466 012737 000001 003756  
5846 103474

30\$:

TST TIMER1 ; INIT DONE?  
BNE 30\$ ; BRANCH IF NOT

5847 103474 005737 003756  
5848 103500 001375

CLR CSR(R2) ; FINISH THE RESET  
CLR @CLKCSR ; DISABLE CLOCK

5849 103502 005062 000016  
5850 103506 005077 100224

SETPRI #PRI07 ; SET PROCESSOR PRIORITY BACK TO ?

5851 103512

103512 012700 000340  
103516 104441  
5852 103520  
103520 014502  
5853 103522  
103522 104432  
103524 000002

P#POP R2  
EXIT CLN

; REMOVE R2 FROM STACK  
MOV

MOV #PRI07,R0  
TRAP C#SPRI  
-(R5),R2  
TRAP C#EXIT  
.WORD L10012.

5854  
5856  
5857  
5858  
5859  
5860  
5861  
5862  
5863  
5864  
5866  
5867  
5868  
5869

;\*\*\*\*\*  
; INSERT LOCAL STORAGE THAT IS USED ONLY  
; DURING THE CLEANUP SECTION.  
;\*\*\*\*\*  
;\*\*\*\*\*  
; INSERT MESSAGES THAT ARE USED ONLY  
; DURING THE CLEANUP SECTION.  
;\*\*\*\*\*

.EVEN  
ENDCLN

103526  
103526  
103526 104412

L10012: TRAP C#CLEAN

5871  
5872  
5873  
5874  
5875  
5876  
5877  
5878 103530  
103530

.SBTTL DROP UNIT SECTION  
: \* \*  
: THE DROP UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE  
: TO NO LONGER BE TESTED.  
: -

BGNDU

L\$DU::

5879  
5881  
5882  
5883  
5884  
5885  
5886  
5888  
5889 103530

: \*\*\*\*\*  
: INSERT DROP CODE HERE. THIS CODE WILL BE EXECUTED AFTER  
: A "DROP" COMMAND OR A "DODU" MACRO EXECUTION. THE PURPOSE  
: OF THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A  
: UNIT HAS BEEN DROPPED. THIS SECTION IS OPTIONAL.  
: \*\*\*\*\*

EXIT JU

.WORD J\$JMP  
.WORD L10013-2 .

103530 000167  
103532 000000

5890  
5892  
5893  
5894  
5895  
5896  
5897  
5898  
5899  
5900  
5902  
5903  
5904  
5905

: \*\*\*\*\*  
: INSERT LOCAL STORAGE THAT IS USED ONLY  
: DURING THE DROP-UNIT SECTION.  
: \*\*\*\*\*

: \*\*\*\*\*  
: INSERT MESSAGES THAT ARE USED ONLY  
: DURING THE DROP-UNIT SECTION.  
: \*\*\*\*\*

.EVEN

ENDDU

L10013: TRAP C\$DU

103534  
103534  
103534 104453



5907  
5908  
5909  
5910  
5911  
5912  
5913  
5914  
5915 103536  
103536  
5916  
5918  
5919  
5920  
5921  
5922  
5923  
5925  
5926 103536  
103536 000167  
103540 000000  
5927  
5929  
5930  
5931  
5932  
5933  
5934  
5935  
5936  
5937  
5939  
5940  
5941  
5942 103542  
103542  
103542 104452

.SBTTL ADD UNIT SECTION

\*\*\*  
: THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES  
: TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK  
: TO THE TEST CYCLE.  
:-

BGNAU

( \$AU)::

\*\*\*\*\*  
: INSERT ADD CODE HERE. THIS CODE WILL BE EXECUTED AFTER  
: AN "ADD" COMMAND. THE PURPOSE OF THIS CODE IS TO DO ANY  
: HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.  
: THIS SECTION IS OPTIONAL.  
\*\*\*\*\*

EXIT AU

.WORD J\$JMP  
.WORD L10014-2 .

\*\*\*\*\*  
: INSERT LOCAL STORAGE THAT IS USED ONLY  
: DURING THE ADD-UNIT SECTION.  
\*\*\*\*\*

\*\*\*\*\*  
: INSERT MESSAGES THAT ARE USED ONLY  
: DURING THE ADD-UNIT SECTION.  
\*\*\*\*\*

.EVEN

ENDAU

L10014: TRAP C\$AU

TEST 1:

5944  
5945  
5947  
5948  
5949  
5950  
5952  
5953  
5954  
5955  
5956  
5957  
5959  
5960  
5961  
5962  
5964  
5966  
5967  
5968  
5970  
5971  
5972  
5974  
5975  
5976

103544  
103544

.SBTTL TEST 1:

```

;*****
; APPEND THE NAME OF THIS TEST TO THE .SBTTL, AS SHOWN IN THE
; FOLLOWING EXAMPLE. .SBTTL TEST 1: NAME OF TEST
;*****

; **
; TEST TO ...
; -

;*****
; CHANGE THE PHRASE "TEST TO ..." TO BE A FUNCTIONAL
; DESCRIPTION OF THE HARDWARE TEST WHICH FOLLOWS.
;*****

;*****
; INSERT PROGRAM EQUATES THAT ARE USED ONLY IN THIS TEST.
;*****

BGNTST

T1::

;*****
; INSERT THE CODING FOR THIS HARDWARE TEST.
;*****

```

5979  
5980  
5981  
5982  
5983  
5984  
5985  
5986  
5987  
5988  
5989  
5990  
5991  
5992  
5993  
5994  
5995  
5996  
5997  
5998  
5999  
6000  
6001  
6002  
6003  
6004  
6005  
6006  
6007  
6008  
6009  
6010  
6011  
6012  
6013  
6014  
6015  
6016  
6017  
6018  
6019  
6020  
6021  
6022  
6023  
6024  
103544  
103554  
103554  
103556  
103560  
103562  
103564  
103566  
103570  
103572  
103574  
6025  
6026

103544  
105037 003221  
103550 105037 003220  
104443  
000406  
002202  
000142  
052104  
000000  
000001  
000110  
103574 012737 002202 003204  
103602 012737 050516 003206

```
.SBTTL          GETCL  COMMAND LINE FETCH & INTERPRETATION SECTION
: **
: FUNCTIONAL DESCRIPTION:
:
:   THIS PART OF TEST 1 IS USED TO PROMPT THE USER FOR A COMMAND
:
: CALLING SEQUENCE
:
:   ENTERED VIA DSRS TEST DISPATCH ROUTINE
:
: INPUT
:
:   NONE
:
: IMPLICIT INPUT
:
:   NONE
:
: OUTPUT
:
:   A COMMAND FROM THE USER
:
: IMPLICIT OUTPUT
:
:   NONE
:
: COMPLETION CODES
:
:   NONE
:
: SIDE EFFECTS
:
:   NONE
:
: REGISTERS USED
:
:   NONE
:
: DEBUG
:
:   NONE
: -
```

```
GETCL:
  CLRB  P$GDBD          ;CLEAR CMD LINE PARSING ERROR FLAG
  CLRB  P$NNUF
  GMANID CLI$PM,CMDBUF,A,0,1,72.,NO ;GET CMD LINE FROM OPERATOR
                                TRAP  C$GMAN
                                BR     10000$
                                .WORD  CMDBUF
                                .WORD  T$CODE
                                .WORD  CLI$PM
                                .WORD  0
                                .WORD  T$LQIM
                                .WORD  T$HILIM
                                10000$:
```

```
MOV  #CMDBUF,P$BUFA
MOV  #CLITRE,P$TREE
```

6027 103610 012737 103726 003210  
 6028 103616 005037 003734  
 6029 103622 004737 100544  
 6030 103626 105737 003221  
 6031 103632 001412  
 6032 103634

MOV @CLIACT,P\$ACT  
 CLR CFLAG  
 JSR PC,P\$TRV  
 TSTB P\$GDBD  
 BEQ 10\$  
 PRINTF @CLIERM

;CLEAR QUALIFIER FLAG  
 ;GO PARSE COMMAND TREE  
 ;SEE IF PARSED OK, OR AN ERROR

103634 012746 052113  
 103640 012746 000001  
 103644 010600  
 103646 104417

;IF NOT PRINT ERROR MESSAGE  
 MOV @CLIERM, (SP)  
 MOV @1, (SP)  
 MOV SP,RO  
 TRAP C\$PNTF  
 ADD @4,SP

6033 103654 000137 103544  
 6034 103660 105737 003220  
 6035 103664 001412  
 6036 103666

10\$:

JMP GETCL  
 TSTB P\$NNUF  
 BEQ 20\$  
 PRINTF @CLINUF

;SEE IF INCOMPLETE COMMAND TYPED

103666 012746 052144  
 103672 012746 000001  
 103676 010600  
 103700 104417

;IF NOT PRINT ERROR MESSAGE  
 MOV @CLINUF, (SP)  
 MOV @1, (SP)  
 MOV SP,RC  
 TRAP C\$PNTF  
 ADD @4,SP

6037 103706 000137 103544  
 6038 103712 022737 000020 003734 20\$:  
 6039 103720 001311

20\$:

JMP GETCL  
 CMP @CEXIT,CFLAG  
 BNE GETCL  
 EXIT TST

;WAS EXIT COMMAND TYPED?  
 ; IF NOT GET NEW COMMAND LINE  
 ; ELSE EXIT

6040 103722 104432  
 103724 013164

TRAP C\$EXIT  
 .WORD L10015 .

|||

```

6042 .SBTTL          CLI ACTION TABLE AND ROUTINES
6043 :
6044 : **
6045 : FUNCTIONAL DESCRIPTION:
6046 :
6047 :     THESE ARE THE ACTION TABLES TO DEFING WHAT TO DO WITH A COMMAND
6048 :     RECEIVED FROM A USER
6049 :     USER MUST CLEAR/SET P$GDBD IF USE "CLIBIF" IN CONNECTION WITH ACTION
6050 :
6051 : CALLING SEQUENCE
6052 :
6053 :     JMP     CL
6054 :
6055 : INPUT
6056 :
6057 :     COMMAND FROM THE USER
6058 :
6059 : IMPLICIT INPUT
6060 :
6061 :     NONE
6062 :
6063 : OUTPUT
6064 :
6065 :     NONE
6066 :
6067 : IMPLICIT OUTPUT
6068 :
6069 :     NONE
6070 :
6071 : COMPLETION CODES
6072 :
6073 :     NONE
6074 :
6075 : SIDE EFFECTS
6076 :
6077 :     NONE
6078 :
6079 : REGISTERS USED
6080 :
6081 :     R2 - HOLDS ACTION CODE FROM THE PARSING (CLI) NODE
6082 :
6083 : DEBUG
6084 :
6085 :     NONE
6086 :
6087 : --
6087 103726          CLIACT:
6088 103726 006302          ASL      R2          ;MULTIPLY ACTION CODE BY 2
6089 103730 016202 103744  MOV     10$(R2),R2      ;OFFSET VALUE
6090 103734 062702 103744  ADD     @10$,R2        ;ADD BASE VALUE
6091 103740 004712          JSR     PC,(R2)        ;GO DO ACTION
6092 103742 000207          RTS     PC          ;RETURN TO TRVACT
6093
6094
6095 103744 000124          10$:   .WORD  ACTNUL-10$      ;BRIEF DESCRIPTION OF ACTION TAKEN
6096 103746 000126          .WORD  ACTHLP-10$      ;0-NULL
6097 103750 000164          .WORD  ACTNOD-10$     ;1-HELP
6098 103752 000474          .WORD  ACTBLD 10$     ;2-NODE
6099                                     ;3 BUILD

```

6099	103754	003520	.WORD	ACTRUN-10\$	;4-RUN SPECIFIED TEST
6100	103756	006366	.WORD	ACTPAT-10\$	;5-SET 'MESSAGE PATTERN' TEST FLAG
6101	103760	010172	.WORD	ACTSAV-10\$	;6-SAVE NODE TABLE
6102	103762	001224	.WORD	ACTSUM-10\$	;7-PRINT SUMMARY TABLE
6103	103764	001606	.WORD	ACTIDT-10\$	;10-REQUEST ID
6104	103766	002632	.WORD	ACTEXT-10\$	;11-EXIT
6105	103770	000116	.WORD	ACTNUF-10\$	;12-NOT ENOUGH INFO
6106	103772	002642	.WORD	ACTXAD-10\$	;13-EXTRACT NI NODE ADDRESS FROM INPUT LINE
6107	103774	002724	.WORD	ACTSR4-10\$	;14-SAVE POINTER TO BEGINING OF ADDRESS STRING
6108	103776	007732	.WORD	ACTSND-10\$	;15-SET 'NODE' FLAG FOR SHOW COMMAND
6109	104000	002732	.WORD	ACTALP-10\$	;16-SET 'ALPHA' FLAG
6110	104002	002742	.WORD	ACTONE-10\$	;17-SET 'ONES' FLAG
6111	104004	002752	.WORD	ACTZRO-10\$	;20-SET 'ZEROS' FLAG
6112	104006	002762	.WORD	ACTIAL-10\$	;21-SET 'IALT' FLAG
6113	104010	002772	.WORD	ACTOAL-10\$	;22-SET 'OALT' FLAG
6114	104012	003002	.WORD	ACTCTT-10\$	;23-SET 'CCITT' FLAG
6115	104014	003012	.WORD	ACTOPR-10\$	;24-SET 'OPER SEL' FLAG
6116	104016	003140	.WORD	ACTTYP-10\$	;25-DETERMINE MESSAGE TYPE
6117	104020	003146	.WORD	ACTSZE-10\$	;26-DETERMINE MESSAGE SIZE
6118	104022	003224	.WORD	ACTCPY-10\$	;27-DETERMINE MESSAGE COPIES
6119	104024	003302	.WORD	ACTNAD-10\$	;30-SET 'NODE/ADDRESS' FLAG
6120	104026	003440	.WORD	ACTNAL-10\$	;31-SET 'NODE/ALL' FLAG
6121	104030	003646	.WORD	ACTRNA-10\$	;32-SET 'ALL' FLAG FOR RUN COMMAND
6122	104032	004724	.WORD	ACTRNL-10\$	;33-SET 'LOOPPAIR' FLAG FOR RUN CMD
6123	104034	006450	.WORD	ACTSMS-10\$	;34-SHOW CURRENT MESSAGE PARAMETERS
6124	104036	006542	.WORD	ACTCMS-10\$	;35-RESET MESSAGE PARAMETERS TO DEFAULT
6125	104040	006646	.WORD	ACTCNT-10\$	;36-SET 'COUNTER' FLAG FOR SHOW COMMAND
6126	104042	010106	.WORD	ACTCNL-10\$	;37-CLEAR LOGICAL NODE NAMED FROM TABLE
6127	104044	000124	.WORD	ACTNUL-10\$	;40 - DO NOTHING
6128	104046	010236	.WORD	ACTUNS-10\$	;41-UNSAVE NODE TABLE
6129	104050	010340	.WORD	ACTCSU-10\$	;42-CLEAR SUMMARY TABLE
6130	104052	004272	.WORD	ACTDIR-10\$	;43-SET 'LOOP DIRECT' FLAG FOR RUN COMMAND
6131	104054	010414	.WORD	ACTDFT-10\$	;44-LOOK FOR PASS COUNT DEFAULT
6132	104056	010462	.WORD	ACTUSF-10\$	;45-UNSAVE NODE TABLE FROM A FILE
6133	104060	010744	.WORD	ACTRSP-10\$	;46 RESPONDER MODF

```

6135 ;
6136 ;ACTION ROUTINE TO INDICATE THAT NOT ENOUGH COMMAND
6137 ;INFORMATION HAS BEEN ENTERED
6138 ;
6139 ;
6140 ACTNUF:
6141 104062 112737 177777 003220 MOVB # 1,P$NNUF ;SET FLAG TO SAY NEED MORE OF COMMAND
6142 ;
6143 ;
6144 ;ACTION ROUTINE TO DO NOTHING
6145 ;
6146 ;
6147 ACTNUL:
6148 104070 000207 RTS PC ;RETURN TO PARSER
6149 ;
6150 ;
6151 ;ACTION ROUTINE TO PRINT OUT HELP FILE
6152 ;
6153 ;
6154 ;
6155 ACTHLP:
6156 104072 P$PUSH R2 ;SAVE R2
        MOV R2,(R5)+
6157 104072 010225 ;MOVE POINTER TO BEGINING OF HELP FILE
        MOV #HLPTAB,R2
6158 104074 012702 003224 10$: PRINTF (R2)+ ;PRINT LINE AND INCREMENT POINTER
        MOV (R2)+,-(SP)
        MOV #1,-(SP)
        MOV SP,R0
        TRAP C$PNTF
        ADD #4,SP
6159 104116 020227 003322 CMP R2,#HLPEND ;SEE IF ENTIRE FILE PRINTED
6160 104122 001366 BNE 10$ ;IF NOT, PRINT MORE
6161 104124 P$POP R2 ;ELSE, RESTORE R2 AND RETURN
        MOV -(R5),R2
6162 104126 000207 RTS PC
6163 ;
6164 ;ACTION ROUTINE TO READ IN NODE PHY. ADDRESS, STORE IT IN ADRBUF
6165 ;AND ENTER IT INTO THE NODE TABLE
6166 ;
6167 ;
6168 ;
6169 ACTNOD:
6170 104130 105037 003220 CLRB P$NNUF ;CLEAR NOTNUF FLAG
6171 104134 004737 101656 JSR PC,TRVADR ;TRAVERSE ADDRESS, CHECK IF TARGET OR ASSIST
6172 104140 105737 003221 TSTB P$GDBD ;CHECK IF RESULTS OK
6173 104144 001134 BNE 60$ ;IF NOT, RETURN WITH -1 IN P$GDBD
6174 104146 10$: CALL EDPACK CBOADR,#ADRBUF,#6 ;GET ADDRESS INTO BUFFER
        MOV #6,(R5)+
        MOV #ADRBUF,(R5)+
        MOV CBOADR,(R5)+
        JSR R4,PREG14
        .WORD EDPACK-ANCHOR
6175 104170 P$POP R1 ;CHECK RESULTS FOR NUMBER OF CHAR.S
        MOV -(R5),R1
6176 104172 001411 BEQ 20$ ;IF OK, BRANCH TO 20$
6177 104174 PRINTF #CADRER ;ELSE PRINT ERROR MESSAGE
        MOV #CADRER,(SP)
104174 012746 052746

```





```

6201
6202 ; ACTION ROUTINE TO BUILD NODE TABLE USING THE PERIODIC ADDRESS
6203 ; IDENTIFICATION MESSAGES SENT OUT BY NODE ON THE NI
6204
6205
6206 104440 ACTBLD:
6207 104440 PRINTS #MSG1 ; PRINT BUILD COMMAND MESSAGE
        MOV #MSG1, (SP)
        MOV #1, (SP)
        MOV SP, R0
        TRAP C:PNTS
        ADD #4, SP
        104440 012746 054415
        104444 012746 000001
        104450 010600
        104452 104416
        6208 104454 062706 000004
        104460 PRINTS #MSG11
        MOV #MSG11, (SP)
        MOV #1, -(SP)
        MOV SP, R0
        TRAP C:PNTS
        ADD #4, SP
        104460 012746 054530
        104464 012746 000001
        104470 010600
        104472 104416
        104474 062706 000004
        6209 104500 PRINTS #MSG12
        MOV #MSG12, (SP)
        MOV #1, (SP)
        MOV SP, R0
        TRAP C:PNTS
        ADD #4, SP
        104500 012746 054643
        104504 012746 000001
        104510 010600
        104512 104416
        104514 062706 000004
        6210 104520 P#PUSH R2,R3,R4 ; SAVE REGISTERS
        MOV R4, (R5)
        MOV R3, (R5)
        MOV R2, (R5)
        104520 010425
        104522 010325
        104524 010225
        6211 104526 CALL SETUP #ADMUL ; WRITE MULTICAST ADDRESS
        MOV #ADMUL, (R5)
        JSR R4, PREG14
        .WORD SETUP-ANCHOR
        104526 012725 000001
        104532 004437 070656
        104536 001232
        6212 104540 P#POP R2 ; CHECK FOR ERROR
        MOV -(R5), R2
        104540 014502
        6213 104542 BEQ 10$ ; IF OK, CONTINUE
        6214 104544 ERRHRD 25,EMSG25 ; ELSE REPORT ERROR
        TRAP C:ERRHRD
        .WORD 25
        .WORD EMSG25
        .WORD 0
        104544 104456
        104546 000031
        104550 064157
        104552 000000
        6215 104554 000137 105154
        6216 104560
        6217 104560 005237 002200 10$: JMP 100$ ; LEAVE
        6218 104564 005037 050314
        6219 104570 005037 050316
        6220 104574 005037 050320
        6221 104600 012737 000013 050322
        6222 104606 012737 002412 002410
        6223 104614 012704 003762
        6224 104620
        6225 104620 012714 000074 20$: MOV #60., (R4)
        6226 104624
        6227 104624 30$: BREAK ; ALLOW FOR CONTROL C INTERRUPTION
        104624 104422
        6228 104626 005714
        6229 104630 001503
        6230 104632
        TST (R4) ; SEE IF INTERVAL IS UP
        BEQ 70$ ; IF YES, BRANCH
        CALL RECEVE ; ELSE, CHECK FOR RECEPTION OF IL MESSAGE
  
```

Address	Instruction	Comments	Labels
104632	004437 070656		
104636	003044		
6231 104640	P\$POP R2	; R2 HOLDS NO OF MESSAGES RECEIVED	
104640		JSR R4,PREG14	
104640		.WORD RECEIVE-ANCHOR	
6232 104642	BEQ 30\$	; IF NONE, KEEP LOOKING	
6233 104644	MOV RRGXNT,R3	; IF ONE, GET RECEIVE RING POINTER	
6234 104650	CALL GETRNX #1,#RRGXNT	; UPDATE POINTER, VALIDATE PREVIOUS DESCRIPTOR	
104650		MOV #RRGXNT,(R5)	
104654		MOV #1,(R5)	
104660		JSR R4,PREG14	
104664		.WORD GETRNX ANCHOR	
6235 104666	MOV LOADD(R3),R3	; POINT R3 TO MESSAGE BUFFER	
6236 104672	ADD #SOURCC,R3	; POINT R3 TO NODE ADDRESS	
6237 104676	MOV #NODTBL,R2	; POINT R2 TO NODE TABLE	
6238 104702			
6239 104702	40\$: CALL CMPADR R2,R3	; SEE IF NODE ALREADY ON TABLE	
104702		MOV R3,(R5)	
104704		MOV R2,(R5)	
104706		JSR R4,PREG14	
104712		.WORD CMPADR ANCHOR	
6240 104714	P\$POP R1	; GET RESULTS OF COMPARE	
104714		MOV (R5),R1	
6241 104716	BEQ 30\$	; DON'T ADD TO TABLE IF ALREADY THERE	
6242 104720			
6243 104720	50\$: ADD #10,R2	; ELSE CHECK NEXT TABLE ENTRY	
6244 104724	CMP #1,(R2)	; SEE IF AT END OF TABLE	
6245 104730	BNE 40\$	; IF NO, COMPARE NEXT ENTRY	
6246 104732	CALL FINDSL	; IF YES, GET NEXT TABLE SLOT	
104732		JSR R4,PREG14	
104736		.WORD FINDSL ANCHOR	
6247 104740	P\$POP R2	; SEE IF TABLE FULL	
104740		MOV (R5),R2	
6248 104742	BNE 60\$	; IF YES, BRANCH	
6249 104744	MOV SLOT,R2	; IF NO, ADD NODE TO TABLE	
6250 104750	MOV (R3)+,(R2)		
6251 104752	MOV (R3)+,(R2)		
6252 104754	MOV (R3),(R2)	; SIX BYTES WORTH	
6253 104756	MOVB TEMP1,3(R2)	; SET NODE TYPE (TARGET OR ASSIST)	
6254 104764	COMB TEMP1	; SWITCH TYPE FOR NEXT TIME	
6255 104770	BICB #376,TEMP1		
6256 104776	INC TEMP	; INCREMENT 'NODES IN LAST MIN. COUNTER	
6257 105002	MOV #13,TEMP3	; RESET 'MINS. SINCE LAST NODE' COUNTER	
6258 105010	BR 30\$		
6259 105012	60\$: PRINTB #TABFUL,#NOD	; PRINT 'TABLE FULL' MESSAGE	
6260 105012		MOV #NOD,(SP)	
105016		MOV #TABFUL,(SP)	
105022		MOV #2,(SP)	
105026		MOV SP,R0	
105030		TRAP C\$PNTB	
105032		ADD #6,SP	
6261 105036	70\$: BR 80\$		
6262 105040			
6263 105040	DEC TEMP3	; SEE IF 10 MINS SINCE LAST NODE	
6264 105044	BEQ 80\$	; IF YES, EXIT	
6265 105046	INC TEMP2	; SEE IF TIME IS UP	
6266 105052	CMP TEMP2,#40		

```

6267 105060 001417          BEQ      80$          ; IF YES, EXIT
6268 105062          PRINTS  @BLDMSG,TEMP,TEMP2 ; ELSE, PRINT "STILL WORKING MESSAGE
                                MOV      TEMP2,(SP)
                                MOV      TEMP,(SP)
                                MOV      @BLDMSG,(SP)
                                MOV      @3,(SP)
                                MOV      SP,R0
                                TRAP     C$PNTS
                                ADD      @10,SP
6269 105112 005037 050314    CLR      TEMP
6270 105116 000640          BR       20$
6271 105120          80$:
6272 105120          CALL     SETUP @KILMUL      ; KILL THE MULTICAST ADDRESS
                                MOV      @KILMUL,(R5)
                                JSR      R4,PREG14
                                .WORD   SETUP ANCHOR
6273 105132          P$POP   R2          ; CHECK FOR ERROR
                                MOV      -(R5),R2
6274 105134 014502          BEQ      90$          ; IF OK CONTINUE
6275 105136          ERRHRD  34,EMSG25    ; ELSE, REPORT ERROR
                                TRAP     C$ERHRD
                                .WORD   34
                                .WORD   EMSG25
                                .WORD   0
6276 105146 000402          90$: BR       100$          ; AND LEAVE
6277 105150          100$:
6278 105150 004737 113676    JSR      PC,ACTSND    ; PRINT NODE TABLE
6279 105154          100$:
6280 105154 005037 002200    CLR      BLDLFG
6281 105160          P$POP   R2,R3,R4    ; INDICATE THAT WE ARE NO LONGER BUILDING
                                ; RESTORE REGISTERS
                                MOV      -(R5),R2
                                MOV      -(R5),R3
                                MOV      (R5),R4
6282 105166 000207          RTS      PC
6283
6284 ;ACTION ROUTINE TO PRINT OUT THE SUMMARY DATA
6285 ;
6286 105170 105037 003220    ACTSUM: CLRB   P$NNUF          ;CLEAR NOTNUF FLAG
6287 105174          P$PUSH  R2,R3,R4
                                MOV      R4,(R5)
                                MOV      R3,(R5)
                                MOV      R2,(R5)
6288 105202 012701 002714    MOV      @STATBL,R1    ; MOVE ADDRESS OF TABLE TO R1
6289 105206 005711          TST      (R1)          ; SEE IF TABLE EMPTY
6290 105210 001013          BNE     10$          ; IF NOT, CONT.
6291 105212          PRINTF  @TABFMT,@SUMM ; ELSE PRINT 'TABLE EMPTY' MESSAGE
                                MOV      @SUMM,(SP)
                                MOV      @TABEMT,(SP)
                                MOV      @2,(SP)
                                MOV      SP,R0
                                TRAP     C$PNTF
                                ADD      @6,SP
6292 105236 000541          10$: BR       20$          ; EXIT
6293 105240 021127 177777    CMP      (R1),@-1    ; SEE IF AT END OF TABLE
6294 105244 001536          BEQ     20$          ; IF YES, EXIT
6295 105246 005711          TST      (R1)          ; SEE IF REST OF TABLE EMPTY
6296 105250 001534          BEQ     20$          ; IF YES, EXIT

```

6297	105252			CALL	BINHEX R1,06,0STRBUF	:	PRINT SUMMARY DATA	
	105252	012725	002324				MOV	0STRBUF,(R5).
	105256	012725	000006				MOV	06,(R5).
	105262	010125					MOV	R1,(R5).
	105264	004437	070656				JSR	R4,PREG14
	105270	005150					.WORD	BINHEX ANCHOR
6298	105272			PRINTF	0SUMMS1,0STRBUF	:	NODE ADDRESS	
	105272	012746	002324				MOV	0STRBUF,(SP)
	105276	012746	070043				MOV	0SUMMS1,(SP)
	105302	012746	000002				MOV	02,-(SP)
	105306	010600					MOV	SP,R0
	105310	104417					TRAP	C0PNTF
	105312	062706	000006				ADD	06,SP
6299	105316			PRINTF	0SUMMS2	:	FIRST HEADER	
	105316	012746	070064				MOV	0SUMMS2,(SP)
	105322	012746	000001				MOV	01,-(SP)
	105326	010600					MOV	SP,R0
	105330	104417					TRAP	C0PNTF
	105332	062706	000004				ADD	04,SP
6300	105336	012702	000026	MOV	026,R2	:	GET STATISTICS SIZE FOR THIS NODE	
6301	105342	012703	002664	MOV	0STATBF,R3	:	GET BUFFER ADDRESS FOR STATISTICS	
6302	105346							
6303	105346	112123		MOVB	(R1),.(R3).	:	BUFFER THE STATISTICS	
6304	105350	005302		DEC	R2	:	HAVE ALL BYTES FOR NODE BEEN XFERRED?	
6305	105352	001375		BNE	15\$	:	BRANCH IF NOT	
6306	105354	013702	002672	MOV	STATBF+6,R2	:	RX NOT COMPLETE	
6307	105360	013703	002674	MOV	STATBF+10,R3	:	RX COMPLETE	
6308	105364	013704	002676	MOV	STATBF+12,R4	:	LENGTH ERRORS	
6309	105370			PRINTF	0SUMMS3,R2,R3,R4	:	PRINT THEM OUT	
	105370	010446					MOV	R4,(SP)
	105372	010346					MOV	R3,(SP)
	105374	010246					MOV	R2,(SP)
	105376	012746	070150				MOV	0SUMMS3,(SP)
	105402	012746	000004				MOV	04,(SP)
	105406	010600					MOV	SP,R0
	105410	104417					TRAP	C0PNTF
	105412	062706	000012				ADD	012,SP
6310	105416			PRINTF	0SUMMS4	:	SECOND HEADER	
	105416	012746	070177				MOV	0SUMMS4,(SP)
	105422	012746	000001				MOV	01,(SP)
	105426	010600					MOV	SP,R0
	105430	104417					TRAP	C0PNTF
	105432	062706	000004				ADD	04,SP
6311	105436	013702	002700	MOV	STATBF+14,R2	:	COMPARE ERRORS	
6312	105442			CALL	BINDEC 0STATBF+16	:	PUT BYTES COMPARED INTO ASCII STRING	
	105442	012725	002702				MOV	0STATBF+16,(R5).
	105446	004437	070656				JSR	R4,PREG14
	105452	007446					.WORD	BINDEC ANCHOR
6313	105454			PRINTF	0SUMMS5,R2,0DECSTR	:	PRINT THEM OUT	
	105454	012746	100526				MOV	0DECSTR,-(SP)
	105460	010246					MOV	R2,-(SP)
	105462	012746	070262				MOV	0SUMMS5,(SP)
	105466	012746	000003				MOV	03,-(SP)
	105472	010600					MOV	SP,R0
	105474	104417					TRAP	C0PNTF
	105476	062706	000010				ADD	010,SP
6314	105502			CALL	BINDEC 0STATBF+22	:	PUT BYTES TRANSFERED INTO ASCII STRING	

15\$:

```

105502 012725 002706
105506 004437 070656
105512 007446
6315 105514          PRINTF  @SUMMS6,@DECSTR          ; PRINT
105514 012746 100526
105520 012746 070300
105524 012746 000002
105530 010600
105532 104417
105534 062706 000006
6316 105540 000637
6317 105542          BR      10$          ; PLAY IT AGAIN, SAM
105542 014502          P$POP  R2,R3,R4
105544 014503
105546 014504
6318 105550          RTS      PC
6319
6320 ; ACTION ROUTINE TO INITIATE THE REQUEST ID TEST TO THE SPECIFIED NODE
6321 ;
6322 ; --*
6323 ; FUNCTIONAL DESCRIPTION
6324 ; THIS SUBROUTINE BUILDS AND TRANSMITS REQUEST ID PACKETS
6325 ; TO THE NODE SPECIFIED BY THE OPERATOR IN THE COMMAND LINE.
6326 ; THE SYSTEM ID INFO OF THE SPECIFIED NODE IS THEN DISPLAYED.
6327 ; IF THE NODE DOES NOT RESPOND BEFORE 60 SECONDS HAVE PASSED
6328 ; AN ERROR IS REPORTED TO THE OPERATOR.
6329 ;
6330 ; INPUTS - IMPLICIT - THE SPECIFIED NODE ADDRESS IS LOCATED IN ADRBUF.
6331 ;
6332 ; OUTPUTS - SYSTEM ID INFO OR ERROR MESSAGE PRINTED TO OPERATOR.
6333 ;
6334 ; CALLING PROCEDURE - JSR PC, ACTIDT
6335 ;
6336 ; SIDE EFFECTS - XRGXNT POINTER IS UPDATED BY A CALL TO BLDREQ SUB.
6337 ;
6338 ; REGISTER USAGE - R1 POINTS TO $WOMO FOR WRITE MODE OPERATIONS.
6339 ; R2 IS SCRATCH.
6340 ; R3 POINTS TO THE RECEIVED MESSAGE BUFFER.
6341 ; R4 POINTS TO TIMEOUT TIMER
6342 ;
6343 ; --*
6344 ACTIDT:
6345 105552 105737 003222          TSTB   P$AERP          ;SEE IF ADDRESS ENTERED WAS VALID
6346 105556 001402          BEQ    10$          ; IF NOT, EXIT ACTION ROUTINE
6347 105560 000137 106466          JMP    130$
6348 105564
6349 105564 105037 003220          CLRB   P$NNUF          ;CLEAR NOTNUF FLAG
6350 105570          CALL  CMPADR @ADRBUF,@ILLADR ; SEE IF ILLEGAL ADDRESS
105570 012725 002656          MOV    @ILLADR,(R5)
105574 012725 002316          MOV    @ADRBUF,(R5)
105600 004437 070656          JSR    R4,PREG14
105604 026004          .WORD CMPADR ANCHOR
6351 105606          P$POP  R1
105606 014501          MOV    -(R5),R1
6352 105610 001012          BNE   20$          ; IF NO, CONTINUE
6353 105612
6354 105612          PRINTF @ILADMS          ; ELSE PRINT ILLEGAL ADDRESS MESSAGE

```

	105612	012746	052565						MOV	#ILADMS, (SP)
	105616	012746	000001						MOV	#1, (SP)
	105622	010600							MOV	SP,RO
	105624	104417							TRAP	C\$PNTF
	105626	062706	000004						ADD	#4,SP
6355	105632	000137	106466							
6356	105636			20\$:	JMP	130\$				
6357	105636				P\$PUSH	R1,R2,R3,R4				; SAVE REGISTERS
	105636	010425							MOV	R4,(R5).
	105640	010325							MOV	R3,(R5).
	105642	010225							MOV	R2,(R5).
	105644	010125							MOV	R1,(R5).
6358	105646				CALL	CMPADR #ADRBUF,#PHYADR				; SEE IF ADDRESS IS OWN (HOST NODE)
	105646	012725	004010						MOV	#PHYADR,(R5).
	105652	012725	002316						MOV	#ADRBUF,(R5).
	105656	004437	070656						JSR	R4,PREG14
	105662	026004							.WORD	CMPADR ANCHOR
6359	105664				P\$POP	R1				
	105664	014501							MOV	(R5),R1
6360	105665	001006			BNE	25\$				
6361	105670				ERRHRD	64,MSG64				; REPORT THAT WE CAN T USE OUR OWN ADDRESS
	105670	104456							TRAP	C\$ERRRD
	105672	000100							.WORD	64
	105674	066615							.WORD	MSG64
	105676	000000							.WORD	0
6362	105700	000137	106456		JMP	120\$				; RESTORE REGISTERS AND EXIT
6363	105704			25\$:	MOV	#-2,TEMP2				; SET COUNTER FOR NO. OF TIMES TRIED
6364	105704	012737	177776	050320	CALL	BLDREQ				; BUILD REQUEST ID MESSAGE PACKET
6365	105712								JSR	R4,PREG14
	105712	004437	070656						.WORD	BLDREQ-ANCHOR
	105716	006330								
6366	105720				CALL	XMIT #XMTDAT				; TRANSMIT REQUEST
	105720	012725	000001						MOV	#XMTDAT,(R5).
	105724	004437	070656						JSR	R4,PREG14
	105730	002326							.WORD	XMIT-ANCHOR
6367	105732				P\$POP	R2				; GET RESULTS. R2 = SUCCESS/FAILURE
	105732	014502							MOV	-(R5),R2
6368	105734	001402			BEQ	30\$				; IF OK BRANCH
6369	105736	000137	106430			90\$				; ELSE JUMP TO 90\$
6370	105742			30\$:	TST	RETRY5				; SEE IF FAILED DUE TO EXCESSIVE COLLISIONS
6371	105742	005737	050274							; IF NO, CONT.
6372	105746	001412			BEQ	40\$				; YES, PRINT 'EXCESSIVE COLLISIONS' MESSAGE
6373	105750				PRINTF	#RTRYER				
	105750	012746	052411						MOV	#RTRYER, (SP)
	105754	012746	000001						MOV	#1, (SP)
	105760	010600							MOV	SP,RO
	105762	104417							TRAP	C\$PNTF
	105764	062706	000004						ADD	#4,SP
6374	105770	000137	106456		JMP	120\$				; EXIT
6375	105774			40\$:	MOV	#TIMERS,R4				; SET UP FOR 10 SECOND TIMEOUT
6376	105774	012704	003762		MOV	#10.,(R4)				
6377	106000	012714	000012							
6378	106004			50\$:	BREAK					
6379	106004	104422							TRAP	C\$BRK
6380	106006	005714			TST	(R4)				; SEE IF TIME HAS EXPIRED
6381	106010	001427			BEQ	60\$				; IF YES, BRANCH

6382	106012			CALL	RECEVE				; CHECK FOR ANSWER
	106012	004437	070656						JSR R4,PREG14
	106016	003044							.WORD RECEIVE-ANCHOR
6383	106020			P\$POP	R2				; R2 HOLDS NO. OF BUFFERS RECEIVED
	106020	014502							MOV -(R5),R2
6384	106022	001770		BEQ	50\$				; IF NO BUFFERS RECEIVED, LOOP
6385	106024	013703	024226	MOV	RRGNXT,R3				; GET RECEIVE RING POINTER
6386	106030	016303	000004	MOV	LOADD(R3),R3				; POINT R3 TO MESSAGE BUFFER
6387	106034	026327	000022	051115	CMP	SIRCPT(R3),@MR			; SEE IF MESSAGE RECEIVED IS IN REPLY TO ONE SENT
6388	106042	001426		BEQ	70\$				; IF YES, BRANCH TO 70\$
6389	106044	005237	050320	INC	TEMP2				; INCREMENT RETRY COUNTER
6390	106050			CALL	GETRNX @1,@RRGNXT				; THROW OUT THE PACKET
	106050	012725	024226						MOV @RRGNXT,(R5)+
	106054	012725	000001						MOV @1,(R5)+
	106060	004437	070656						JSR R4,PREG14
	106064	006450							.WORD GETRNX-ANCHOR
6391	106066	001346		BNE	50\$				; IF NO, LOOK FOR CORRECT REPLY MESSAGE
6392	106070								
6393	106070			PRINTF	@MSG22				; ELSE, REPORT ERROR
	106070	012746	064050						MOV @MSG22,(SP)
	106074	012746	000001						MOV @1,(SP)
	106100	010600							MOV SP,R0
	106102	104417							TRAP C\$PNTF
	106104	062706	000004						ADD @4,SP
6394	106110	005237	050032	INC	S.NREC				; UPDATE SUMMARY DATA
6395	106114	000137	106440	JMP	100\$				; AND EXIT
6396	106120								
6397	106120	005237	050030	70\$:	INC	S.REC			; INCREMENT RECEIVED MESSAGES COUNTER
6398	106124	062737	000056	050042	ADD	@46,S.XFER			; UPDATE BYTES TRANSFERED COUNTER
6399	106132	010302			MOV	R3,R2			; PUT POINTER INTO R2
6400	106134	062702	000042		ADD	@SIADDR,R2			; POINT R2 TO ADDRESS
6401	106140			80\$:					
6402	106140			CALL	BINHEX R2,@6,@STRBUF				; PUT ADDRESS INTO STRBUF
	106140	012725	002324						MOV @STRBUF,(R5)+
	106144	012725	000006						MOV @6,(R5)+
	106150	010225							MOV R2,(R5)+
	106152	004437	070656						JSR R4,PREG14
	106156	005150							.WORD BINHEX ANCHOR
6403	106160			PRINTF	@SIMSG1,@STRBUF				; PRINT REMOTE NODE ADDRESS
	106160	012746	002324						MOV @STRBUF,(SP)
	106164	012746	067266						MOV @SIMSG1,(SP)
	106170	012746	000002						MOV @2,(SP)
	106174	010600							MOV SP,R0
	106176	104417							TRAP C\$PNTF
	106200	062706	000006						ADD @6,SP
6404	106204	016302	000022	MOV	SIRCPT(R3),R2				; GET RECEIPT NUMBER
6405	106210			PRINTF	@SIMSG2,R2				; PRINT RECEIPT NUMBER
	106210	010246							MOV R2,(SP)
	106212	012746	067325						MOV @SIMSG2,(SP)
	106216	012746	000002						MOV @2,(SP)
	106222	010600							MOV SP,R0
	106224	104417							TRAP C\$PNTF
	106226	062706	000006						ADD @6,SP
6406	106232	116302	000027	MOVB	SIVERS(R3),R2				; GET VERSION NO. AND PRINT
6407	106236			PRINTF	@SIMSG3,R2				
	106236	010246							MOV R2,(SP)
	106240	012746	067360						MOV @SIMSG3,(SP)

	106244	012746	000002				MOV	#2, -(SP)
	106250	010600					MOV	SP, R0
	106252	104417					TRAP	C#PNTF
	106254	062706	000006				ADD	#6, SP
6408	106260	116302	000030	MOVB	SIECO(R3),R2	; GET ECO NO. AND PRINT		
6409	106264			PRINTF	#SIMSG4,R2			
	106264	010246					MOV	R2, -(SP)
	106266	012746	067420				MOV	#SIMSG4, -(SP)
	106272	012746	000002				MOV	#2, (SP)
	106276	010600					MOV	SP, R0
	106300	104417					TRAP	C#PNTF
	106302	062706	000006				ADD	#6, SP
6410	106306	116302	000031	MOVB	SIUECO(R3),R2	; GET USER ECO NO. AND PRINT		
6411	106312			PRINTF	#SIMSG5,R2			
	106312	010246					MOV	R2, -(SP)
	106314	012746	067441				MOV	#SIMSG5, -(SP)
	106320	012746	000002				MOV	#2, (SP)
	106324	010600					MOV	SP, R0
	106326	104417					TRAP	C#PNTF
	106330	062706	000006				ADD	#6, SP
6412	106334	116302	000035	MOVB	SIFNCT(R3),R2	; GET FUNCTON CODE AND PRINT		
6413	106340			PRINTF	#SIMSG6,R2			
	106340	010246					MOV	R2, (SP)
	106342	012746	067467				MOV	#SIMSG6, (SP)
	106346	012746	000002				MOV	#2, -(SP)
	106352	010600					MOV	SP, R0
	106354	104417					TRAP	C#PNTF
	106356	062706	000006				ADD	#6, SP
6414	106362	116302	000053	MOVB	SIDEV(R3),R2	; GET DEVICE TYPE AND PRINT		
6415	106366			PRINTF	#SIMSG7,R2			
	106366	010246					MOV	R2, -(SP)
	106370	012746	067515				MOV	#SIMSG7, (SP)
	106374	012746	000002				MOV	#2, -(SP)
	106400	010600					MOV	SP, R0
	106402	104417					TRAP	C#PNTF
	106404	062706	000006				ADD	#6, SP
6416	106410			CALL	GETRNX #1, #RRGNXT	; UPDATE POINTER, VALIDATE PREVIOUS DESCRIPTOR		
	106410	012725	024226				MOV	#RRGNXT, (R5).
	106414	012725	000001				MOV	#1, (R5).
	106420	004437	070656				JSR	R4, PREG14
	106424	006450					.WORD	GETRNX ANCHOR
6417	106426	000404		BR	100\$	; EXIT		
6418	106430			90\$:				
6419	106430			ERRDF	24, EMSG24, ERR1	; ERROR CAN'T TRANSMIT PACKETS		
	106430	104455					TRAP	C#ERDF
	106432	000030					.WORD	24
	106434	064103					.WORD	EMSG24
	106436	070306					.WORD	ERR1
6420	106440			100\$:				
6421	106440			CALL	WRITES #1, #ADRBUF	; UPDATE SUMMARY TABLE		
	106440	012725	002316				MOV	#ADRBUF, (R5).
	106444	012725	000001				MOV	#1, (R5).
	106450	004437	070656				JSR	R4, PREG14
	106454	007176					.WORD	WRITES-ANCHOR
6422	106456			110\$:				
6423	106456			120\$:				
6424	106456			P#POP	R1,R2,R3,R4	; RESTORE REGISTERS		



106456	014501					MOV	(R5),R1
106460	014502					MOV	-(R5),R2
106462	014503					MOV	(R5),R3
106464	014504					MOV	-(R5),R4
6425	106466				130\$:		
6426	106466	000207				RTS	PC
6427							
6428							
6429							
6430							
6431	106470						
6432	106470	105714					
6433	106472	001401					
6434	106474	000437					
6435	106476						
6436	106476	012737	061644	002312	10\$:		
6437	106504	013701	002372			MOV	@CNDTY6,KEYWD1
6438	106510	006301				MOV	P\$TYPE,R1
6439	106512	062701	003324			ASL	R1
6440	106516					ADD	@MSGTAB,R1
	106516	012746	054345			PRINTF	@MSGPRM
	106522	012746	000001				
	106526	010600					
	106530	104417					
	106532	062706	000004				
6441	106536					PRINTF	@MSG4,(R1),P\$SIZE,P\$CPYS
	106536	013746	002376				
	106542	013746	002374				
	106546	011146					
	106550	012746	055002				
	106554	012746	000004				
	106560	010600					
	106562	104417					
	106564	062706	000012				
6442	106570	105037	003220			CLRB	P\$NNUF
6443	106574						
6444	106574	000207			20\$:		
6445						RTS	PC
6446							
6447							
6448	106576						
6449	106576	012737	000020	003734			
6450	106604	000207					
6451							
6452							
6453							
6454							
6455	106606						
6456	106606						
	106606	012725	000006			CALL	EDPACK CBOADR,@ADRBUF,@6
	106612	012725	002316				
	106616	013725	002370				
	106622	004437	070656				
	106626	004600					
6457	106630						
	106630	014537	003222				
6458	106634	105737	003222				

```

6459 106640 001412          BEQ      10$
6460 106642          PRINTF   @CADRER          ;ELSE, PRINT ERROR MESSAGE
      106642 012746 052746          MOV      @CADRER, -(SP)
      106646 012746 000001          MOV      #1, -(SP)
      106652 010600          MOV      SP, RO
      106654 104417          TRAP     C$PNTF
      106656 062706 000004          ADD      #4, SP
6461 106662 105037 003220          CLRB    P$NNUF          ; AND CLEAR 'NOT ENOUGH' FLAG
6462 106666          10$:      RTS      PC
6463 106666 000207          ;
6464          ;
6465          ;ACTION ROUTINE TO STORE POINTER TO BEGINING OF OPERATOR INPUT ADDRESS
6466          ;IN COMMAND INPUT BUFFER
6467          ;
6468 106670 010437 002370          ACTSR4: MOV      R4, CBOADR          ;SAVE STRING POINTER
6469 106674 000207          10$:      RTS      PC
6470          ;
6471          ;ACTION ROUTINE TO SET MESSAGE TYPE = ALPHA FLAG
6472          ;
6473          ;
6474 106676 012737 000000 002372          ACTALP: MOV      @ALPHA, P$TYPE          ;SET MESSAGE TYPE
6475 106704 000207          RTS      PC
6476          ;
6477          ;ACTION ROUTINE TO SET MESSAGE TYPE = ALL ONES FLAG
6478          ;
6479 106706 012737 000001 002372          ACTONE: MOV      @ONES, P$TYPE          ;SET MESSAGE TYPE
6480 106714 000207          RTS      PC
6481          ;
6482          ;ACTION ROUTINE TO SET MESSAGE TYPE = ALL ZEROS FLAG
6483          ;
6484 106716 012737 000002 002372          ACTZRO: MOV      @ZEROS, P$TYPE          ;SET MESSAGE TYPE
6485 106724 000207          RTS      PC
6486          ;
6487          ;ACTION ROUTINE TO SET MESSAGE TYPE = ALTERNATING ONES FLAG
6488          ;
6489 106726 012737 000003 002372          ACT1AL: MOV      @ONEALT, P$TYPE          ;SET MESSAGE TYPE
6490 106734 000207          RTS      PC
6491          ;
6492          ;ACTION ROUTINE TO SET MESSAGE TYPE = ALTERNATING ZEROS FLAG
6493          ;
6494 106736 012737 000004 002372          ACTOAL: MOV      @ZROALT, P$TYPE          ;SET MESSAGE TYPE
6495 106744 000207          RTS      PC
6496          ;
6497          ;ACTION ROUTINE TO SET MESSAGE TYPE = CCITT FLAG
6498          ;
6499 106746 012737 000005 002372          ACTCTT: MOV      @CCITT, P$TYPE          ;SET MESSAGE TYPE
6500 106754 000207          RTS      PC
6501          ;
6502          ;ACTION ROUTINE TO SET MESSAGE TYPE = OPERATOR SELECTED INPUT
6503          ;
6504 106756 105037 003223          ACTOPR: CLRB    P$MERR          ;CLEAR MESSAGE ERROR FLAG
6505 106762 004737 101656          JSR      PC, TRVADR          ;PARSE THROUGH INPUT STRING
6506 106766 105737 003221          TSTB    P$GDBD          ;TEST GOOD/BAD FLAG
6507 106772 001403          BEQ      10$          ;IF GOOD, BR 10$
6508 106774 105037 003221          CLRB    P$GDBD          ;CLEAR FLAG
6509 107000 000415          BR      20$          ;SET CTARGT FLAG AND RETURN
6510 107002 022737 000006 003734 10$:      CMP      @OPRSEL, CFLAG          ;CHECK TO SEE IF STRING VALID
  
```

```

6511 107010 001011                                BNE      20$
6512 107012 012737 000006 002372                MOV      @OPRSEL,P$TYPE
6513 107020                                CALL     SELMSG CBOADR
; IF NOT OK, RETURN WITH ERROR FLAG SET
; SET MESSAGE TYPE
; PUT OPERATOR SELECTED STRING INTO BUFFER
;
; MOV      CBOADR,(R5)
; JSR      R4,PREG14
; .WORD   SELMSG-ANCHOR
6514 107032 000423                                BR       40$
6515 107034 022737 000000 003734 20$:          CMP      @CTARGET,CFLAG
6516 107042 001011                                BNE      30$
6517 107044                                PRINTF   @UNBOND
; RETURN
; SEE IF CTARGET FLAG SET, IF YES ERROR
; IF NOT SET, BR 30$
; ELSE PRINT UNBOUNDED INPUT STRING ERR. MSG.
;
; MOV      @UNBOND,-(SP)
; MOV      @1,(SP)
; MOV      SP,R0
; TRAP     C$PNTF
; ADD      @4,SP
6518 107064 000406                                BR       40$
6519 107066 105737 003223                        30$:    TSTB   P$MERR
6520 107072 001003                                BNE      40$
6521 107074 112737 177777 003221                MOVB     @-1,P$GDBD
6522 107102 000207                        40$:    RTS    PC
; RETURN
; ACTION ROUTINE TO CHECK FOR MORE INPUT AFTER MESSAGE TYPE HAS BEEN
; ALTERED
;
; ACTTYP: JSR      PC,ACTMSG
; CHECK FOR ADDITIONAL COMMANDS
; RTS      PC
;
; ACTION ROUTINE TO INPUT MESSAGE SIZE PARAMETER, CHECK TO SEE IF
; IT IS WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO
; SEE IF MORE INPUT EXISTA
;
; ACTSIZE: CMP      P$NUM,@32.
; CHECK FOR VALID SIZE RANGE
; BLE      10$
; CMP      @513.,P$NUM
; BLE      10$
; IF VALID CONTINUE
; MOV      P$NUM,P$SIZE
; SET MESSAGE SIZE
; BR       20$
6523                                10$:    PRINTF @SIZLMT
; PRINT SIZE LIMITS EXCEEDED MESSAGE
; MOV      @SIZLMT,-(SP)
; MOV      @1,(SP)
; MOV      SP,R0
; TRAP     C$PNTF
; ADD      @4,SP
6524                                20$:    JSR      PC,ACTMSG
6525                                RTS      PC
; CHECK FOR ADDITIONAL COMMANDS
;
; ACTION ROUTINE TO INPUT COPIES PARAMETER, CHECK TO SEE IF IT IS
; WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO SEE IF
; MORE INPUT PARAMETERS EXIST
;
; ACTCPY: CMP      P$NUM,@0
; CHECK FOR VALID COPIES RANGE
; BLE      10$
; CMP      @256.,P$NUM
; BLE      10$
; IF VALID, CONTINUE
; MOV      P$NUM,P$CPYS
; SET MESSAGE COPIES
; BR       20$
6526                                10$:    PRINTF @CPYLMT
; PRINT COPY LIMIT EXCEEDED MESSAGE

```

```

107220 012746 053511                                MOV      @CPYLMT, (SP)
107224 012746 000001                                MOV      @1, (SP)
107230 010600                                        MOV      SP,R0
107232 104417                                        TRAP     C$PNTF
107234 062706 000004                                ADD      @4,SP
6555 107240 004737 106470      20$:      JSR      PC,ACTMSG      ;CHECK FOR ADDITIONAL COMMANDS
6556 107244 000207
6557
6558      ;ACTION ROUTINE TO CLEAR NODE SPECIFIED BY PHYSICAL ADDRESS FROM NODE TABLE
6559
6560 107246 105037 003220      ACTNAD:  CLRB   P$NNUF      ;CLEAR NOTNUF FLAG
6561 107252 105737 003222      TSTB   P$AERR      ;SEE IF ADDRESS ENTERED WAS VALID
6562 107256 001051      BNE    40$      ; IF NOT, EXIT ACTION ROUTINE
6563 107260      P$PUSH R2,R3      ;SAVE R2 AND R3
107260 010325
107262 010225
6564 107264 012702 002316      MOV     @ADRBUF,R2      ;MOVE ADDRESS OF ADDRESS INTO R2
6565 107270 012703 002412      MOV     @NODTBL,R3      ;MOVE ADDRESS OF NODE TABLE INTO R3
6566 107274      10$:      CALL    CMPADR R2,R3      ;SEE IF ADDRESSES MATCH
107274 010325
107276 010225
107300 004437 070656
107304 026004
6567 107306      P$POP  R1
107306 014501
6568 107310 001416      BEQ    20$      ;IF YES, BR 20$
6569 107312 062703 000010      ADD    @10,R3      ;ELSE POINT R3 TO NEXT ENTRY
6570 107316 022713 177777      CMP    @-1,(R3)      ;SEE IF END OF TABLE
6571 107322 001364      BNE    10$      ;IF NOT, COMPARE NEXT ENTRY
6572 107324      PRINTF @NOCMPR      ;ELSE, PRINT ADDRESS DOESN'T COMPARE MSG.
107324 012746 053664                                MOV      @NOCMPR, -(SP)
107330 012746 000001                                MOV      @1, -(SP)
107334 010600                                        MOV      SP,R0
107336 104417                                        TRAP     C$PNTF
107340 062706 000004                                ADD      @4,SP
6573 107344 000414
6574 107346 005023      20$:      BR      30$      ;RETURN
6575 107350 005023      CLR    (R3)+      ;ELSE, CLEAR NODE FROM TABLE
6576 107352 005023      CLR    (R3)+
6577 107354 005013      CLR    (R3)
6578 107356      PRINTF @ADRDEL      ;PRINT NODE DELETED FROM TABLE MESSAGE
107356 012746 054044                                MOV      @ADRDEL, -(SP)
107362 012746 000001                                MOV      @1, -(SP)
107366 010600                                        MOV      SP,R0
107370 104417                                        TRAP     C$PNTF
107372 062706 000004                                ADD      @4,SP
6579 107376      30$:      P$POP  R2,R3      ;RESTORE R2 AND R3
107376 014502                                MOV      -(R5),R2
107400 014503                                MOV      -(R5),R3
6580 107402 000207      40$:      RTS     PC      ;RETURN
6581
6582      ;ACTION ROUTINE TO CLEAR NODE TABLE
6583
6584 107404      ACTNAL: P$PUSH R2,R3      ;SAVE R2,R3
107404 010325
107406 010225
6585 107410 012703 000050      MOV     @TBLEN,R3      ;SET INCR. COUNTER TO 40

```

6586	107414	012702	002412			MOV	#NODTBL,R2		;MOVE NODE TABLE ADDRESS INTO R2	
6587	107420	005022		10%:		CLR	(R2),		;CLEAR BYTE IN NODE TABEL	
6588	107422	005303				DEC	R3		;DECREMENT COUNTER	
6589	107424	001375				BNE	10%		;CONTINUE UNTIL DONE	
6590	107426					PRINTF	#TABCLR,#NOD		;PRINT NODE TABLE CLEARED MESSAGE	
	107426	012746	053415						MOV	#NOD,(SP)
	107432	012746	054220						MOV	#TABCLR,-(SP)
	107436	012746	000002						MOV	#2,-(SP)
	107442	010600							MOV	SP,R0
	107444	104417							TRAP	C\$PNTF
	107446	062706	000006						ADD	#6,SP
6591	107452	105037	003220			CLRB	P\$NNUF		;CLEAR NOTNUF FLAG	
6592	107456					P\$POP	R2,R3		;RESTORE R2 AND R3	
	107456	014502							MOV	-(R5),R2
	107460	014503							MOV	-(R5),R3
6593	107462	000207				RTS	PC			
6594										
6595										
6596										
6597	107464									
6598	107464	042737	100000	000000		BIC	#BIT15,FLAG		; CLEAR OPERATOR FLAG	
6599	107472	105037	003220			CLRB	P\$NNUF		; CLEAR 'NOT ENOUGH' FLAG	
6600	107476	013737	003214	002400		MOV	P\$NUM,P\$PASS			
6601	107504	022737	000032	002312	10%:	CMP	#CRNALL,KEYWD1		; SEE IF 'ALL TEST	
6602	107512	001004				BNE	20%		; IF NO, CONTINUE	
6603	107514					CALL	RUNALL		; IF YES, DO ALLNODE	
	107514	004437	070656						JSR	R4,PREG14
	107520	016726							.WORD	RUNALL-ANCHOR
6604	107522	000423				BR	50%			
6605	107524	022737	000033	002312	20%:	CMP	#CLUPPR,KEYWD1		; IS IT 'LOOPPAIR' TEST	
6606	107532	001004				BNE	30%		; IF NO, CONTINUE	
6607	107534					CALL	RUNLUP		; IF YES, DO LOOPPAIR	
	107534	004437	070656						JSR	R4,PREG14
	107540	020004							.WORD	RUNLUP ANCHOR
6608	107542	000413				BR	50%			
6609	107544	022737	000043	002312	30%:	CMP	#CDIR,KEYWD1		; IS IT 'DIRECT TEST	
6610	107552	001004				BNE	40%		; IF NO, CONTINUE	
6611	107554					CALL	RUNDIR		; IF YES, DO DIRECT	
	107554	004437	070656						JSR	R4,PREG14
	107560	017352							.WORD	RUNDIR ANCHOR
6612	107562	000403				BR	50%			
6613	107564					CALL	RUNPAT		; ELSE, ITS 'PATTERN TEST	
	107564	004437	070656						JSR	R4,PREG14
	107570	021446							.WORD	RUNPAT ANCHOR
6614	107572	023727	002400	177777	50%:	CMP	P\$PASS,# 1		; SEE IF PASS SET FOR INDEFINATE	
6615	107600	001741				BEQ	10%		; IF YES, LOOP	
6616	107602	005337	002400			DEC	P\$PASS		; HAVE WE DONE ALL PASSES?	
6617	107606	001336				BNE	10%		; IF NO, LOOP	
6618	107610	000207				RTS	PC			
6619										
6620										
6621										
6622	107612	012737	000032	002312		MOV	#CRNALL,KEYWD1		; SET FLAG	
6623	107620	000207				RTS	PC			
6624	107622									
	107622	004437	070656			RUNALL:	CALL	DIRCOM	; RUN LOOPDIRECT TEST	
	107626	017372							JSR	R4,PREG14
									.WORD	DIRCOM ANCHOR



```

110056 012746 002324
110062 012746 061757
110066 012746 060770
110072 012746 000005
110076 010600
110100 104414
110102 062706 000014
6648 110106          CALL    RUNCOM          ; DO RECEIVE LOOP
110106 004437 070656          JSR    R4,PREG14
110112 021104          .WORD  RUNCOM ANCHOR
6649 110114          P#POP  R4              ; CHECK RESULTS
110114 014504          MOV    (R5),R4
6650 110116 001405          BEQ    70$            ; IF OK, LOOP SOME MORE
6651 110120          60$: ERRMRD 28,EMSG42,ERR3 ; ELSE PRINT ERROR MESSAGE
110120          TRAP  C1ERRMRD
110122          .WORD  28
110124          .WORD  EMSG42
110126          .WORD  ERR3
6652 110130 000410          BR     80$
6653 110132          70$: PRINTB #OKFU
110132 012746 061157          MOV    #OKFU,(SP)
110136 012746 000001          MOV    #1,(SP)
110142 010600          MOV    SP,R0
110144 104414          TRAP  C1PNTB
110146 062706 000004          ADD   #4,SP
6654 110152 005337 050326          80$: DEC    CPYCNT      ; DECREMENT 'COPIES' COUNTER
6655 110156 001272          BNE   40$            ; IF MORE TO DO, LOOP
6656 110160          CALL  WRITES #2,R1,SLOT ; ELSE, UPDATE SUMMARY TABLE
110160 013725 002410          MOV    SLOT,(R5)+
110164 010125          MOV    R1,(R5)+
110166 012725 000002          MOV    #2,(R5)+
110172 004437 070656          JSR    R4,PREG14
110176 007176          .WORD  WRITES-ANCHOR
6657 110200 000642          BR     30$
6658 110202 062701 000010          90$: ADD    #10,R1      ; POINT R1 TO NEXT TARGET NODE
6659 110206 010137 002410          MOV    R1,SLOT      ; UPDATE SLOT
6660 110212          CALL  FULSLT        ; GET ADDRESS FROM TABLE
110212 004437 070656          JSR    R4,PREG14
110216 025556          .WORD  FULSLT-ANCHOR
6661 110220 013701 002410          MOV    SLOT,R1
6662 110224 022737 177777 002410          CMP    #-1,SLOT     ; SEE IF END OF TABLE
6663 110232 001225          BNE   30$            ; IF NO, CONTINUE ELSE, FINISHED
6664 110234          100$: RETURN
110234 000207          RTS    PC
6665          ;
6666          ; ACTION ROUTINE TO SET RUN LOOP DIRECT FLAG
6667          ;
6668          ;
6669 110236          ACTDIR:
6670 110236 012737 000043 002312          MOV    #CDIR,KEYWD1 ; SET FLAG
6671 110244 000207          RTS    PC
6672          ;
6673 110246          RUNDIR: CALL    DIRCOM ; CALL COMMON CODE
110246 004437 070656          JSR    R4,PREG14
110252 017372          .WORD  DIRCOM ANCHOR
6674 110254          P#POP  R1
110254 014501          MOV    (R5),R1

```

```

6675 110256 022701 000001          CMP      #1,R1          ; WAS TABLE EMPTY?
6676 110262 001400                  BEQ      10$           ; IF YES, DON'T PRINT
6677 110264                  10$:      RETURN
                                RTS      PC
6678 110266 000207          DIRCOM: CLR      R1          ; CLEAR RESULTS REGISTER
6679 110270 012737 002412 002410  MOV      #NODTBL,SLOT ; MOVE NODE TABLE ADDRESS TO SLOT
6680 110276          CALL      FULSLT      ; SEE IF TABLE EMPTY
                                JSR      R4,PREG14
                                .WORD    FULSLT-ANCHOR
6681 110304 022737 177777 002410  CMP      #-1,SLOT      ; IF NO CONTINUE
6682 110312 001015          BNE      10$           ; ELSE, PRINT "TABLE EMPTY" MESSAGE
6683 110314          PRINTF  #TABEMT,#NOD
                                MOV      #NOD,(SP)
                                MOV      #TABEMT,-(SP)
                                MOV      #2,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTF
                                ADD      #6,SP
6684 110340 012701 000001          MOV      #1,R1          ; PUT 'TABLE EMPTY' INDICATOR IN R1
6685 110344 000547          BR      100$
6686 110346 012737 002412 002410 10$:      MOV      #NODTBL,SLOT
6687 110354 013737 002376 050326 20$:      MOV      P$CPYS,CPYCNT
6688 110362          CALL      FULSLT      ; SET UP FOR NO. OF COPIES
                                ; GET NEXT NODE IN TABLE
                                JSR      R4,PREG14
                                .WORD    FULSLT-ANCHOR
6689 110370 022737 177777 002410  CMP      #-1,SLOT      ; SEE IF AT END OF TABLE
6690 110376 001532          BEQ      100$         ; IF YES, EXIT
6691 110400          CALL      BINMEX SLOT,#6,#STRBUF ; PRINT ADDRESS BEING TESTED
                                MOV      #STRBUF,(R5)
                                MOV      #6,(R5)
                                MOV      SLOT,(R5)
                                JSR      R4,PREG14
                                .WORD    BINMEX-ANCHOR
6692 110422          30$:      PRINTB #TSTMS2,#DIRECT,#STRBUF ; NODE ADDRESS
                                MOV      #STRBUF,-(SP)
                                MOV      #DIRECT,-(SP)
                                MOV      #TSTMS2,-(SP)
                                MOV      #3,(SP)
                                MOV      SP,R0
                                TRAP     C$PNTB
                                ADD      #10,SP
6693 110452 022737 000005 002312  CMP      #CPATRN,KEYWD1
6694 110460 001016          BNE      40$
6695 110462 013701 002372          MOV      P$TYPE,R1
6696 110466 006301          ASL      R1
6697 110470 062701 003324          ADD      #MSGTAB,R1
6698 110474          PRINTB #MESPA1,(R1)
                                MOV      (R1),(SP)
                                MOV      #MESPA1,(SP)
                                MOV      #2,(SP)
                                MOV      SP,R0
                                TRAP     C$PNTB
                                ADD      #6,SP
6699 110516          40$:      CALL      BLDLD SLOT      ; CALL BUILD LOOPDIRECT SUBROUTINE
                                MOV      SLOT,(R5)
                                JSR      R4,PREG14
                                .WORD    BLDLD ANCHOR
110516 013725 002410
110522 004437 070656
110526 005236

```



6700	110530				CALL	XMIT	@XMTDAT		; TRANSMIT LOOPDIRECT MESSAGES
	110530	012725	000001						MOV @XMTDAT,(R5).
	110534	004437	070656						JSR R4,PREG14
	110540	002326							.WORD XMIT-ANCHOR
6701	110542				P&POP	R2			; GET RESULTS, R2 = SUCCESS/FAILURE
	110542	014502							MOV (R5),R2
6702	110544	001405			BEQ	60%			; IF OK, EXIT
6703	110546			50%:	ERRHRD	26,EMSG24			; ELSE PRINT ERROR MESSAGE
	110546	104456							TRAP C%ERRHRD
	110550	000032							.WORD 26
	110552	064103							.WORD EMSG24
	110554	000000							.WORD 0
6704	110556	000676			BR	20%			
6705	110560			60%:	CALL	RUNCOM			; DO RECEIVE LOOP
	110560	004437	070656						JSR R4,PREG14
	110564	021104							.WORD RUNCOM-ANCHOR
6706	110566				P&POP	R4			; GET RESULTS
	110566	014504							MOV (R5),R4
6707	110570	001407			BEQ	70%			; IF NO ERRORS, CONTINUE
6708	110572				ERRHRD	27,EMSG34,ERR2			
	110572	104456							TRAP C%ERRHRD
	110574	000033							.WORD 27
	110576	064322							.WORD EMSG34
	110600	070344							.WORD ERR2
6709	110602	012701	177777		MOV	#1,R1			; PUT ERROR INDICATOR INTO R1
6710	110606	000410			BR	80%			
6711	110610			70%:	PRINTB	#OK			; RESPONSE OK
	110610	012746	061025						MOV #OK,(SP)
	110614	012746	000001						MOV #1,-(SP)
	110620	010600							MOV SP,R0
	110622	104414							TRAP C%PNTB
	110624	062706	000004						ADD #4,SP
6712	110630	005337	050326	80%:	DEC	CPYCNT			; DECREMENT 'COPIES' COUNTER
6713	110634	001272			BNE	30%			; IF MORE TO DO, LOOP
6714	110636				CALL	WRITES #1,SLOT			; ELSE,UPDATE SUMMARY TABLE
	110636	013725	002410						MOV SLOT,(R5).
	110642	012725	000001						MOV #1,(R5).
	110646	004437	070656						JSR R4,PREG14
	110652	007176							.WORD WRITES ANCHOR
6715	110654	062737	000010	002410	90%:	ADD	#10,SLOT		; INCREMENT TO NEXT NODE TABLE ENTRY
6716	110662	000634			BR	20%			
6717	110664			100%:	RETURN	R1			
	110664	010125							MOV R1,(R5).
	110666	000207							RTS PC
6718									
6719									; ACTION ROUTINE TO SET 'RUN LOOPPAIR' FLAG
6720									
6721	110670	012737	000033	002312	ACTRNL:	MOV	@CLUPPR,KEYWD1		; SET FLAG
6722	110676	000207				RTS	PC		
6723									
6724	110700				RUNLUP:				
6725	110700				CALL	CHECK			
	110700	004437	070656						JSR R4,PREG14
	110704	025624							.WORD CHECK ANCHOR
6726	110706	005037	050316		CLR	TEMP1			; CLEAR 'HEADER PRINTED' FLAG
6727	110712	012737	002412	002410	MOV	#NODTBL,SLOT			; MOVE NODE TABLE ADDRESS TO SLOT
6728	110720				CALL	FULSLT			; SEE IF TABLE EMPTY

```

110720 004437 070656
110724 025556
6729 110726 022737 177777 002410      CMP     # 1,SLOT
6730 110734 001014                      BNE     9$
6731 110736                      PRINTF  #TABEMT, #NOD
    110736 012746 053415
    110742 012746 053346
    110746 012746 000002
    110752 010600
    110754 104417
    110756 062706 000006
6732 110762 000137 111776                      JMP     30$
6733 110766 012737 002412 002410 9$:        MOV     #NODTBL,SLOT
6734 110774 013737 002376 050326 10$:      MOV     P#CPYS,CPYCNT
6735 111002                      CALL    FULSLT
    111002 004437 070656
    111006 025556
6736 111010 022737 177777 002410      CMP     # 1,SLOT
6737 111016 001002                      BNE     11$
6738 111020 000137 111776                      JMP     30$
6739 111024 013701 002410 11$:          MOV     SLOT,R1
6740 111030 126127 000007 000000          CMPB   7(R1),#CTARGET
6741 111036 001422                      BEQ    18$
6742 111040 17$:          PRINTF  #EMSG32
    111040 012746 064245
    111044 012746 000001
    111050 010600
    111052 104417
    111054 062706 000004
6743 111060                      PRINTF  #PASABT
    111060 012746 060672
    111064 012746 000001
    111070 010600
    111072 104417
    111074 062706 000004
6744 111100 000137 111776                      JMP     30$
6745 111104 010102 18$:          MOV     R1,R2
6746 111106 062702 000010 112$:      ADD     #10,R2
6747 111112 021227 000000          CMP     (R2),#0
6748 111116 001773                      BEQ    112$
6749 111120 022712 177777          CMP     #-1,(R2)
6750 111124 001002                      BNE    110$
6751 111126 000137 111776                      JMP     30$
6752 111132 126227 000007 000001 110$:   CMPB   7(R2),#CASIST
6753 111140 001337                      BNE    17$
6754 111142 20$:          CALL    BLDAST R2,R1
    111142 010125
    111144 010225
    111146 004437 070656
    111152 006030
6755 111154                      CALL    XMIT
    111154 004437 070656
    111160 002326
6756 111162                      P#POP  R4
    111162 014504
6757 111164 001406                      BEQ    22$
6758 111166 21$:          ERRHRD  26,EMSG24

                                   JSR     R4,PREG14
                                   .WORD  FULSLT ANCHOR
;
; IF NO, CONTINUE
; ELSE, PRINT "TABLE EMPTY" MESSAGE
                                   MOV     #NOD, (SP)
                                   MOV     #TABEMT, (SP)
                                   MOV     #2, (SP)
                                   MOV     SP,R0
                                   TRAP   C#PNTF
                                   ADD     #6,SP
; MOVE NODE TABLE ADDRESS TO SLOT
; SET UP FOR NO. OF COPIES
; GET NEXT NODE IN TABLE
                                   JSR     R4,PREG14
                                   .WORD  FULSLT ANCHOR
; SEE IF AT END OF TABLE
; IF YES, EXIT
; MOVE SLOT TO R1
; SEE IF TARGET NODE
; IF YES, BRANCH
; ELSE PRINT ERROR MESSAGE
                                   MOV     #EMSG32, (SP)
                                   MOV     #1, (SP)
                                   MOV     SP,R0
                                   TRAP   C#PNTF
                                   ADD     #4,SP
                                   MOV     #PASABT, -(SP)
                                   MOV     #1, -(SP)
                                   MOV     SP,R0
                                   TRAP   C#PNTF
                                   ADD     #4,SP
; EXIT
; POINT R1 TO TARGET NODE
; AND R2 TO ASSIST NODE
; IS R2 POINTING TO AN ASSIST NODE?
; IF NOT, ERROR, ELSE CONTINUE
; BUILD TRANSMIT ASSIST MESSAGE
                                   MOV     R1,(R5)
                                   MOV     R2,(R5)
                                   JSR     R4,PREG14
                                   .WORD  BLDAST-ANCHOR
; TRANSMIT MESSAGE
                                   JSR     R4,PREG14
                                   .WORD  XMIT ANCHOR
; GET RESULTS, R2 = SUCCESS/FAILURE
                                   MOV     (R5),R4
; IF OK, EXIT
; ELSE PRINT ERROR MESSAGE
    
```

	111166	104456						TRAP	C\$ERHRD
	111170	000032						.WORD	26
	111172	064103						.WORD	MSG24
	111174	000000						.WORD	0
6759	111176	000137	111712		JMP	28\$			
6760	111202			22\$:	CALL	BINHEX R1,06,@STRBUF	; PRINT ERROR MESSAGE		
	111202	012725	002324					MOV	@STRBUF,(R5).
	111206	012725	000006					MOV	06,(R5).
	111212	010125						MOV	R1,(R5).
	111214	004437	070656					JSR	R4,PREG14
	111220	005150						.WORD	BINHEX ANCHOR
6761	111222				CALL	B_NHEX R2,06,@STRBU1	;		
	111222	012725	002346					MOV	@STRBU1,(R5).
	111226	012725	000006					MOV	06,(R5).
	111232	010225						MOV	R2,(R5).
	111234	004437	070656					JSR	R4,PREG14
	111240	005150						.WORD	BINHEX ANCHOR
6762	111242				PRINTB	@TSTMS4,@ARGTY7,@STRBUF,@ARGTY6,@STRBU1	; ASSIST NODE =		
	111242	012746	002346					MOV	@STRBU1,(SP)
	111246	012746	061747					MOV	@ARGTY6,(SP)
	111252	012746	002324					MOV	@STRBUF,(SP)
	111256	012746	061757					MOV	@ARGTY7,(SP)
	111262	012746	060770					MOV	@TSTMS4,-(SP)
	111266	012746	000005					MOV	05,(SP)
	111272	010600						MOV	SP,R0
	111274	104414						TRAP	C\$PNTB
	111276	062706	000014					ADD	014,SP
6763	111302				CALL	RUNCOM	; DO RECIEVE LOOP		
	111302	004437	070656					JSR	R4,PREG14
	111306	021104						.WORD	RUNCOM ANCHOR
6764	111310				P\$POP	R3	; CHECK RESULTS		
	111310	014503						MOV	-(R5),R3
6765	111312	001405			BEQ	23\$	; IF OK, CONT.		
6766	111314				ERRHRD	28,MSG40,ERR3			
	111314	104456						TRAP	C\$ERHRD
	111316	000034						.WORD	28
	111320	064561						.WORD	MSG40
	111322	070432						.WORD	ERR3
6767	111324	000410			BR	101\$			
6768	111326			23\$:	PRINTB	00KRE			
	111326	012746	061046					MOV	00KRE,(SP)
	111332	012746	000001					MOV	01,(SP)
	111336	010600						MOV	SP,R0
	111340	104414						TRAP	C\$PNTB
	111342	062706	000004					ADD	04,SP
6769	111346			101\$:	CALL	BLDAST R1,R2	; BUILD RECEIVE ASSIST MESSAGE		
	111346	010225						MOV	R2,(R5).
	111350	010125						MOV	R1,(R5).
	111352	004437	070656					JSR	R4,PREG14
	111356	006030						.WORD	BLDAST ANCHOR
6770	111360				CALL	XMIT	; TRANSMIT MESSAGE		
	111360	004437	070656					JSR	R4,PREG14
	111364	002326						.WORD	XMIT ANCHOR
6771	111366				P\$POP	R4	; CHECK RESULTS		
	111366	014504						MOV	-(R5),R4
6772	111370	001276			BNE	21\$	; IF OK CONTINUE, ELSE REPORT ERROR		
6773	111372				CALL	BINHEX R1,06,@STRBUF	; PRINT ERROR MESSAGE		

	111372	012725	002324				MOV	#STRBUF,(R5).
	111376	012725	000006				MOV	#6,(R5).
	111402	010125					MOV	R1,(R5).
	111404	004437	070656				JSR	R4,PREG14
	111410	005150					.WORD	BINHEX-ANCHOR
6774	111412			CALL	BINHEX	R2,#6,#STRBU1	:	
	111412	012725	002346				MOV	#STRBU1,(R5).
	111416	012725	000006				MOV	#6,(R5).
	111422	010225					MOV	R2,(R5).
	111424	004437	070656				JSR	R4,PREG14
	111430	005150					.WORD	BINHEX-ANCHOR
6775	111432			PRINTB	#TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1	:	ASSIST	MODE =
	111432	012746	002346				MOV	#STRBU1,(SP)
	111436	012746	061747				MOV	#ARGTY6,-(SP)
	111442	012746	002324				MOV	#STRBUF,-(SP)
	111446	012746	061757				MOV	#ARGTY7,(SP)
	111452	012746	060770				MOV	#TSTMS4,(SP)
	111456	012746	000005				MOV	#5,-(SP)
	111462	010600					MOV	SP,R0
	111464	104414					TRAP	C#PNTB
	111466	062706	000014				ADD	#14,SP
6776	111472			CALL	RUNCOM	:	DO	RECEIVE LOOP
	111472	004437	070656				ISR	R4,PREG14
	111476	021104					.WORD	RUNCOM-ANCHOR
6777	111500			P#POP	R3	:	GET	RESULTS
	111500	014503					MOV	(R5),R3
6778	111502	001405		BEQ	25#	:	IF	OK, CONT.
6779	111504			ERRHRD	28,MSG41,ERR3			
	111504	104456					TRAP	C#ERRHRD
	111506	000034					.WORD	28
	111510	064662					.WORD	MSG41
	111512	070432					.WORD	ERR3
6780	111514	000410		BR	102#			
6781	111516			PRINTB	#OKTR			
	111516	012746	061112				MOV	#OKTR,(SP)
	111522	012746	000001				MOV	#1,-(SP)
	111526	010600					MOV	SP,R0
	111530	104414					TRAP	C#PNTB
	111532	062706	000004				ADD	#4,SP
6782	111536			102#:	CALL	BLDFA5	R1,R2	: BUILD FULL ASSIST MESSAGE
	111536	010225					MOV	R2,(R5).
	111540	010125					MOV	R1,(R5).
	111542	004437	070656				JSR	R4,PREG14
	111546	005502					.WORD	BLDFAS-ANCHOR
6783	111550			CALL	XMIT	:	TRANSMIT	MESSAGE
	111550	004437	070656				JSR	R4,PREG14
	111554	002326					.WORD	XMIT-ANCHOR
6784	111556			P#POP	R4	:	CHECK	RESULTS
	111556	014504					MOV	-(R5),R4
6785	111560	001402		BEQ	26#	:	IF	OK CONTINUE, ELSE REPORT ERROR
6786	111562	000137	111166	JMP	21#			
6787	111566			26#:	CALL	BINHEX	R1,#6,#STRBUF	: PRINT ERROR MESSAGE
	111566	012725	002324				MOV	#STRBUF,(R5).
	111572	012725	000006				MOV	#6,(R5).
	111576	010125					MOV	R1,(R5).
	111600	004437	070656				JSR	R4,PREG14
	111604	005150					.WORD	BINHEX ANCHOR

```

6788 111606          CALL    BINHEX  R2,06,0STRBU1  ;
      111606 012725 002346
      111612 012725 000006
      111616 010225
      111620 004437 070656
      111624 005150
      6789 111626          PRINTB 0TSTMS4,0ARGTY7,0STRBUF,0ARGTY6,0STRBU1 ;
      111626 012746 002346
      111632 012746 061747
      111636 012746 002324
      111642 012746 061757
      111646 012746 060770
      111652 012746 000005
      111656 010600
      111660 104414
      111662 062706 000014
      6790 111666          CALL    RUNCOM          ; DO RECEIVE LOOP
      111666 004437 070656
      111672 021104
      6791 111674          P$POP   R3          ; CHECK RESULTS
      111674 014503
      6792 111676          BEQ     28$          ; IF NO ERRORS, CONT
      6793 111700          ERRHRD 28,MSG42,ERR3
      111700 104456
      111702 000034
      111704 064764
      111706 070432
      6794 111710          BR      103$
      6795 111712          28$: PRINTB 0OKFU
      111712 012746 061157
      111716 012746 000001
      111722 010600
      111724 104414
      111726 062706 000004
      6796 111732          103$: DEC    CPYCNT          ; DECREMENT 'COPIES COUNTER
      6797 111736          BEQ     29$          ; IF MORE TO DO, LOOP
      6798 111740          JMP     20$
      6799 111744          29$: CALL   WRITES 02,R1,R2          ; ELSE,UPDATE SUMMARY TABLE
      111744 010225
      111746 010125
      111750 012725 000002
      111754 004437 070656
      111760 007176
      6800 111762          ADD     010,R2
      6801 111766          MOV     R2,SLOT
      6802 111772          JMP     10$
      6803 111776          30$: RETURN
      111776 000207
      6804 112000          RUNCOM: TST   RETRYS          ; SEE IF FAILED DUE TO EXCESSIVE COLLISIONS
      6805 112004          BNE    30$          ; IF YES, BR, ELSE CONT.
      6806 112006          MOV    0TIMERS,R4          ; SET UP FOR 10 SECOND TIMEOUT
      6807 112012          MOV    010.,(R4)
      6808 112016          10$: BREAK
      112016 104422
      6809 112020          TST   (R4)          ; SEE IF TIME HAS EXPIRED
      6810 112022          BEQ   40$          ; IF YES, BRANCH
      6811 112024          CALL  RECEVE          ; CHECK FOR ANSWER
      111616 010225
      111620 004437 070656
      111624 005150
      111626 012746 002346
      111632 012746 061747
      111636 012746 002324
      111642 012746 061757
      111646 012746 060770
      111652 012746 000005
      111656 010600
      111660 104414
      111662 062706 000014
      111666 004437 070656
      111672 021104
      111674 014503
      111676 001405
      111700 104456
      111702 000034
      111704 064764
      111706 070432
      111710 000410
      111712 012746 061157
      111716 012746 000001
      111722 010600
      111724 104414
      111726 062706 000004
      111732 005337 050326
      111736 001402
      111740 000137 111142
      111744 010225
      111746 010125
      111750 012725 000002
      111754 004437 070656
      111760 007176
      111762 062702 000010
      111766 010237 002410
      111772 000137 110774
      111776 000207
      112000 005737 050274
      112004 001120
      112006 012704 003762
      112012 012714 000012
      112016 104422
      112020 005714
      112022 001517
      112024
  
```

```

112024 004437 070656
112030 003044
6812 112032          P;POP  R1          ; R2 HOLDS NO. OF BUFFERS RECEIVED
112032 014501          MOV      -(R5),R1
6813 112034 001770          BEQ      10;          ; IF NO BUFFERS RECEIVED, LOOP
6814 112036 063737 050324 050042          ADD      XFER,S,XFER          ; UPDATE BYTES TRANSFERED SUM. COUNTER
6815 112044 005237 050030          INC      S.REC              ; UPDATE PACKETS RECEIVED SUM. COUNTER
6816 112050 013703 024226          MOV      RRGXNT,R3          ; GET RECEIVE RING POINTER
6817 112054 116301 000012          MOVVB   STAT2(R3),R1        ; GET RCV BYTE COUNT BITS 0 7
6818 112060 116302 000011          MOVVB   STAT1+1(R3),R2      ; GET RCV BYTE COUNT BITS 8-10
6819 112064 042701 177400          BIC     #177400,R1          ; CLEAR ALL UNWANTED BITS
6820 112070 042702 000370          BIC     #370,R2            ; CLEAR ALL UNWANTED BITS
6821 112074 000302          SWAB   R2                  ; PUT 8-10 INTO HIGH BYTE
6822 112076 050201          BIS     R2,R1              ; MAKE BYTE COUNT COMPLETE
6823 112100 062701 000074          ADD     #60.,R1            ; ADD BYTES LOST WHILE COMPARING ADDRESSES
6824 112104 020137 050332          CMP     R1,BUFLEN          ; CHECK FOR LENGTH ERROR
6825 112110 001423          BEQ     20;                ; IF OK, BR
6826 112112 005237 050034          INC     S.LEN              ; ELSE, UPDATE LENGTH ERRORS COUNTER
6827 112116 012737 061506 002314          MOV     #LENGTH,KEYWD2     ; MOVE 'LENGTH' TO ERROR INDICATOR
6828 112124 012702 177777          MOV     #-1,R2             ; INDICATE ERROR TO R2
6829 112130          PRINTX #LGERMS,BUFLEN,R1 ; PRINT LENGTH ERROR MESSAGE
112130 010146          MOV     R1,-(SP)
112132 013746 050332          MOV     BUFLN,(SP)
112136 012746 067745          MOV     #LGERMS,-(SP)
112142 012746 000003          MOV     #3,(SP)
112146 010600          MOV     SP,R0
112150 104415          TRAP   C;PNTX
112152 062706 000010          ADD     #10,SP
6830 112156 000452          BR      60;                ; AND EXIT
6831 112160 016303 000004 20;          MOV     LOADD(R3),R3        ; POINT R3 TO MESSAGE BUFFER
6832 112164 066303 000016          ADD     16(R3),R3          ; POINT R3 TO DATA AFTER SKIP COUNT
6833 112170 062703 000030          ADD     #30,R3            ; POINT R3 TO FIRST DATA BYTE
6834 112174 063737 002374 050040          ADD     P;SIZE,S,BYTE      ; UPDATE BYTES COMPARED SUMMARY COUNTER
6835 112202          CALL   DATCMP P;SIZE,CMPBUF,R3 ; CHECK FOR DATA COMPARE ERRORS
112202 010325          MOV     R3,(R5)+
112204 013725 050334          MOV     CMPBUF,(R5)+
112210 013725 002374          MOV     P;SIZE,(R5)+
112214 004437 070656          JSR     R4,PREG14
112220 007040          .WORD  DATCMP ANCHOR
6836 112222          P;POP  R3                  ; CHECK RESULTS
112222 014503          MOV     -(R5),R3
6837 112224 001426          BEQ     50;                ; BRANCH IF NO ERRORS
6838 112226 060337 050036          ADD     R3,S,COMP          ; UPDATE COMPARE ERRORS SUMMARY COUNTER
6839 112232 012737 061515 002314          MOV     #COMPARE,KEYWD2    ; MOVE 'COMPARE' TO ERROR INDICATOR
6840 112240 012702 177777          MOV     #-1,R2             ; INDICATE ERROR TO R2
6841 112244 000417          BR      60;
6842
6843 112246 012737 061462 002314 30;          MOV     #RETRY,KEYWD2      ; MOVE 'EXCESSIVE COLLISIONS' TO ERROR INCICATOR
6844 112254 012702 177777          MOV     #-1,R2             ; INDICATE ERROR IN R2
6845 112260 000411          BR      60;
6846
6847 112262 005237 050032 40;          INC     S.NREC              ; UPDATE MESSAGES NOT RECEIVED COUNTER
6848 112266 012737 061446 002314          MOV     #NORESP,KEYWD2    ; MOVE 'NO RESPONCE' TO ERROR INDICATOR
6849 112274 012702 177777          MOV     #-1,R2             ; INDICATE ERROR TO R2
6850 112300 000412          BR      65;                ; LEAVE
6851 112302          50;
6852 112302 005002          CLR     R2                  ; INDICATE SUCCESS

```

```

6853 112304
6854 112304 013703 024226
6855 112310
      112310 012725 024226
      112314 012725 000001
      112320 004437 070656
      112324 006450
6856 112326
6857 112326
      112326 010225
      112330 000207
6858
6859
6860
6861
6862
6863 112332 012737 000005 002312 ACTPAT: MOV #CPATRN,KEYWD1 ;SET FLAG
6864 112340 000207 RTS PC
6865
6866
6867 112342
      112342 013725 002372
6868 112346 005037 002372
6869 112352
      112352 004437 070656
      112356 017372
6870 112360
      112360 014501
6871 112362 001403
6872 112364 022701 000001
6873 112370 001406
6874 112372 005237 002372
6875 112376 022737 000005 002372
6876 112404 002362
6877 112406
      112406 014537 002372
6878 112412
      112412 000207
6879
6880
6881
6882
6883
6884 112414 013701 002372
6885 112420 006301
6886 112422 062701 003324
6887 112426
      112426 012746 054345
      112432 012746 000001
      112436 010600
      112440 104417
      112442 062706 000004
6888 112446
      112446 013746 002376
      112452 013746 002374
      112456 011146
      112460 012746 055002

60$:
      MOV RRG NXT,R3 ; GET DESCRIPTOR
      CALL GETRNX #1,#RRGNXT ; UPDATE POINTER, VALIDATE PREVIOUS DESCRIPTOR
      MOV #RRGNXT,(R5)+
      MOV #1,(R5)+
      JSR R4,PREG14
      .WORD GETRNX ANCHOR

65$:
      RETURN R2 ; RETURN
      MOV R2,(R5)+
      RTS PC

; ACTION ROUTINE TO SET 'RUN PATTERN' FLAG
;
ACTPAT: MOV #CPATRN,KEYWD1 ;SET FLAG
        RTS PC

RUNPAT: P$PUSH P$TYPE ; SAVE TYPE PARAMETER
        MOV P$TYPE,(R5)+
        CLR P$TYPE ; SET TYPE TO FIRST TYPE
10$: CALL DIRCOM ; SEND MESSAGES
        JSR R4,PREG14
        .WORD DIRCOM ANCHOR
        MOV (R5),R1
        ; GET RESULTS TO KEEP STACK IN ORDER
        ; IF OK, CONT
        ; ELSE, WAS TABLE EMPTY
        ; IF YES, RETURN
        ; SET TO NEXT TYPE
        ; SEE IF DONE ALL OF THEM
        ; IF NOT, DO MORE
30$: P$POP P$TYPE ; RESTORE MESSAGE TYPE
        MOV (R5),P$TYPE
        RTS PC

; ACTION ROUTINE TO SHOW THE CURRENT MESSAGE PARAMETERS
;
ACTSMS: MOV P$TYPE,R1 ;GET MESSAGE TYPE INTO R1
        ASL R1 ;MULTIPLY BY 2
        ADD #MSGTAB,R1 ;ADD MESSAGE TABLE OFFSET
        PRINTF #MSGPRM ;PRINT MESSAGE PARAMETER MESSAGE
        MOV #MSGPRM,(SP)
        MOV #1,(SP)
        MOV SP,R0
        TRAP C$PNTF
        ADD #4,SP
        ;PRINT PARAMETERS
        MOV P$CPYS,-(SP)
        MOV P$SIZE,-(SP)
        MOV (R1),-(SP)
        MOV #MSG4,-(SP)
  
```

```

112464 012746 000004
112470 010600
112472 104417
112474 062706 000012
6889 112500 105037 003220          CLRB   P$NNUF
6890 112504 000207          RTS    PC
6891
6892
6893          ; ACTION ROUTINE TO CLEAR THE CURRENT MESSAGE PARAMETERS AND
6894          ; RESET THEM TO THE DEFAULT VALUE
6895          ;
6896
6897 112506 012737 000000 002372 ACTCMS: MOV    @ALPHA,P$TYPE          ; RESET TYPE
6898 112514 012737 001000 002374      MOV    @512.,P$SIZE          ; RESET SIZE
6899 112522 012737 000001 002376      MOV    ^1,P$CPYS           ; RESET COPIES
6900 112530          PRINTF @CLRMSG          ; PRINT MESSAGE PARAMETERS RESET MESSAGE
          MOV    @CLRMSG,(SP)
          MOV    @1,(SP)
          MOV    SP,RO
          TRAP  C$PNTF
          ADD   @4,SP
112530 012746 053432
112534 012746 000001
112540 010600
112542 104417
112544 062706 000004
6901 112550          PRINTF @MSG4,MSGTAB,P$SIZE,P$CPYS      ; PRINT PARAMETERS
          MOV    P$CPYS,-(SP)
          MOV    P$SIZE,-(SP)
          MOV    MSGTAB,-(SP)
          MOV    @MSG4,-(SP)
          MOV    @4,-(SP)
          MOV    SP,RO
          TRAP  C$PNTF
          ADD   @12,SP
112550 013746 002376
112554 013746 002374
112560 013746 003324
112564 012746 055002
112570 012746 000004
112574 010600
112576 104417
112600 062706 000012
6902 112604 105037 003220          CLRB   P$NNUF          ; CLEAR NOTNUF FLAG
6903 112610 000207          RTS    PC
6904
6905
6906
6907          ; ACTION ROUTINE TO SET SHOW COUNTERS FLAG
6908          ;
6909
6910 112612          ACTCNT:
6911 112612          P$PUSH R1,R2,R3          ; SAVE REGISTERS
          MOV    R3,(R5)+
          MOV    R2,(R5)+
          MOV    R1,(R5)+
6912 112620 010325          MOV    @CNTRS,R1          ; GET ADDRESS OF COUNTERS
6913 112624 012701 050144          MOV    @UCB12,R2          ; GET ADDRESS OF COUNTER BUFFER
6914 112630 016103 000000          MOV    C.SIZ(R1),R3       ; GET SIZE OF BUFFER
6915 112634          10$:
6916 112634 112122          MOVB  (R1)+,(R2)+          ; MOV COUNTERS TO BUFFER
6917 112636 005303          DEC   R3                  ; SUBTRACT ONE FROM THE COUNT
6918 112640 001375          BNE   10$                 ; BRANCH UNTIL ALL ARE DONE
6919 112642          P$POP  R1,R2,R3         ; RESTORE THE REGISTERS
          MOV    -(R5),R1
          MOV    -(R5),R2
          MOV    -(R5),R3
6920 112650          CALL  BINHEX @PHYADR,@6,@STRBUF ; GET ADDRESS INTO ASCII
          MOV    @STRBUF,(R5)+
          MOV    @6,(R5)+
112650 012725 002324
112654 012725 000006

```



112660	012725	004010				MOV	#PHYADR,(R5).
112664	004437	070656				JSR	R4,PREG14
112670	005150					.WORD	BINHEX-ANCHOR
6921	112672		PRINTF	#CNTR00,#STRBUF			
	112672	012746				MOV	#STRBUF,-(SP)
	112676	012746				MOV	#CNTR00,-(SP)
	112702	012746				MOV	#2,-(SP)
	112706	010600				MOV	SP,R0
	112710	104417				TRAP	C#PNTF
	112712	062706				ADD	#6,SP
6922	112716		PRINTF	#CNTR01,UCB12+2			
	112716	013746				MOV	UCB12+2,-(SP)
	112722	012746				MOV	#CNTR01,(SP)
	112726	012746				MOV	#2,-(SP)
	112732	010600				MOV	SP,R0
	112734	104417				TRAP	C#PNTF
	112736	062706				ADD	#6,SP
6923	112742		CALL	BINDEC #UCB12+4			
	112742	012725				MOV	#UCB12+4,(R5).
	112746	004437				JSR	R4,PREG14
	112752	007446				.WORD	BINDEC-ANCHOR
6924	112754		PRINTF	#CNTR02,#DECSTR			
	112754	012746				MOV	#DECSTR,-(SP)
	112760	012746				MOV	#CNTR02,-(SP)
	112764	012746				MOV	#2,-(SP)
	112770	010600				MOV	SP,R0
	112772	104417				TRAP	C#PNTF
	112774	062706				ADD	#6,SP
6925	113000		CALL	BINDEC #UCB12+10			
	113000	012725				MOV	#UCB12+10,(R5).
	113004	004437				JSR	R4,PREG14
	113010	007446				.WORD	BINDEC-ANCHOR
6926	113012		PRINTF	#CNTR03,#DECSTR			
	113012	012746				MOV	#DECSTR,-(SP)
	113016	012746				MOV	#CNTR03,-(SP)
	113022	012746				MOV	#2,-(SP)
	113026	010600				MOV	SP,R0
	113030	104417				TRAP	C#PNTF
	113032	062706				ADD	#6,SP
6927	113036		PRINTF	#CNTR04,UCB12+14			
	113036	013746				MOV	UCB12+14,-(SP)
	113042	012746				MOV	#CNTR04,-(SP)
	113046	012746				MOV	#2,-(SP)
	113052	010600				MOV	SP,R0
	113054	104417				TRAP	C#PNTF
	113056	062706				ADD	#6,SP
6928	113062		PRINTF	#CNTR05,UCB12+16			
	113062	013746				MOV	UCB12+16,-(SP)
	113066	012746				MOV	#CNTR05,-(SP)
	113072	012746				MOV	#2,-(SP)
	113076	010600				MOV	SP,R0
	113100	104417				TRAP	C#PNTF
	113102	062706				ADD	#6,SP
6929	113106		CALL	BINDEC #UCB12+20			
	113106	012725				MOV	#UCB12+20,(R5).
	113112	004437				JSR	R4,PREG14
	113116	007446				.WORD	BINDEC-ANCHOR

6930	113120			PRINTF	@CNTR06,@DECSTR		
	113120	012746	100526			MOV	@DECSTR,(SP)
	113124	012746	062424			MOV	@CNTR06,-(SP)
	113130	012746	000002			MOV	@2,(SP)
	113134	010600				MOV	SP,RO
	113136	104417				TRAP	C1PNTF
	113140	062706	00G006			ADD	@6,SP
6931	113144			CALL	BINDEC @UCB12.24		
	113144	012725	050070			MOV	@UCB12.24,(R5).
	113150	004437	070656			JSR	R4,PREG14
	113154	007446				.WORD	BINDEC-ANCHOR
6932	113156			PRINTF	@CNTR07,@DECSTR		
	113156	012746	100526			MOV	@DECSTR,-(SP)
	113162	012746	062463			MOV	@CNTR07,-(SP)
	113166	012746	000002			MOV	@2,-(SP)
	113172	010600				MOV	SP,RO
	113174	104417				TRAP	C1PNTF
	113176	062706	000006			ADD	@6,SP
6933	113202			PRINTF	@CNTR08,UCB12.30		
	113202	013746	050074			MOV	UCB12.30,-(SP)
	113206	012746	062533			MOV	@CNTR08,(SP)
	113212	012746	000002			MOV	@2,(SP)
	113216	010600				MOV	SP,RO
	113220	104417				TRAP	C1PNTF
	113222	062706	000006			ADD	@6,SP
6934	113226			PRINTF	@CNTR09,UCB12.32		
	113226	013746	050076			MOV	UCB12.32,(SP)
	113232	012746	062606			MOV	@CNTR09,(SP)
	113236	012746	000002			MOV	@2,(SP)
	113242	010600				MOV	SP,RO
	113244	104417				TRAP	C1PNTF
	113246	062706	000006			ADD	@6,SP
6935	113252			CALL	BINDEC @UCB12.34		
	113252	012725	050100			MOV	@UCB12.34,(R5).
	113256	004437	070656			JSR	R4,PREG14
	113262	007446				.WORD	BINDEC ANCHOR
6936	113264			PRINTF	@CNTR10,@DECSTR		
	113264	012746	100526			MOV	@DECSTR,(SP)
	113270	012746	062657			MOV	@CNTR10,(SP)
	113274	012746	000002			MOV	@2,-(SP)
	113300	010600				MOV	SP,RO
	113302	104417				TRAP	C1PNTF
	113304	062706	000006			ADD	@6,SP
6937	113310			CALL	BINDEC @UCB12.40		
	113310	012725	050104			MOV	@UCB12.40,(R5).
	113314	004437	070656			JSR	R4,PREG14
	113320	007446				.WORD	BINDEC-ANCHOR
6938	113322			PRINTF	@CNTR11,@DECSTR		
	113322	012746	100526			MOV	@DECSTR,(SP)
	113326	012746	062716			MOV	@CNTR11,(SP)
	113332	012746	000002			MOV	@2,(SP)
	113336	010600				MOV	SP,RO
	113340	104417				TRAP	C1PNTF
	113342	062706	000006			ADD	@6,SP
6939	113346			CALL	BINDEC @UCB12.44		
	113346	012725	050110			MOV	@UCB12.44,(R5).
	113352	004437	070656			JSR	R4,PREG14

CVN	ADDR	OPCODE	DATA1	DATA2	DATA3	ASSEMBLY	OPERANDS
6940	113356	007446				PRINTF	@CNTR12,@DECSTR
	113360						
	113360	012746	100526				
	113364	012746	062766				
	113370	012746	000002				
	113374	010600					
	113376	104417					
	113400	062706	000006				
6941	113404					CALL	BINDEC @UCB12.50
	113404	012725	050114				
	113410	004437	070656				
	113414	007446					
6942	113416					PRINTF	@CNTR13,@DECSTR
	113416	012746	100526				
	113422	012746	063034				
	113426	012746	000002				
	113432	010600					
	113434	104417					
	113436	062706	000006				
6943	113442					CALL	BINDEC @UCB12.54
	113442	012725	050120				
	113446	004437	070656				
	113452	007446					
6944	113454					PRINTF	@CNTR14,@DECSTR
	113454	012746	100526				
	113460	012746	063101				
	113464	012746	000002				
	113470	010600					
	113472	104417					
	113474	062706	000006				
6945	113500					CALL	BINDEC @UCB12.60
	113500	012725	050124				
	113504	004437	070656				
	113510	007446					
6946	113512					PRINTF	@CNTR15,@DECSTR
	113512	012746	100526				
	113516	012746	063135				
	113522	012746	000002				
	113526	010600					
	113530	104417					
	113532	062706	000006				
6947	113536					CALL	BINDEC @UCB12.64
	113536	012725	050130				
	113542	004437	070656				
	113546	007446					
6948	113550					PRINTF	@CNTR16,@DECSTR
	113550	012746	100526				
	113554	012746	063177				
	113560	012746	000002				
	113564	010600					
	113566	104417					
	113570	062706	000006				
6949	113574					PRINTF	@CNTR17,UCB12.70
	113574	013746	050134				
	113600	012746	063245				
	113604	012746	000002				
	113610	010600					

  

.WORD	BINDEC ANCHOR
	MOV @DECSTR,(SP)
	MOV @CNTR12,-(SP)
	MOV @2,-(SP)
	MOV SP,R0
	TRAP C#PNTF
	ADD @6,SP
	MOV @UCB12.50,(R5)
	JSR R4,PREG14
.WORD	BINDEC ANCHOR
	MOV @DECSTR,-(SP)
	MOV @CNTR13,-(SP)
	MOV @2,-(SP)
	MOV SP,R0
	TRAP C#PNTF
	ADD @6,SP
	MOV @UCB12.54,(R5)
	JSR R4,PREG14
.WORD	BINDEC ANCHOR
	MOV @DECSTR,(SP)
	MOV @CNTR14,-(SP)
	MOV @2,(SP)
	MOV SP,R0
	TRAP C#PNTF
	ADD @6,SP
	MOV @UCB12.60,(R5)
	JSR R4,PREG14
.WORD	BINDEC ANCHOR
	MOV @DECSTR,(SP)
	MOV @CNTR15,-(SP)
	MOV @2,(SP)
	MOV SP,R0
	TRAP C#PNTF
	ADD @6,SP
	MOV @UCB12.64,(R5)
	JSR R4,PREG14
.WORD	BINDEC ANCHOR
	MOV @DECSTR,(SP)
	MOV @CNTR16,-(SP)
	MOV @2,(SP)
	MOV SP,R0
	TRAP C#PNTF
	ADD @6,SP
	MOV UCB12.70,-(SP)
	MOV @CNTR17,-(SP)
	MOV @2,-(SP)
	MOV SP,R0

```

113612 104417
113614 062706 000006
6950 113620          PRINTF  @CNTR18,UCB12.72
113620 013746 050136
113624 012746 063320
113630 012746 000002
113634 010600
113636 104417
113640 062706 000006
6951 113644          PRINTF  @CNTR19,UCB12.74
113644 013746 050140
113650 012746 063365
113654 012746 000002
113660 010600
113662 104417
113664 062706 000006
6952 113670          CLRB   P#NNUF
6953 113674          RTS    PC
6954
6955
6956
6957          ; ACTION ROUTINE TO PRINT OUT THE NODE TABLE
6958          ;
6959
6960 113676 105037 003220 002410 ACTSND: CLRB   P#NNUF
6961 113702 012737 002412          MOV    @NODTBL,SLOT          ;MOVE NODE TABLE ADDRESS INTO SLOT
6962 113710          CALL   FULSLT          ;SEE IF TABLE EMPTY
113710 004437 070656          JSR    R4,PREG14
113714 025556          .WORD FULSLT-ANCHOR
6963 113716 022737 177777 002410          CMP    @ 1,SLOT          ;IF YES, DON'T PRINT HEADER
6964 113724 001437          BEQ   20$
6965 113726          PRINTF @NTBMOR          ;PRINT NODE TABLE HEADER
113726 012746 053176          MOV    @NTBMOR,-(SP)
113732 012746 000001          MOV    @1,-(SP)
113736 010600          MOV    SP,RO
113740 104417          TRAP  C#PNTF
113742 062706 000004          ADD   @4,SP
6966 113746          10$: CALL   FULSLT          ;FIND LOCATION IN TABLE WITH AN ADDRESS
113746 004437 070656          JSR    R4,PREG14
113752 025556          .WORD FULSLT-ANCHOR
6967 113754 022737 177777 002410          CMP    @ 1,SLOT          ;CHECK IF AT END OF TABLE
6968 113762 001432          BEQ   30$          ;IF YES, RETURN
6969 113764          CALL   BINHEX SLOT,@6,@STRBUF ;ELSE, PUT ASCII ADDRESS INTO BUFFER
113764 012725 002324          MOV    @STRBUF,(R5)
113770 012725 000006          MOV    @6,(R5)
113774 013725 002410          MOV    SLOT,(R5)
114000 004437 070656          JSR    R4,PREG14
114004 005150          .WORD BINHEX-ANCHOR
6970 114006          CALL   PRTNOD          ;PRINT NODE TABLE ENTRY
114006 004437 070656          JSR    R4,PREG14
114012 026040          .WORD PRTNOD-ANCHOR
6971 114014 062737 000010 002410          ADD   @8,SLOT          ;INCR. SLOT TO POINT TO NEXT TABLE ENTRY
6972 114022 000751          BR    10$          ;CONTINUE UNTIL ALL ENTRIES PRINTED
6973 114024          20$: PRINTF @TABEMT,@NOD
114024 012746 053415          MOV    @NOD,(SP)
114030 012746 053346          MOV    @TABEMT,(SP)
114034 012746 000002          MOV    @2,(SP)

```

```

114040 010600
114042 104417
114044 062706 000006
6974 114050 000207 30$: RTS PC ;RETURN
6975
6976
6977 ;ACTION ROUTINE TO CLEAR A NODE SPECIFIED BY NODE LOGICAL NAME
6978 ;FROM THE NODE TABLE
6979
6980
6981 114052 ACTCNL: P$PUSH R2 ;SAVE R2
114052 010225 MOV R2,(R5);
6982 114054 013702 003214 MOV P$NUM,R2 ;PUT NODE LOGICAL NUMBER INTO R2
6983 114060 006302 ASL R2 ;MULTIPLY BY 3
6984 114062 006302 ASL R2 ;NODE TABLE ADDRESS =
6985 114064 006302 ASL R2 ; (LOG. NO. X 8) + #NODTBL
6986 114066 062702 002412 ADD #NODTBL,R2 ;ADD OFFSET
6987 114072 005022 CLR (R2); ;CLEAR ENTRY (8 BYTES)
6988 114074 005022 CLR (R2);
6989 114076 005022 CLR (R2);
6990 114100 005012 CLR (R2);
6991 114102 P$POP R2 ;RESTORE R2
114102 014502 MOV -(R5),R2
6992 114104 105037 003220 CLRB P$NNUF ;CLEAR NOTNUF FLAG
6993 114110 PRINTF #LOGDEL,P$NUM ;PRINT MESSAGE INDICATING DELETION
114110 013746 003214 MOV P$NUM,(SP)
114114 012746 054132 MOV #LOGDEL,(SP)
114120 012746 000002 MOV #2,(SP)
114124 010600 MOV SP,R0
114126 104417 TRAP C$PNTF
114130 062706 000006 ADD #6,SP
6994 114134 000207 RTS PC ;RETURN
6995
6996 ;ACTION ROUTINE TO SAVE NODE TABLE
6997
6998
6999 ACTSAV: P$PUSH R2,R3 ;SAVE R2 AND R3
114136 010325 MOV R3,(R5);
114136 010325 MOV R2,(R5);
114140 010225
7000 114142 012702 002412 MOV #NODTBL,R2 ;SET REGISTERS FOR COPYING
7001 114146 012703 002536 MOV #SAVTBL,R3 ;R2 = FROM, R3 = TO
7002 114152 PRINTF #UNSMMSG,#SAVFD ;PRINT TABLE SAVED MESSAGE
114152 012746 054324 MOV #SAVED,(SP)
114156 012746 054265 MOV #UNSMMSG,(SP)
114162 012746 000002 MOV #2,(SP)
114166 010600 MOV SP,R0
114170 104417 TRAP C$PNTF
114172 062706 000006 ADD #6,SP
7003 114176 000137 114260 JMP SAVCOM
7004
7005
7006 ;ACTION ROUTINE TO UNSAVE NODE TABLE
7007
7008
7009 114202 105037 003220 ACTUNS: CLRB P$NNUF ;CLEAR 'NOT ENOUGH' FLAG
7010 114206 P$PUSH R2,R3 ;SAVE R2 AND R3
114206 010325 MOV R3,(R5);

```

```

114210 010225
7011 114212 121427 000057          CMPB   (R4),#57          MOV     R2,(R5)+
7012 114216 001002          BNE    10$
7013 114220 000137 114272          JMP    QUIT
7014 114224 012703 002412          10$:  MOV   @NODTBL,R3      ;SET REGISTERS FOR COPYING
7015 114230 012702 002536          MOV   @SAVTBL,R2       ;R2 = FROM, R3 = TO
7016 114234          PRINTF @UNSMMSG,@RESTOR ;PRINT 'TABLE RESTORED' MESSAGE
114234 012746 054333          MOV   @RESTOR,-(SP)
114240 012746 054265          MOV   @UNSMMSG,-(SP)
114244 012746 000002          MOV   #2,-(SP)
114250 010600          MOV   SP,R0
114252 104417          TRAP  C$PNTF
114254 062706 000006          ADD   #6,SP
7017 114260 012701 000050          SAVCOM: MOV  @TBLEN,R1      ;MOVE TABLE LENGTH TO R1
7018 114264 012223          10$:  MOV  (R2)+,(R3)+      ;MOVE WORD
7019 114266 005301          DEC   R1               ;DECREMENT COUNTER
7020 114270 001375          BNE   10$              ;IF MORE, LOOP
7021 114272          QUIT: P$POP R2,R3     ; ELSE, RESTORE COUNTERS
114272 014502          MOV   -(R5),R2
114274 014503          MOV   -(R5),R3
7022 114276 105037 003220          CLRB  P$NNUF          ; CLEAR 'NOT ENOUGH' FLAG
7023 114302 000207          RTS   PC
7024
7025          ;
7026          ;ACTION ROUTINE TO CLEAR SUMMARY TABLE
7027          ;
7028
7029 114304 105037 003220          ACTCSU: CLRB P$NNUF      ;CLEAR 'NOT ENOUGH' COUNTER
7030 114310          P$PUSH R2             ;SAVE R2
114310 010225          MOV   R2,(R5)+
7031 114312 012701 000132          MOV   @STBLN,R1      ;MOVE TABLE LENGTH TO R1
7032 114316 012702 002714          MOV   @STATBL,R2     ;MOVE SUMMARY TABLE ADDRESS TO R2
7033 114322 005022          10$:  CLR  (R2)+           ;CLEAR FIRST WORD
7034 114324 005301          DEC   R1              ;SEE IF FINISHED
7035 114326 001375          BNE   10$             ; IF NO, DO MORE
7036 114330          PRINTF @TABCLR,@SUMM ; ELSE, PRINT 'TABLE CLEARED' MESSAGE
114330 012746 053422          MOV   @SUMM,-(SP)
114334 012746 054220          MOV   @TABCLR,-(SP)
114340 012746 000002          MOV   #2,-(SP)
114344 010600          MOV   SP,R0
114346 104417          TRAP  C$PNTF
114350 062706 000006          ADD   #6,SP
7037 114354          P$POP R2             ; AND RESTORE R2
114354 014502          MOV   -(R5),R2
7038 114356 000207          RTS   PC
7039
7040          ;
7041          ;ACTION ROUTINE TO CHECK FOR PA.S DEFAULT VALUE
7042          ;
7043
7044 114360          ACTDFT:
7045 114360 121427 000040          10$:  CMPB  (R4),#40      ;SEE IF SPACES
7046 114364 001002          BNE   20$             ; IF NO, CONT.
7047 114366 005204          INC   R4              ; ELSE, POINT TO NEXT CHAR
7048 114370 000773          BR    10$            ; AND CHECK AGAIN
7049 114372 121427 000000          20$:  CMPB  (R4),#0      ;SEE IF DEFAULT VALUE
7050 114376 001007          BNE   30$            ; IF NO, BR
114376 001007          BR    30$
114376 001007          BR    30$

```

```

7051 114400 012763 000030 000002      MOV    #30,2(R3)      ; IF YES, POINT R3 TO SKIP CHECK PASS COUNT
7052 114406 012737 000001 003214      MOV    #1,P#NUM      ; SET DEFAULT TO 1
7053 114414 000403                BR     40$           ; RETURN
7054 114416 012763 000004 000002 30$:  MOV    #4,2(R3)      ; POINT R3 TO CHECK FOR PASS COUNT
7055 114424 000207                40$:  RTS     PC
7056
7057
7058                                     ; ACTION ROUTINE TO READ A FILE FROM EXTERNAL MEDIA ONTO THE NODE TABLE
7059                                     ;
7060
7061 114426                ACTUSF:
7062 114426                P$PUSH R2            ; SAVE R2
114426 010225                CLR     R2           ; INITIALIZE R2 TO NODE TYPE 'TARGET'
7063 114430 005002                OPEN   CBOADR       ; OPEN FILE, NAME=ASCIZ STRING
7064 114432                MOV    CBOADR,RO    ;
114432 013700 002370                TRAP  C$OPEN
114436 104434                BCOMPLETE 10$      ; RETURN IF SUCCESSFUL
7065 114440                PRINTF #OPNERR,CBOADR ; ELSE PRINT "OPEN ERROR"
114440 103413                MOV    CBOADR,(SP)
114442 013746 002370                MOV    #OPNERR,-(SP)
114446 012746 116072                MOV    #2,(SP)
114452 012746 000002                MOV    SP,RO
114456 010600                TRAP  C$PNTF
114460 104417                ADD   #6,SP
114462 062706 000006                CLOSE
7067 114466                ; CLOSE FILE
114466 104435                TRAP  C$CLOS
7068 114470                10$: CALL   RDLIN        ; READ A LINE AT A TIME
114470 004437 070656                JSR    R4,PREG14
114474 024520                .WORD RDLIN ANCHOR
7069 114476 005737 116066                TST   BAD           ; SEE IF AN ERROR DURING READ
7070 114502 001064                BNE   40$           ; BR ON ERROR TO LEAVE
7071 114504 005737 116070                TST   EOF           ; SEE IF EOF BEFORE PROCESS
7072 114510 001411                BEQ   20$           ; IF VALID, PROCESS
7073 114512                PRINTF #EOFFND      ; ELSE SAY 'END OF FILE' AND LEAVE
114512 012746 116205                MOV    #EOFFND,-(SP)
114516 012746 0000J1                MOV    #1,(SP)
114522 010600                MOV    SP,RO
114524 104417                TRAP  C$PNTF
114526 062706 000004                ADD   #4,SP
7074 114532 000450                BR     40$
7075 114534                20$: PRINTF #PLINE,#FILLIN ; PRINT LINE READ FROM FILE
114534 012746 115542                MOV    #FILLIN,(SP)
114540 012746 116154                MOV    #PLINE,-(SP)
114544 012746 000002                MOV    #2,(SP)
114550 010600                MOV    SP,RO
114552 104417                TRAP  C$PNTF
114554 062706 000006                ADD   #6,SP
7076 114560                CALL   EDPACK #FILLIN,#ADRBUF,#6 ; PUT ADDRESS INTO BINARY
114560 012725 000006                MOV    #6,(R5)
114564 012725 002316                MOV    #ADRBUF,(R5)
114570 012725 115542                MOV    #FILLIN,(R5)
114574 004437 070656                JSR    R4,PREG14
114600 004600                .WORD EDPACK ANCHOR
7077 114602                P$POP R1           ; CHECK RESULTS
114602 014501                MOV    (R5),R1

```

```

7078 114604 001411
7079 114606 012746 053022
      114606 012746 000001
      114612 012746 000001
      114616 010600
      114620 104417
      114622 062706 000004
7080 114626 000412
7081 114630 110237 002406
7082 114634 105102
7083 114636 142702 000376
7084 114642 004437 070656
      114642 025410
7085 114650 014501
      114650 000706
7086 114652 000706
7087 114654 014502
      114654 105037 003220
7088 114656 105037 003220
7089 114662 012746 054333
      114662 012746 054265
      114672 012746 000002
      114676 010600
      114700 104417
      114702 062706 000006
7090 114706 000207
      114706 000207

      BEQ 30$
      PRINTF @CADERR

      BR 40$
      MOV R2,NODTY
      COMB R2
      BICB @376,R2
      CALL ENTRND

      P$POP R1

      BR 10$
      P$POP R2

      CLRB P$NNUF
      PRINTF @UNSMMSG,@RESTOR

      RETURN

      ; IF OK, BR
      ; ELSE PRINT ERROR MESSAGE
      MOV @CADERR, (SP)
      MOV @1, (SP)
      MOV SP,R0
      TRAP C$PNTF
      ADD @4,SP

      ; AND EXIT
      ;SET UP NODE TYPE
      ;SWITCH TYPE FOR NEXT TIME

      ;ENTER IN NODE TABLE
      JSR R4,PREG14
      .WORD ENTRND-ANCHOR

      ;GET RESULTS
      MOV (R5),R1

      ;READ MORE ADDRESS
      ;RESTORE R2
      MOV -(R5),R2

      ;CLEAR 'NOT ENOUGH' FLAG
      ;PRINT 'TALBE RESTORED' MESSAGE
      MOV @RESTOR,-(SP)
      MOV @UNSMMSG,(SP)
      MOV @2,(SP)
      MOV SP,R0
      TRAP C$PNTF
      ADD @6,SP

      RTS PC

```



7092  
 7093  
 7094  
 7095  
 7096  
 7097  
 7098  
 7099  
 7100  
 7101  
 7102  
 7103  
 7104  
 7105  
 7106  
 7107  
 7108  
 7109  
 7110  
 7111  
 7112  
 7113  
 7114  
 7115  
 7116  
 7117  
 7118  
 7119  
 7120  
 7121  
 7122  
 7123  
 7124  
 7125  
 7126  
 7127  
 7128  
 7129  
 7130  
 7131  
 7132  
 7133  
 7134  
 7135 114710  
 7136 114710 005237 003202  
 7137 114714  
 7138 114714 013701 003750  
 7139 114720 062701 000010  
 7140 114724  
 7141 114724  
 114724 104422  
 7142 114726 020137 003750  
 7143 114732 002016  
 7144 114734 005001  
 7145 114736 012703 004032  
 7146 114742 012704 047674  
 7147 114746 012314

.SBTTL RESPONDER TEST

```

: **
: FUNCTIONAL DESCRIPTION:
:     THIS TEST FORWARDS LOOP-SERVER FORWARD REQUEST PACKETS,
:     AND TRANSMITS A CONSOLE ID EVERY 8 TO 9 MINUTES.
: CALLING SEQUENCE
:     NONE
: INPUT
:     NONE
: IMPLICIT INPUT
:     NONE
: OUTPUT
:     NONE
: IMPLICIT OUTPUT
:     NONE
: COMPLETION CODES
:     NONE
: SIDE EFFECTS
:     NONE
: REGISTERS USED
:     NONE
: DEBUG
:     NONE
: --
ACTRSP:
10$: INC      RSPFLG           ; INDICATE WE ARE IN LISTEN MODE
      MOV     TIMMIN,R1       ; GET CURRENT NUMBER OF MINUTES
      ADD     #8.,R1          ; AND ADD 9 MINUTES TO IT
20$:  BREAK                  ; ALLOW OPERATOR TO RETAKE CONTROL
      TRAP   C$BRK           ; HAS NINE MINUTES GONE BY?
      BGE    30$             ; BRANCH IF NOT
      CLR    R1              ; INDICATE THAT TIME HAS GONE BY
      MOV    #MCSTAD,R3      ; CONSOLE ID MULTICAST ADDRESS
      MOV    #IDTDAT,R4     ; ADDRESS OF CONSOLE ID MESSAGE
      MOV    (R3)+,(R4)     ; SET MULTICAST ADDRESS
    
```

7148	114750	012364	000002		MOV	(R3)+,2(R4)		; AGAIN
7149	114754	012364	000004		MOV	(R3)+,4(R4)		; AND LAST TWO BYTES
7150	114760	013702	024220		MOV	XRGCUR,R2		; GET CURRENT XMIT DESCRIPTOR
7151	114764	000137	115260		JMP	75\$		; GO XMIT THE CONSOLE ID
7152	114770			30\$:				
7153	114770				CALL	RECEVE		; SEE IF THERE IS A BUFFER
	114770	004437	070656					JSR R4,PREG14
	114774	003044						.WORD RECEVE-ANCHOR
7154	114776				P\$POP	R2		; GET COUNT OF VALID RECEIVES
	114776	014502						MOV -(R5),R2
7155	115000	001751			BEQ	20\$		; BRANCH IF NONE
7156	115002	013702	024226		MOV	RRGNXT,R2		; ELSE, GET ADDRESS OF DESCRIPTOR
7157	115006	032762	040000	000010	BIT	#MRERRS,STAT1(R2)		; ARE ANY ERRORS INDICATED
7158	115014	001411			BEQ	40\$		; BRANCH IF NOT
7159	115016	032762	010000	000010	BIT	#MDISC,STAT1(R2)		; ELSE, ARE THEY VALID?
7160	115024	001405			BEQ	40\$		; BRANCH IF NOT
7161	115026				ERRHRD	58,EMSG58		; ELSE, REPORT ERROR
	115026	104456						TRAP C\$ERHRD
	115030	000072						.WORD 58
	115032	066211						.WORD EMSG58
	115034	000000						.WORD 0
7162	115036	000555			BR	110\$		; DO IT ALL AGAIN
7163	115040			40\$:				
7164	115040	016203	000004		MOV	LOADD(R2),R3		; GET THE DATA BUFFER ADDRESS
7165	115044	016304	000014		MOV	PROTOT(R3),R4		; GET PROTOCOL TYPE CODE
7166	115050	023704	050306		CMP	PROTO0,R4		; LOOP SERVER MESSAGE?
7167	115054	001060			BNE	70\$		; BRANCH IF NOT
7168	115056	010304			MOV	R3,R4		; POINT TO BEGINNING OF BUFFER
7169	115060	062704	000016		ADD	@LDSKIP,R4		; BUMP THE POINTER
7170	115064	062404			ADD	(R4)+,R4		; POINT TO FUNCTION CODE
7171	115066	022724	000002		CMP	#FORWRD,(R4)+		; IS THIS A FORWARD MESSAGE?
7172	115072	001405			BEQ	50\$		; BRANCH IF YES
7173	115074				ERRHRD	59,EMSG59		; ELSE, REPORT UNINVITED REPLY RCVD
	115074	104456						TRAP C\$ERHRD
	115076	000073						.WORD 59
	115100	066267						.WORD EMSG59
	115102	000000						.WORD 0
7174	115104	000532			BR	110\$		; DO IT ALL AGAIN
7175	115106			50\$:				
7176	115106	012423			MOV	(R4)+,(R3)+		; FILL IN NEW DESTINATION ADDRESS
7177	115110	012423			MOV	(R4)+,(R3)+		; AND AGAIN
7178	115112	012423			MOV	(R4)+,(R3)+		; AND AGAIN
7179	115114	012704	0040		MOV	#PHYADR,R4		; POINT TO OUR PHYSICAL ADDRESS
7180	115120	012423			MOV	(R4)+,(R3)+		; FILL IN SOURCE ADDRESS
7181	115122	012423			MOV	(R4)+,(R3)+		; AND AGAIN
7182	115124	012423			MOV	(R4)+,(R3)+		; AND AGAIN
7183	115126	005723			TST	(R3)+		; POINT PAST TYPE CODE
7184	115130	062713	000010		ADD	#10,(R3)		; UPDATE THE SKIP COUNT TO NEXT FUNC CODE
7185	115134	016203	000010		MOV	STAT1(R2),R3		; GET STATUS WORD 2
7186	115140	042703	174377		BIC	#174377,R3		; CLEAR ALL BUT RCVD BYTE CNT BITS 8 10
7187	115144	116204	000012		MOV	STAT2(R2),R4		; GET RCVD BYTE CNT 0 7
7188	115150	042704	177400		BIC	#177400,R4		; KILL ALL UNWANTED BITS
7189	115154	050403			BIS	R4,R3		; COMBINE THE TWO FOR TOTAL BYTE COUNT
7190	115156	062703	000074		ADD	#60.,R3		; ADD MISSING BYTES
7191	115162	006203			ASR	R3		; CONVERT TO A WORD COUNT
7192	115164	005403			NEG	R3		; MAKE IT TWO'S COMPLEMENT
7193	115166	013704	024220		MOV	XRGCUR,R4		; GET CURRENT XMIT DESCRIPTOR

```

7194 115172 010364 000006      MOV      R3,WRDCNT(R4)      ; PUT IT INTO THE XMIT DESCRIPTOR
7195 115176 016404 000004      MOV      LOADD(R4),R4      ; POINT TO THE XMIT DATA BUFFER
7196 115202 016202 000004      MOV      LOADD(R2),R2      ; GET THE RECEIVE DATA BUFFER
7197 115206                                60$:
7198 115206 012224                                MOV      (R2)+,(R4)+      ; START FILLING TN THE DATA BUFFER
7199 115210 005203                                INC      R3                ; BUMP WORDCOUNT
7200 115212 001375                                BNE     60$                ; DO IT UNTIL THE PACKET HAS BEEN .OPIED
7201 115214 000434                                BR      90$                ; GO XMIT THE PACKET
7202 115216                                70$:
7203 115216 022763 000005 000020      CMP      @IDFUNC,RIFUNC(R3) ; IS THIS A REQUEST ID CONSOLE T PE?
7204 115224 001062                                BNE     110$                ; IF NOT, THROW IT OUT
7205 115226 012704 047674                                MOV      @IDTDAT,R4        ; POINT TO ID TRANSMIT DATA
7206 115232 016364 000006                                MOV      SOURCC(R3),DESTIN(R4) ; FILL IN DESTINATION OF ID MESSAGE
7207 115240 016364 000010 000002      MOV      SOURCC+2(R3),DESTIN+2(R4) ; FILL IN DESTINATION OF ID MESSAGE
7208 115246 016364 000012 000004      MOV      SOURCC+4(R3),DESTIN+4(R4) ; FILL IN DESTINATION OF ID MESSAGE
7209 115254 013702 024220                                MOV      XRGCUR,R2        ; GET XMIT DESCRIPTOR
7210 115260                                75$:
7211 115260 013703 047770                                MOV      IDTSIZ,R3        ; SET SIZE OF XMISSION
7212 115264 006203                                ASR     R3                ; CHANGE TO WORD COUNT
7213 115266 005403                                NEG     R3                ; TWOS COMPLEMENT IT
7214
7215 115270 010362 000006      MOV      R3,WRDCNT(R2)      ; SET XFER SIZE
7216 115274 016202 000004      MOV      LOADD(R2),R2      ; GET XMIT DATA BUFFER ADDRESS
7217 115300                                80$:
7218 115300 012422                                MOV      (R4)+,(R2)+      ; PUT ID DATA INTO XMIT DATA BUFFER
7219 115302 005203                                INC      R3                ; BUMP COUNT
7220 115304 001375                                BNE     80$                ; DO UNTIL COMPLETE
7221 115306                                90$:
7222 115306      CALL      XMIT      @XMTDAT      ; TRANSMIT THE PACKET
                                MOV      @XMTDAT,(R5)+
                                JSR      R4,PREG14
                                .WORD    XMIT-ANCHOR
7223 115320      P$POP     R2                ; GET STATUS
                                MOV      (R5),R2
7224 115322 001404      BEQ      100$              ; BRANCH IF OKAY
7225 115324      ERRHRD   62,EMSG62
                                TRAP    C$ERRHRD
                                .WORD    62
                                .WORD    EMSG62
                                .WORD    0
7226 115334                                100$:
7227 115334      CALL      GETXNX  @1,@XRGNXT      ; UPDATE XMIT RING POINTERS
                                MOV      @XRGNXT,(R5)+
                                MOV      @1,(R5)+
                                JSR      R4,PREG14
                                .WORD    GETXNX ANCHOR
7228 115352 005701      TST     R1                ; WAS TRANSMISSION A 9 MINUTE SYS ID?
7229 115354 001006      BNE     110$              ; BRANCH IF NOT, RECEIVE POINTER NEEDS UPDATING
7230 115356 013701 003750      MOV      TIMMIN,R1        ; ELSE GET MINUTES SINCE START
7231 115362 062701 000010      ADD     @8.,R1            ; AND ADD NINE MINUTES TO IT
7232 115366 000137 114724      JMP     20$                ; NOTHING RECEIVED CAUSED XMIT. DON T UPDATE RCV PN*
R
7233 115372                                110$:
7234 115372      CALL      GETRNX  @1,@RRGNXT      ; UPDATE RECEIVE RING POINTERS
                                MOV      @RRGNXT,(R5)+
                                MOV      @1,(R5)+
                                JSR      R4,PREG14
                                .WORD    GETRNX ANCHOR
115376 012725 000001
115402 004437 070656
115406 006450

```

L14

CVNIAAO DEQNA NI EXERCISER DIAG MACRO M1200 22 MAR 84 13:12 PAGE 53 3  
RESPONDER TEST

SEQ 180

7235 115410 000137 114724  
7236

JMP 204

; DO IT ALL AGAIN

```

7238 .SBTTL READ LINE OF OPENED FILE
7239 ;
7240 ; THIS ROUTINE GETS BYTES FROM AN OPENED FILE UNTIL A CR IS ENCOUNTERED
7241 ; "EOF" AND "BAD" FLAGS ARE SET IF END OF-FILE OR ERRORS ARE ENCOUNTERED
7242 ;
7243 ; NOTE: ASSUMING A ASCII TEXT FILE IS BEING READ, FOR EXAMPLE:
7244 ; AA-00 03-00-01-AB<CR><LF>
7245 ; "
7246 ; AA-00 03-00-01-AB<CR><LF>
7247 ;
7248 ; WHAT YOU SEE READ BYTE BY BYTE IS:
7249 ; "A..-AB<CR><LF>A..-AB<CR><LF>..<O><O><O>.....???"
7250 ; SO I MADE ASSUMPTION THAT SINCE SEE "O-PADDING" AFTER LAST CHAR TO
7251 ; END-OF-FILEBLOCK, ANY CHARACTER THAT IS NOT "SPACE OR GREATER" OR A
7252 ; <CR> OR <LF> THEN I'LL TAKE THAT AS END-OF-FILE(TEXT), SET EOF FLAG
7253 ; AND LEAVE.
7254 ;
7255 ; INPUTS:
7256 ; FILLIN BUFFER TO HOLD LINE OF BYTES READ FROM OPENED FILE
7257 ; (CR NOT INCLUDED, O-BYTE TERMINATED)
7258 ;
7259 ; OUTPUTS:
7260 ; BAD IF NON-ZERO, ERROR IN READING A BYTE FROM FILE
7261 ; EOFF IF NON-ZERO, END OF FILE WAS ENCOUNTERED
7262 ; FILLIN ASCII STRING THAT WAS READ AS CHAR-CR-LF STRING
7263 ; (CR-LF REMOVED)
7264 115414 012702 115542 RDLIN: MOV #FILLIN,R2 ;POINT R2 TO A LINE BUFFER
7265 115420 005037 116066 CLR BAD ;CLEAR FLAGS
7266 115424 005037 116070 CLR EOFF
7267 115430 104426 104426 104426 104426 104426 104426 104426 104426 104426
7268 115434 103414 103414 103414 103414 103414 103414 103414 103414 103414
7269 115436 012746 116126 000001 000001 000001 000001 000001 000001 000001
7270 115456 012737 177777 116066 MOV #-1,BAD ; SET BAD-TRY FLAG AND LEAVE
7271 115464 000416 000416 000416 000416 000416 000416 000416 000416 000416
7272 ;
7273 115466 122712 000015 000015 000015 000015 000015 000015 000015 000015
7274 115472 001756 001756 001756 001756 001756 001756 001756 001756 001756
7275 115474 122712 000012 000012 000012 000012 000012 000012 000012 000012
7276 115500 001410 001410 001410 001410 001410 001410 001410 001410 001410
7277 115502 122712 000040 000040 000040 000040 000040 000040 000040 000040
7278 ;
7279 115506 101002 101002 101002 101002 101002 101002 101002 101002 101002
7280 115510 005202 005202 005202 005202 005202 005202 005202 005202 005202
7281 115512 000746 000746 000746 000746 000746 000746 000746 000746 000746
7282 ;
7283 115514 012737 177777 116070 40%: MOV # 1,EOFF ;IF YES, TERMINATE INPUT BUFF
7284 ; AND SET EOF FLAG
7285 115522 105012 105012 105012 105012 105012 105012 105012 105012 105012
7286 115524 105012 105012 105012 105012 105012 105012 105012 105012 105012

```

RTS PC

115524 000207

7287

7288 115526

7289 115542

7290 115746

7291 116066 000000

7292 116070 000000

7293

7294 116072 045 116 045

116075 101 077 125

116100 116 101 102

116103 114 105 040

116106 124 117 040

116111 117 120 105

116114 116 040 042

116117 045 124 045

116122 101 042 077

116125 000

7295 116126 045 116 045

116131 101 077 106

116134 111 114 105

116137 040 122 105

116142 101 104 040

116145 105 122 122

116150 117 122 077

116153 000

7296 116154 045 116 045

116157 101 106 111

116162 114 105 040

116165 114 111 116

116170 105 040 127

116173 101 123 072

116176 045 116 045

116201 124 045 116

116204 000

7297 116205 045 116 045

116210 101 105 116

116213 104 055 117

116216 106 055 106

116221 111 114 105

116224 040 106 117

116227 125 116 104

116232 054 040 106

116235 111 114 105

116240 040 122 105

116243 101 104 000

7298

7299

7300

7301

7302

7303

7304

7305

7306

7307

7308

FILENM: .BLKB 12. ;BUFFER FOR FILE NAME  
FILLIN: .BLKB 132. ;BUFFER FOR SINGLE LINE READ FROM FILE  
MATCH: .BLKB 80. ;BUFFER FOR WORD TO MATCH FROM FILE  
BAD: .WORD 0 ;ERROR/NOT-FOUND FLAG WORD  
EJFF: .WORD 0 ;END-OF-FILE FLAG (<>0 = EOF)

OPNERR: .ASCIZ /#N#A?UNABLE TO OPEN "#T#A"?/

RDERR: .ASCIZ /#N#A?FILE READ ERROR?/

PLINE: .ASCIZ /#N#A?FILE LINE WAS:#N#T#N/

EOFFND: .ASCIZ /#N#A?END-OF-FILE FOUND, FILE READ/

; --  
; SELMSG OPERATOR SELECTED MESSAGE STORAGE  
;  
; THIS ROUTINE WILL TAKE THE OPERATOR SELECTED MESSAGE FROM THE COMMAND  
; LINE INPUT STRING BUFFER AND PUT IT INTO A BUFFER AT LOCATION OPSLBF.  
;  
; INPUTS - P1 ADDRESS OF OPERATOR SELECTED MESSAGE IN  
; INPUT STRING  
;  
; EXPLICIT OUTPUTS - NONE  
; IMPLICIT OUTPUTS THE BUFFER AT OPSLBF WILL CONTAIN THE ASCII

```

7309                                     ;
7310                                     ;
7311                                     ; SUBORDINATE ROUTINES
7312                                     ; CALLING PROCEEDURE
7313                                     ; REGISTER USAGE -
7314                                     ;
7315                                     ;
7316                                     ; *
7317                                     ;
7318 116246 SELMSG: P#POP R1                ;PUT ADDRESS OF OPR. SEL ASCII STRING INTO R1
      116246 014501                                MOV    -(R5),R1
7319 116250 012702 003632                    ;PUT ADDRESS OF OUTPUT BUFFER INTO R2
7320 116254 005003                                CLR    R3
7321 116256 105711                                10#: TSTB (R1)
7322 11 260 001404                                ;CLEAR CHARACTER COUNTER
7323 116262 112122                                ;CHECK FOR END OF STRING
7324 116264 005203                                ;COUNT NUMBER OF CHARACTERS IN INPUT BUFFER
7325 116266 000137 116256                    ;GO TO 20# IF END
7326 116272 112712 000000                    ;ELSE, MOVE BYTE TO OUTPUT BUFFER
7327 116276 010337 003356                    ;COUNT NUMBER OF CHARACTERS IN INPUT BUFFER
7328 116302                                ;GO DO MORE CHARACTERS
      116302 000207                                ;PUT ZERO AT END OF OUTPUT BUFFER
                                                    ;STORE NUMBER OF CHARACTERS FOR USE IN BUF. BUILDING
                                                    RTS    PC
7329
7330                                     ; *
7331                                     ; ENTRND                                ENTER NODE IN TABLE
7332                                     ;
7333                                     ; THIS ROUTINE ENTERS A NODE INTO THE NODE TABLE
7334                                     ;
7335                                     ; INPUTS                                NONE
7336                                     ; EXPLICIT OUTPUTS                    P1 ZERO IF SUCCESSFUL, 1 IF TABLE FULL
7337                                     ; IMPLICIT OUTPUTS                    THE ADDRESS CONTAINED IN ADRBUF WILL BE
7338                                     ;                                     ADDED TO THE NODE TABLE IN THE FIRST
7339                                     ;                                     AVAILABLE SLOT WITH THE NODE TYPE CONTAINED
7340                                     ;                                     IN NODTY (TARGET OR ASSIST)
7341                                     ; SUBORDINATE ROUTINES                    FINDSL FIND EMPTY SLOT IN TABLE
7342                                     ; CALLING PROCEEDURE                    CALL ENTRND
7343                                     ; P#POP P1                                ;OUTPUT GOOD/BAD RESULT
7344                                     ;
7345                                     ; *
7346                                     ;
7347 116304 ENTRND: CALL FINDSL                ;FIND AVAILABLE SLOT IN TABLE
      116304 004437 070656                    JSR    R4,PREG14
      116310 025466                    .WORD FINDSL ANCHOR
7348 116312 P#POP R1                            ;CHECK IF TABLE FULL
7349 116314 001403                                BEQ    10#
7350 116316 P#PUSH # 1                            ;IF NOT FULL BR TO 10#
      116316 012725 177777                    ;ELSE PUT FULL INDICATION ON STACK
                                                    MOV    # 1,(R5)
7351 116322 000416                                ;RETURN
7352 116324 012703 000006                    10#: MOV    #6,R3
7353 116330 013701 002410                    ;SET INCR. COUNTER TO 6 (BYTES)
7354 116334 012702 002316                    ;MOV ADDRESS OF AVAILABLE SLOT TO R1
7355 116340 112221                                ;MOV ADDRESS OF NODE ADDRESS TO R2
7356 116342 005303                                20#: MOV    (R2),.(R1)
7357 116344 001375                                ;MOV BYTE OF ADDRESS
      116344 005201                                ;DECR. COUNTER
      113711 002406                    DEC    R3
7358 116346                                ;CONTINUE UNTIL 6 BYTES TRANSFERED
7359 116350                                ;SET POINTER TO NODE TYPE LOCATION
      113711 002406                    INC    R1
      113711 002406                    MOV    NODTY,(R1)
      113711 002406                    ;MOVE NODE TYPE INTO TABLE

```

```

7360 116354          P&PUSH  #0          ;PUT ADDRESS ADDED INDICATION ON STACK
      116354 012725 000000          MOV      #0,(R5).
7361 116360          30$: RETURN          ;RETURN
      116360 000207          RTS      PC
7362
7363 : - *
7364 :
7365 : FINDSL          FIND EMPTY SLOT IN NODE TABLE
7366 :
7367 : INPUTS          NONE
7368 : EXPLICIT OUTPUTS NONE
7369 : IMPLICIT OUTPUTS
      THE ADDRESS OF THE FIRST AVAILABLE SLOT IN THE
7370 :          NODE TABLE WILL BE LOCATED IN SLOT. THE
7371 :          PARAMETER STACK WILL CONTAIN -1 IF THE NODE
7372 :          TABLE IS FULL AND 0 IF AN EMPTY SLOT WAS FOUND
7373 : SUBORDINATE ROUTINES NONE
7374 : CALLING PROCEDURE CALL FINDSL
7375 :          P&POP P1          ; 1 IF FULL/ 0 IF SLOT AVAILABLE
7376 :
7377 : - *
7378 116362 012702 002412 FINDSL: MOV      #NODTBL,R2          ;MOVE ADDRESS OF NODE TABLE TO R2
7379 116366 022712 000000 10$: CMP      #0,(R2)          ;SEE IF SLOT EMPTY
7380 116372 001422          BEQ      20$          ;IF YES, BR 20$
7381 116374 062702 000010          ADD      #8.,R2          ;ELSE MOVE POINTER TO NEXT ENTRY LOC.
7382 116400 022712 177777          CMP      #-1,(R2)          ;SEE IF AT END OF TABLE
7383 116404 001370          BNE     10$          ;IF NOT, CONTINUE LOOKING
7384 116406          PRINTF  #TABFUL,#NOD          ;ELSE, PRINT TABLE FULL MESSAGE
      116406 012746 053415          MOV      #NOD,(SP)
      116412 012746 053274          MOV      #TABFUL,(SP)
      116416 012746 000002          MOV      #2,(SP)
      116422 010600          MOV      SP,R0
      116424 104417          TRAP    C&PNTF
      116426 062706 000006          ADD      #6,SP
7385 116432          P&PUSH  #-1          ;PUT TABLE FULL INDICATION ON STACK
      116432 012725 177777          MOV      #-1,(R5).
7386 116436 000404          BR      30$          ;RETURN
7387 116440 010237 002410 20$: MOV      R2,SLOT          ;MOVE ADDRESS OF EMPTY LOC. INTO SLOT
7388 116444          P&PUSH  #0          ;PUT LOC. FOUND INDICATION ON STACK
      116444 012725 000000          MOV      #0,(R5).
7389 116450          30$: RETURN          ;RETURN
      116450 000207          RTS      PC
7390
7391 : - *
7392 :
7393 : FULSLT          FULL SLOT ROUTINE
7394 :
7395 : THIS ROUTINE FINDS A LOCATION IN THE TABLE WHERE A NODE PHYSICAL
7396 : ADDRESS EXISTS. IT IS USED WHEN PRINTING OUT THE NODE TABLE.
7397 :
7398 : INPUTS          NONE
7399 : EXPLICIT OUTPUTS NONE
7400 : IMPLICIT OUTPUTS
      THE LOCATION SLOT WILL CONTAIN THE PHYSICAL
7401 :          ADDRESS OF A NODE TABLE ENTRY. SLOT WILL
7402 :          CONTAIN 1 WHEN POINTING TO THE END OF THE
7403 :          NODE TABLE
7404 : SUBORDINATE ROUTINES NONE
7405 : CALLING PROCEDURE CALL FULSLT
  
```



```

7406
7407
7408 116452 013701 002410      FULSLT: MOV     SLOT,R1      ;MOVE SLOT LOCATION TO R1
7409 116456 022711 000000      10$:  CMP     #0,(R1)      ;CHECK IF EMPTY
7410 116462 001412                BEQ     30$                ;IF YES, BR 30$
7411 116464 022711 177777      CMP     #1,(R1)          ;SEE IF END OF NODE TABLE
7412 116470 001403                BEQ     20$                ;IF YES, BR 20$
7413 116472 010137 002410      MOV     R1,SLOT          ;ELSE PUT EMPTY LOC. ADDRESS INTO SLOT
7414 116476 000407                BR      40$                ;RETURN
7415 116500 012737 177777 002410 20$:  MOV     #1,SLOT          ;PUT 1 INTO SLOT TO SHOW END OF TABLE
7416 116506 000403                BR      40$                ;RETURN
7417 116510 062701 000010      30$:  ADD     #8.,R1          ;INCR. POINTER TO NEXT LOCATION
7418 116514 000760                BR      10$                ;CHECK NEXT LOC.
7419 116516 000207      40$:  RETURN                    ;RETURN
                                           RTS      PC
7420 ;
7421 ;
7422 116520      CHECK:
7423 116520 010146      MOV     R1,-(SP)
7424 116522 010346      MOV     R3,-(SP)
7425 116524 010446      MOV     R4,-(SP)
7426 116526 010546      MOV     R5,-(SP)
7427
7428 116530 005005      CLR     R5
7429 116532 012701 177776      3$:  MOV     #2.,R1
7430 116536
7431 116536 062701 000002      ADD     #2.,R1
7432 116542 022761 000000 002412  CMP     #0.,NODTBL(R1)
7433 116550 001772      BEQ     3$
7434
7435 116552 062701 000006      ADD     #6.,R1
7436 116556 022761 000000 002412  CMP     #0.,NODTBL(R1)
7437 116564 001401      BEQ     5$
7438 116566 000413      BR      7$
7439 116570      5$:
7440 116570 016104 002412      MOV     NODTBL(R1),R4
7441 116574 022761 000400 002412  CMP     #400,NODTBL(R1)
7442 116602 001013      BNE     10$
7443
7444 116604 062705 000001      ADD     #1,R5
7445 116610 022705 000002      CMP     #2,R5
7446 116614 001007      BNE     12$
7447 116616      7$:
7448 116616 005061 002412      CLR     NODTBL(R1)
7449 116622 104457      ERRSOFT 69,MSG69      ; REPORT IT
                                           TRAP    C$ERRSOFT
                                           .WORD  69
                                           .WORD  MSG69
                                           .WORD  0
116622 000105
116626 067204
116630 000000
7450 116632      10$:
7451 116632 005005      CLR     R5
7452 116634      12$:
7453 116634 062701 000002      ADD     #2.,R1
7454 116640 022761 177777 002412  CMP     #1,NODTBL(R1)
7455 116646 001407      BEQ     EXIT1
7456 116650 022761 000000 002412  CMP     #0.,NODTBL(R1)
7457 116656 001766      BEQ     12$

```

7458  
 7459 116660 062701 000006  
 7460 116664 000741  
 7461 116666  
 7462 116666 012605  
 7463 116670 012604  
 7464 116672 012603  
 7465 116674 012601  
 7466 116676 000207

```

ADD    #6.,R1
BR     5$
EXIT1:
MOV    (SP)+,R5
MOV    (SP)+,R4
MOV    (SP)+,R3
MOV    (SP)+,R1
RETURN
  
```

7467  
 7468  
 7469  
 7470  
 7471  
 7472  
 7473  
 7474  
 7475  
 7476  
 7477  
 7478  
 7479  
 7480  
 7481  
 7482

```

RTS    PC
;-----
;-----
; *
;      CMPADR                COMPARE TWO ADDRESSES
;
;      THIS ROUTINE COMPARES TWO SIX BYTE STRINGS
;
;      INPUTS                 P1 - ADDRESS OF FIRST STRING
;                           P2 - ADDRESS OF SECOND STRING
;      OUTPUTS                P3  0 - COMPARISON/ 1 - NO COMPARISON
;
;      CALLING PROCEDURE     CALL  CMPADR P1,P2
;                           P$POP P3
;
;      :---*
  
```

7483 116700  
 116700 014502  
 116702 014503  
 7484 116704 022223  
 7485 116706 001006  
 7486 116710 022223  
 7487 116712 001004  
 7488 116714 021213  
 7489 116716 001002  
 7490 116720 005001  
 7491 116722 000402  
 7492 116724 012701 177777  
 7493 116730  
 116730 010125  
 116732 000207

```

CMPADR: P$POP  R2,R3      ;PUT ADDRESS OF STRING TO BE COMPARED IN R2 AND R3
                                MOV    (R5),R2
                                MOV    (R5),R3
CMP    (R2)+,(R3)+      ;DO FIRST TWO BYTES COMPARE
BNE    10$              ; IF NO, EXIT
CMP    (R2)+,(R3)+      ;DO SECOND TWO BYTES COMPARE
BNE    10$              ; IF NO, EXIT
CMP    (R2),(R3)        ;DO LAST TWO BYTES COMPARE
BNE    10$              ; IF NO, EXIT
CLR    R1                ;PUT COMPARISON OK INDICATOR IN R1
BR     20$
10$:   MOV    #-1,R1      ;PUT NO COMPARISON INDICATOR IN R1
20$:   RETURN  R1
                                MOV    R1,(R5)+
                                RTS    PC
  
```

7494  
 7495  
 7496  
 7497  
 7498  
 7499  
 7500  
 7501  
 7502  
 7503

```

; *
;      PRNOD                PRINT NODE TABLE
;
;      INPUTS                 NONE
;      EXPLICIT OUTPUTS      NONE
;      IMPLICIT OUTPUTS      ONE ENTRY IN THE NODE TABLE WILL BE PRINTED
;      SUBORDINATE ROUTINES  NONE
;      CALLING SEQUENCE      CALL PRNOD
  
```

7504 116734  
 116734 012746 002324  
 116740 012746 053144  
 116744 012746 000002  
 116750 010600  
 116752 104417

```

PRNOD: PRINTF  #NODADR,#STRBUF      ;PRINT NODE ADDRESS
                                MOV    #STRBUF,(SP)
                                MOV    #NODADR,(SP)
                                MOV    #2,(SP)
                                MOV    SP,R0
                                TRAP   C$PNTF
  
```

READ LINE OF OPENED FILE

```

116754 062706 000006
7505 116760 013702 002410      MOV     SLOT,R2           ;MOVE SLOT ADDRESS INTO R2      ADD     #6,SP
7506 116764 162702 002412      SUB     #NODTBL,R2       ;CALCULATE NODE LOGICAL NAME
7507 116770 006202              ASR     R2                ;USING: LOG. NO. "
7508 116772 006202              ASR     R2                ;(SLOT #NODTAB)/8
7509 116774 006202              ASR     R2
7510 116776 010246              PRINTF  #LOGNAM,R2       ;PRINT LOGICAL NAME
116776 010246
117000 012746 053154              MOV     R2,(SP)
117004 012746 000002              MOV     #LOGNAM,(SP)
117010 010600              MOV     #2,-(SP)
117012 104417              MOV     SP,R0
117014 062706 000006              TRAP   C$PNTF
7511 117020 013702 002410      MOV     SLOT,R2           ;SEE IF TARGET OR ASSIST NODE   ADD     #6,SP
7512 117024 062702 000007      ADD     #7,R2            ; INFO CONTAINED IN 7TH BYTE OF ENTRY
7513 117030 111203              MOVVB   (R2),R3          ;MOVE INTO R3
7514 117032 020327 000001      CMP     R3,#CASIST       ;SEE IF ASSIST NODE
7515 117036 001404              BEQ     10$              ; IF YES, BR 10$
7516 117040 012737 061757 002314  MOV     #ARGTY7,KEYWD2    ; ELSE MOVE 'TARGET' INTO KEYWD2
7517 117046 000403              BR      20$              ; CONTINUE
7518 117050 012737 061747 002314 10$:  MOV     #ARGTY6,KEYWD2    ; MOVE 'ASSIST' INTO KEYWD2
7519 117056 013746 002314 20$:  PRINTF  #NODTYP,KEYWD2   ;PRINT NODE TYPE
117056 013746 002314              MOV     KEYWD2,(SP)
117062 012746 053167              MOV     #NODTYP,(SP)
117066 012746 000002              MOV     #2,-(SP)
117072 010600              MOV     SP,R0
117074 104417              TRAP   C$PNTF
117076 062706 000006              ADD     #6,SP
7520 117102              RETURN                    ;RETURN
7521 117104              EXIT  TST
117104 104432              RTS  PC
117106 000002              TRAP   C$EXIT
7522              .WORD  L10015
7524              ; *****
7525              ; INSERT LOCAL STORAGE THAT IS USED ONLY
7526              ; DURING THIS TEST.
7527              ; *****
7528
7529              ; *****
7530              ; INSERT MESSAGES THAT ARE USED ONLY
7531              ; DURING THIS TEST.
7532              ; *****
7533
7534              .EVEN
7535
7536              .EVEN
7537 117110              ENDTST
117110
117110 104401              L10015: TRAP  C$ETST
7538
7540              ; *****
7541              ; BEGIN THE REMAINING TESTS ON NEW PAGES.
7542              ; *****

```

READ LINE OF OPENED FILE

```

7545
7546          .SBTTL  HARDWARE PARAMETER CODING SECTION
7547
7548          ;**
7549          ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
7550          ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
7551          ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7552          ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
7553          ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7554          ; WITH THE OPERATOR.
7555          ;--
7556
7557          BGNHRD
7558
7559          117112      000015
7560
7561          ;*****
7562          ; INSERT HARDWARE PARAMETER INTERPRETIVE CODE HERE.  THIS CODE
7563          ; IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR
7564          ; DEVICE INFORMATION TO PUT IN THE P-TABLE.  THIS CODE IS USED
7565          ; IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
7566          ; USED IN THIS SECTION ARE "GPRMD", "GPRMA" AND "GPRML".
7567          ;*****
7568
7569          GPRMA  ASKCSR,0,0,160000,177776,YES          ; GET CSR ADDRESS
7570
7571          117114      000031          .WORD  T$CODE
7572          117114      117146          .WORD  ASKCSR
7573          117116      160000          .WORD  T$LLOLIM
7574          117120      177776          .WORD  T$HILIM
7575
7576          GPRMA  ASKVEC,2,0,0,776,YES          ; GET VECTOR ADDRESS
7577
7578          117124      001031          .WORD  T$CODE
7579          117124      117201          .WORD  ASKVEC
7580          117126      000000          .WORD  T$LLOLIM
7581          117130      000776          .WORD  T$HILIM
7582
7583          GPRMD  ASKPRI,4,0,340,0,7,YES          ; GET PRIORITY LEVEL
7584
7585          117132      002032          .WORD  T$CODE
7586          117132      117235          .WORD  ASKPRI
7587          117136      000340          .WORD  340
7588          117140      000000          .WORD  T$LLOLIM
7589          117142      000007          .WORD  T$HILIM
7590
7591          117144      000007
7592
7593          ENDHRD
7594
7595          ;*****
7596          ; INSERT MESSAGES THAT ARE USED ONLY
7597          ; DURING THE HARDWARE PARAMETER CODING SECTION.
7598          ;*****
7599
7600          ASKCSR: .ASCIZ  /WHAT IS THE PCSRO ADDRESS?/
7601
7602          117146      127      110      101
7603          117151      124      040      111
7604          117154      123      040      124
7605          117157      110      105      040
7606          117162      120      103      123
7607          117165      122      117      040
7608          117170      101      104      104

```

	117173	122	105	123	
	117176	123	077	000	
7583	117201	127	110	101	ASKVEC: .ASCIZ /WHAT IS THE VECTOR ADDRESS?/
	117204	124	040	111	
	117207	123	040	124	
	117212	110	105	040	
	117215	126	105	103	
	117220	124	117	122	
	117223	040	101	104	
	117226	104	122	105	
	117231	123	123	077	
	117234	000			
7584	117235	127	110	101	ASKPRI: .ASCIZ /WHAT IS THE PRIORITY LEVEL?/
	117240	124	040	111	
	117243	123	040	124	
	117246	110	105	040	
	117251	120	122	111	
	117254	117	122	111	
	117257	124	131	040	
	117262	114	105	126	
	117265	105	114	077	
	117270	000			
7585					.EVEN
7586					

```

7588          .SBTTL  SOFTWARE PARAMETER CODING SECTION
7589
7590          ;**
7591          ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7592          ; THAT ARE USED BY THE SUPERVISOR TO BUILD P TABLES.  THE
7593          ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7594          ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
7595          ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7596          ; WITH THE OPERATOR.
7597          ;
7598
7599 117272          BGNSFT
117272          000000
117274
                                           .WORD L10017-L$SOFT/2
                                           L$SOFT::
7600
7601          ;*****
7602          ; INSERT SOFTWARE PARAMETER INTERPRETIVE CODING HERE.  THIS CODE
7603          ; IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR
7604          ; SOFTWARE INFORMATION WHICH WILL BE PLACED IN THE SOFTWARE
7605          ; TABLE.  THIS SECTION IS OPTIONAL.
7606          ;*****
7607
7608          .EVEN
7609
7610          ENDSFT
7611
7612 117274          .EVEN
117274          L10017:
7613
7614          ;*****
7615          ; INSERT MESSAGES THAT ARE USED ONLY
7616          ; DURING THE SOFTWARE PARAMETER CODING SECTION.
7617          ;*****
7618
7619          $PATCH::
7620          .BLKW  5  ;; N.M.  (CHANGED FROM 100 TO 5)
7621
7622          ;*****
7623          ; THIS IS A PATCH AREA THAT SHOULD BE INCLUDED IN ALL DIAGNOSTICS.
7624          ; ADJUST THE SIZE TO FIT YOUR OWN PREFERENCES.
7625          ;*****
7626
7627          LASTAD
7628
7629          .EVEN
7630          .WORD  0
7631          .WORD  0
7632 117306
117306          000000
117310          000000
117312
L$LAST::
7633 117312          .BLKW  5  ;; N.M.  (CHANGED FROM 40 TO 5)

```

7635  
7636  
7638  
7639  
7640  
7641  
7642  
7643  
7644  
7645  
7646  
7647  
7649  
7650  
7651  
7652  
7653  
7654  
7655  
7656  
7657  
7658  
7659

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
;   ENDPTAB
;   ENDSETUP
;
; END OF THIS MODULE
;-
; .END

```

ACTALP	105376	B	= 000010	CCLNAD	= 000004 G	CMPER2	067677	C#CEFG	= 000045
ACTBLD	104440	BAD	116066	CCLNAL	= 000010 G	CMRSTR	075702	C#CLCK	= 000062
ACTCMS	112506	BCOUNT	050266 G	CCLSUM	= 000042	CNDADR	= 000030	C#CLEA	= 000012
ACTCNL	114052	BD	= 000010 G	CCNTR	= 000036	CNDLOG	= 000037	C#CLOS	= 000035
ACTCNT	112612	BINDEC	100342 G	CCPYS	= 000027	CNDODAL	= 000031	C#CLP1	= 000006
ACTCPY	107170	BINHEX	076044	CDEFLT	= 000044	CNODE	= 000015	C#CVEC	= 000036
ACTCSU	114304	BIT0	= 000001 G	CDIR	= 000043	CNTEND	050242	C#DCLN	= 000044
ACTCTT	106746	BIT00	= 000001 G	CEXADR	= 000013	CNTRS	050144	C#DODU	= 000051
ACTDFT	114360	BIT01	= 000002 G	CEXIT	= 000020 G	CNTR00	062047	C#DRPT	= 000024
ACTDIR	110236	BIT02	= 000004 G	CFLAG	003734	CNTR01	062127	C#DU	= 000053
ACTEXT	106576	BIT03	= 000010 G	CFUNCT	= 000040	CNTR02	062176	C#EDIT	= 000003
ACTHLP	104072	BIT04	= 000020 G	CHECK	116520	CNTR03	062232	C#ERDF	= 000055
ACTIDT	105552	BIT05	= 000040 G	CITSUM	050244	CNTR04	062277	C#ERHR	= 000056
ACTMSG	106470	BIT06	= 000100 G	CITST	050242	CNTR05	062354	C#ERRO	= 000060
ACTNAD	107246	BIT07	= 000200 G	CITVEC	= 000004 G	CNTR06	062424	C#ERSF	= 000054
ACTNAL	107404	BIT08	= 000400 G	CITWEN	= 024100 G	CNTR07	062463	C#ERSO	= 000057
ACTNOD	104130	BIT09	= 001000 G	CITWOR	= 014100 G	CNTR08	062533	C#ESCA	= 000010
ACTNUF	104062	BIT1	= 000002 G	CLIACT	103726	CNTR09	062606	C#ESEG	= 000005
ACTNUL	104070	BIT10	= 002000 G	CLIALN	= 000007	CNTR10	062657	C#ESUB	= 000003
ACTONE	106706	BIT11	= 004000 G	CLIALP	= 000006	CNTR11	062716	C#ETST	= 000001
ACTOPR	106756	BIT12	= 010000 G	CLIBIF	= 000003	CNTR12	062766	C#EXIT	= 000032
ACTPAT	112332	BIT13	= 020000 G	CLIBR	= 000002	CNTR13	063034	C#GETB	= 000026
ACTRNA	107612	BIT14	= 040000 G	CLIBRX	052222	CNTR14	063101	C#GETW	= 000027
ACTRNL	110670	BIT15	= 100000 G	CLIDEC	= 000011	CNTR15	063135	C#GMAN	= 000043
ACTRSP	114710	BIT2	= 000004 G	CLIERM	052113	CNTR16	063177	C#GPHR	= 000042
ACTRUN	107464	BIT3	= 000010 G	CLIERR	= 000000	CNTR17	063245	C#GPLD	= 000030
ACTSAV	114136	BIT4	= 000020 G	CLIXI	= 000001	CNTR18	063320	C#GPRI	= 000040
ACTSMS	112414	BIT5	= 000040 G	CLINBG	052175	CNTR19	063365	C#INIT	= 000011
ACTSND	113676	BIT6	= 000100 G	CLINUF	052144	COMPAR	061515	C#INLP	= 000020
ACTSR4	106670	BIT7	= 000200 G	CLINUM	= 000005	CONES	= 000017	C#MANI	= 000050
ACTSUM	105170	BIT8	= 000400 G	CLIOCT	= 000010	COPRSL	= 000024	C#MEM	= 000031
ACTSZE	107112	BIT9	= 001000 G	CLISPA	= 000004	COUNT	050304 G	C#MSG	= 000023
ACTTYP	107104	BLDAST	076724 G	CLISTR	= 000012	CPATRN	= 000005	C#OPEN	= 000034
ACTUNS	114202	BLDBUF	077652 G	CLITRE	050516	CPYCNT	050326 G	C#PNTB	= 000014
ACTUSF	114426	BLDFAS	076376 G	CLIPM	052104	CPYLMT	053511	C#PNTF	= 000017
ACTXAD	106606	BLDFLG	002200	CLKBR	003740	CR	= 000015 G	C#PNTS	= 000016
ACTZRO	106716	BLDLD	076132 G	CLKCSR	003736	CRNALL	= 000032	C#PNTX	= 000015
ACTOAL	106736	BLDMSG	052473	CLKEN	003746	CRUN	= 000004	C#QIO	= 000377
ACTIAL	106726	BLDREQ	077224 G	CLKHZ	003744	CSAVE	= 000006	C#RDBU	= 000007
ADDML	= 000001 G	BMPCNT	074632	CLKINT	070530 G	CSAVR4	= 000014	C#REFG	= 000047
ADR	= 000020 G	BMPPEX	075356	CLKSET	070504	CSHCTR	= 000002 G	C#RESE	= 000033
ADRBUF	002316	BMPML	075164	CLKVEC	003742	CSHMSG	= 000034	C#REVI	= 000003
ADRDEL	054044	BMPRCV	= 000000 G	CLRMSG	053432	CSIZE	= 000026	C#RFLA	= 000021
ALLNOD	061315	BMPREC	075054	CLUPPR	= 000033	CSR	= 000016 G	C#RPT	= 000025
ALPHA	= 000000 G	BMPXMT	= 000001 G	CMDBUF	002202	CSRBUF	050246	C#SEFG	= 000046
ANCHOR	070674	BOE	= 000400 G	CMDTY1	061610	CSRO	050004 G	C#SPRI	= 000041
ARGTY1	061702	BRDADR	004016	CMDTY2	061615	CSR1	050020 G	C#SVEC	= 000037
ARGTY2	061710	BUFLN	050332 G	CMDTY3	061625	CTARGT	= 000000 G	C#TPRI	= 000013
ARGTY3	061721	BUILD	= 000003	CMDTY4	061633	CTYPE	= 000025	C#COLL	= 000074 G
ARGTY4	061732	CADERR	053022	CMDTY5	061640	CUNSAV	= 000041	C#CRC	= 000001 G
ARGTY5	061743	CADRER	052746	CMDTY6	061644	CUNSVF	= 000045	C#FRAM	= 000002 G
ARGTY6	061747	CALPHA	= 000016	CMDTY7	061654	CZEROS	= 000020	C#LCAR	= 000002 G
ARGTY7	061757	CASIST	= 000001 G	CMDTY8	061661	C#AU	= 000052	C#MLN	= 000004 G
ASKCSR	117146	CBOADR	002370	CMDTY9	061667	C#AUTO	= 000061	C#MREC	= 000010 G
ASKPRI	117235	CCCITT	= 000023	CMPADR	116700	C#BRK	= 000022	C#MXMT	= 000040 G
ASKVEC	117201	CCITT	= 000005 G	CMPBUF	050334 G	C#BSEG	= 000004	C#OPEN	= 000010 G
ASSEMB	= 000010	CCLMSG	= 000035	CMRPER1	067602	C#BSUB	= 000002	C#PREC	= 000004 G



SYMBOL TABLE							
C.PXMD=	000054	G	EMSG37	064451	FAFCT4=	000050	G
C.PXMT=	000034	G	EMSG38	064515	FASIST	050454	G
C.PXM2=	000050	G	EMSG4	003532	FASKIP=	000016	G
C.PXM3=	000044	G	EMSG40	064561	FATFLG	050252	G
C.RDAT=	000020	G	EMSG41	064662	FILENM	115526	
C.RERB=	000014	G	EMSG42	064764	FILLIN	115542	
C.RERR=	000016	G	EMSG43	065062	FINDSL	116362	
C.RLEX=	000032	G	EMSG44	065141	FLAG =	000000	G
C.RLIN=	000030	G	EMSG45	065237	FLAGWO=	000000	G
C.RMDB=	000024	G	EMSG46	065303	FLAG1	050250	G
C.RTRY=	000001	G	EMSG47	065366	FORWRD=	000002	G
C.RUNT=	000010	G	EMSG48	065446	FRDADR=	000004	G
C.SEC5=	000002	G	EMSG5	003632	FREMEM	050024	G
C.SMRT=	000004	G	EMSG50	065544	FRESIZ	050022	G
C.SIZ =	000000	G	EMSG51	065600	FULAST	061353	
C.XABB=	000070	G	EMSG52	065633	FULSLT	116452	
C.XABT=	000072	G	EMSG53	065721	F\$AU =	000015	
C.XDAT=	000060	G	EMSG54	065773	F\$AUTO=	000020	
C.XMDB=	000064	G	EMSG55	066063	F\$BGN =	000040	
COALT =	000022		EMSG56	066116	F\$CLEA=	000007	
C1ALT =	000021		EMSG57	066152	F\$DU =	000016	
DATCMP	077734	G	EMSG58	066211	F\$END =	000041	
DECSTR	100526	G	EMSG59	066267	F\$HARD=	000004	
DEPADR	003774	G	EMSG60	066344	F\$HW =	000013	
DESC =	000002	G	EMSG61	066412	F\$INIT=	000006	
DESTIN=	000000	G	EMSG62	066451	F\$JMP =	000050	
DFPTBL	002170	G	EMSG63	066526	F\$MOD =	000000	
DIAGMC=	000000		EMSG64	066615	F\$MSG =	000011	
DIRC0M	110266		EMSG65	066705	F\$PROT=	000021	
DIRECT	061337		EMSG66	066777	F\$PWR =	000017	
DNIFLG	050262	G	EMSG67	067071	F\$RPT =	000012	
EDPACK	075474		EMSG68	067121	F\$SEG =	000003	
EF.CON=	000036	G	EMSG69	067204	F\$SOFT=	000005	
EF.NEW=	000035	G	ENTRND	116304	F\$SRV =	000010	
EF.PWR=	000034	G	EOFF	116070	F\$SUB =	000002	
EF.RES=	000037	G	EOFFND	116205	F\$SW =	000014	
EF.STA=	000040	G	ERRBLK	052102	F\$TEST=	000001	
EL =	001000	G	ERRFLG	050270	GETCL	103544	
EMSG0	003526	G	ERRMSG	052100	GETCOM	077442	
EMSG01	063436		ERRNBR	052076	GETRNX	077344	G
EMSG02	063467		ERROR	071002	GETXNX	077430	G
EMSG03	063527		ERRTYP	052074	G\$CNT0=	000200	
EMSG04	063563		ERR1	070306	G\$DELM=	000372	
EMSG08	063605		ERR2	070344	G\$DISP=	000003	
EMSG1	003527	G	ERR3	070432	G\$EXCP=	000400	
EMSG10	063670		EVL =	000004	G\$HILI=	000002	
EMSG11	063735		EXIT =	000011	G\$LOLI=	000001	
EMSG14	063774		EXIT1	116666	G\$NO =	000000	
EMSG2	003530	G	E\$END =	002100	G\$OFFS=	000400	
EMSG22	064050		E\$LOAD=	000035	G\$OF5I=	000376	
EMSG24	064103		FAADR1=	000022	G\$PRMA=	000001	
EMSG25	064157		FAADR2=	000032	G\$PRMD=	000002	
EMSG3	003531	G	FAADR3=	000042	G\$PRML=	000000	
EMSG32	064245		FAADR4=	000052	G\$RADA=	000140	
EMSG34	064322		FAFCT1=	000020	G\$RADB=	000000	
EMSG35	064367		FAFCT2=	000030	G\$RADD=	000040	
EMSG36	064424		FAFCT3=	000040	G\$RADL=	000120	
					G\$RADO=	000020	
					G\$XFER=	000004	
					G\$YES =	000010	
					H0MSG1	055046	
					HEADER=	000016	G
					HELP =	000001	
					HELP1	055120	
					HELP10	056141	
					HELP11	056232	
					HELP12	056435	
					HELP13	056542	
					HELP14	056641	
					HELP15	056740	
					HELP16	057043	
					HELP17	057132	
					HELP18	057235	
					HELP19	057305	
					HELP2	055221	
					HELP20	057410	
					HELP21	057466	
					HELP22	057551	
					HELP23	057652	
					HELP24	057752	
					HELP25	060102	
					HELP26	060166	
					HELP27	060272	
					HELP28	060374	
					HELP29	060513	
					HELP3	055314	
					HELP30	060563	
					HELP4	055365	
					HELP5	055436	
					HELP6	055536	
					HELP7	055651	
					HELP8	055762	
					HELP9	36052	
					HEXBIN	075724	
					HEXC	076022	
					HIADD =	000002	G
					HIRCV =	000006	G
					HIXMT =	000012	G
					HLPEND	003322	
					HLPTAB	003224	
					HLP115	056330	
					HN	075700	
					HOE =	100000	G
					HXFORM	075566	
					HXN	075564	
					IBE =	010000	G
					IDENT =	000010	G
					IDFUNC=	000005	G
					IDTDAT	047674	
					IDTSIZ	047770	
					IDU =	000040	G
					IER =	020000	G
					ILADMS	052565	
					ILADM1	052651	
					ILLADR	002656	
					INIADR=	000000	G
					INICLN	103344	
					INIEXI	103354	
					INIT	102126	
					INIT1	102154	
					ISR =	000100	G
					IXE =	004000	G
					I\$AU =	000041	
					I\$AUTO=	000041	
					I\$CLN =	000041	
					I\$DU =	000041	
					I\$HRD =	000041	
					I\$INIT=	000041	
					I\$MOD =	000041	
					I\$MSG =	000041	
					I\$PROT=	000040	
					I\$PTAB=	000041	
					I\$PWR =	000041	
					I\$RPT =	000041	
					I\$SEG =	000041	
					I\$SETU=	000041	
					I\$SFT =	000041	
					I\$SRV =	000041	
					I\$SUB =	000041	
					I\$TST =	000041	
					J\$JMP =	000167	
					KEYWD1	002312	
					KEYWD2	002314	
					KILML =	000002	G
					LAPCNT	050336	
					LCLKEN=	000100	G
					LDADR1=	000022	G
					LDADR2=	000032	G
					LDFCT1=	000020	G
					LDFCT2=	000030	G
					LDRESP	052310	
					LDSKIP=	000016	G
					LENGTH	061506	
					LF =	000012	G
					LGERMS	067745	
					LOAD0 =	000004	G
					LOCDST	075360	
					LOE =	040000	G
					LOGDEL	054132	
					LOGNAM	053154	
					LOPDIR	050346	G
					LORCV =	000004	G
					LORTGT	004040	
					LOT =	000010	G
					LOXMT =	000010	G
					LOXTGT	004060	
					LPADR	004024	G
					LSRTRY	002402	
					LST	076042	
					LUPAIR	061326	
					L\$ACP	002110	G

L\$APT	002036	G	L10004	070502	MSGTY4	061564	NOD111	051414	NOD22	050700
L\$AU	103536	G	L10005	070654	MSGTY5	061571	NOD112	051420	NOD23	050702
L\$AUT	002070	G	L10006	102116	MSGTY6	061577	NOD113	051424	NOD24	050704
L\$AUTO	103370	G	L10010	103366	MSG0C	003342	NOD114	051440	NOD25	050720
L\$CCP	002106	G	L10011	103370	MSG00	003376	NOD115	051444	NOD26	050724
L\$CLEA	103372	G	L10012	103526	MSG01	003526	NOD116	051460	NOD27	050744
L\$CO	002032	G	L10013	103534	MSG02	003527	NOD117	051464	NOD3	050530
L\$DEPO	002011	G	L10014	103542	MSG03	003530	NOD12	050616	NOD30	050750
L\$DESC	002130	G	L10015	117110	MSG04	003531	NOD120	051500	NOD31	050766
L\$DESP	002076	G	L10016	117146	MSG05	003532	NOD121	051504	NOD32	050772
L\$DEVP	002060	G	L10017	117274	MSG1	054415	NOD122	051520	NOD33	051010
L\$DISP	002164	G	L5060	052242	MSG1C	003344	NOD123	051524	NOD34	051014
L\$DLY	002116	G	MABORT	004000	MSG11	054530	NOD124	051540	NOD35	051030
L\$DTP	002040	G	MATCH	115746	MSG12	054643	NOD125	051544	NOD36	051032
L\$DTYP	002034	G	MBOOT	000010	MSG2	054703	NOD126	051560	NOD37	051052
L\$DJ	103530	G	MCARRI	003000	MSG2C	003346	NOD127	051564	NOD4	050544
L\$DUT	002072	G	MCHADR	040000	MSG3	054741	NOD13	050622	NOD40	051056
L\$DVTY	002122	G	MCRC	000002	MSG3C	003350	NOD130	051600	NOD41	051074
L\$EF	002052	G	MCSTAD	004032	MSG4	055002	NOD131	051604	NOD42	051076
L\$ENVI	002044	G	MDISC	010000	MSG4C	003352	NOD132	051610	NOD43	051100
L\$ERRT	052074	G	MELoop	001000	MSG5C	003354	NOD133	051614	NOD44	051102
L\$ETP	002102	G	MEMOM	020000	MSG6C	003356	NOD134	051620	NOD45	051106
L\$EXP1	002046	G	MESPAT	061220	MSHORT	000010	NOD135	051634	NOD46	051112
L\$EXP4	002064	G	MESPA1	061271	MVALID	100000	NOD136	051640	NOD47	051116
L\$EXP5	002066	G	MFAIL	000400	MXERRS	040000	NOD137	051644	NOD5	050546
L\$HARD	117114	G	MFERRS	040000	MXLAST	100000	NOD14	050626	NOD50	051122
L\$HIME	002120	G	MFLAST	100000	MXMTIN	000200	NOD140	051650	NOD51	051124
L\$HPCP	002016	G	MFRAM	000004	MXSETP	010000	NOD141	051666	NOD52	051130
L\$HPTP	002022	G	MFUSED	040000	MXUSED	040000	NOD142	051672	NOD53	051144
L\$HW	002170	G	MHIBYT	000100	NCRTRY	002404	NOD143	051676	NOD54	051150
L\$ICP	002104	G	MILoop	000400	NETADD	000000	NOD144	051702	NOD55	051166
L\$INIT	102126	G	MINTEN	000100	NETADO	050000	NOD145	051706	NOD56	051172
L\$LADP	002026	G	MINVRC	000040	NETAD1	050014	NOD146	051712	NOD57	051212
L\$LAST	117312	G	MINVXM	000020	NEW	103334	NOD147	051732	NOD6	050562
L\$LOAD	002100	G	MLOBYT	000200	NIRCNT	050256	NOD15	050640	NOD60	051216
L\$LUN	002074	G	MLOSS	010000	NOADR	004002	NOD150	051736	NOD61	051222
L\$MREV	002050	G	MNOCAR	002000	NOCLK	061766	NOD151	051750	NOD62	051224
L\$NAME	002000	G	MNXM	000004	NOCMPR	053664	NOD152	051754	NOD63	051230
L\$PRIO	002042	G	MOVF	000001	NOD	053415	NOD153	051772	NOD64	051244
L\$PROT	102120	G	MRCVEN	000001	NODADR	053144	NOD154	051776	NOD65	051250
L\$PRT	002112	G	MRCVIN	100000	NODE	000002	NOD155	052014	NOD66	051254
L\$REPP	002062	G	MRERRS	040000	NODTBL	002412	NOD156	052020	NOD67	051260
L\$REV	002010	G	MRESET	000002	NODTY	002406	NOD157	052034	NOD7	050566
L\$RPT	102106	G	MRLAST	100000	NODTYP	053167	NOD16	050644	NOD70	051272
L\$SOFT	117274	G	MRLONG	020000	NOD0	050516	NOD160	052040	NOD71	051276
L\$SPC	002056	G	MRSETP	020000	NOD1	050522	NOD161	052044	NOD72	051302
L\$SPCP	002020	G	MRUNT	004000	NOD10	050602	NOD162	052046	NOD73	051306
L\$SPTP	002024	G	MRUSED	040000	NOD100	051344	NOD163	052052	NOD74	051312
L\$STA	002030	G	MSANIT	002000	NOD101	051362	NOD164	052056	NOD75	051316
L\$SW	002200	G	MSGAD	003360	NOD102	051366	NOD165	052060	NOD76	051322
L\$TEST	002114	G	MSGCNT	003342	NOD103	051370	NOD166	052064	NOD77	051340
L\$TIML	002014	G	MSGPRM	054345	NOD104	051372	NOD167	052070	NORESP	061446
L\$UNIT	002012	G	MSGTAB	003324	NOD105	051376	NOD17	050662	NOTNUF	000012
L10000	002176		MSGTY0	061536	NOD106	051402	NOD170	052072	NTBHDR	053176
L10001	002200		MSGTY1	061544	NOD107	051406	NOD2	050526	NULL	000000
L10002	070342		MSGTY2	061551	NOD11	050604	NOD20	050664	NULSTR	053076
L10003	070430		MSGTY3	061557	NOD110	051412	NOD21	050676	N10\$	050522

SYMBOL TABLE

N100#	051124	N30#	051014	P#EXIT	100670	RRGSRT	024216 G	SLFIST	050312
N101#	051130	N31#	051032	P#GDBD	003221	RRG001	025674 G	SLOT	002410 G
N102#	051150	N32#	051056	P#MERR	003223	RRG002	026774 G	SMLSIZ	000100 G
N104#	051172	N33#	051076	P#NUMF	003220	RRG003	030074 G	SOURCC	000006 G
N106#	051216	N80#	051102	P#NUM	003214	RRG004	031174 G	SOURCE	075472
N110#	051222	N81#	051106	P#PASS	002400	RRG005	032274 G	SRBL	000010 G
N12#	050530	N82#	051112	P#RADX	003216	RRG006	033374 G	START	102234
N120#	051224	N90#	051116	P#SIZE	002374	RRG007	034474 G	STATBF	002664
N121#	051230	N95#	051122	P#TREE	003206	RRG010	035574 G	STATBL	002714
N122#	051250	OK	061025	P#TRV	100544	RRG011	036004 G	STAT1	000010 G
N123#	051276	OKFU	061157	P#TRS	100554	RRG012	037774 G	STAT2	000012 G
N124#	051312	OKRE	061046	P#TYPE	002372	RRG013	041074 G	STAT3	000010 G
N126#	051316	OKTR	061112	QNAADO	047772 G	RRG014	042174 G	STAT4	000012 G
N130#	051322	ONEALT	000003 G	QNAAD1	050006 G	RRG015	043274 G	STBLN	000132 G
N132#	051344	ONES	000001 G	QNAEXI	072122	RRG016	044374 G	STDR	000016 G
N134#	051366	OPNERR	116072	QNAINI	071114 G	RRG017	045474 G	STRBUF	002324
N135#	051370	OPRSEL	000006 G	QNAIN1	071534	RRG020	046574 G	STRBU1	002346
N14#	050546	OPSLBF	003632	QNAISR	073064	RRING	024260 G	SUMM	053422
N140#	051372	O#APTS	000000	QNAPRO	047776 G	RRINGH	024544 G	SUMMARY	000007
N141#	051376	O#AU	000000	QNAPR1	050012 G	RRGTG	024172 G	SUMMS1	070043
N142#	051402	O#BGNR	000001	QNAVCO	047774 G	RSPFLG	003202	SUMMS2	070064
N143#	051412	O#BGNS	000000	QNAVCI	050010 G	RSPOND	000046	SUMMS3	070150
N16#	050566	O#DU	000000	QUIT	114272	RTRYER	052411	SUMMS4	070177
N160#	051414	O#ERRT	000000	RASIST	050422 G	RUNALL	107622	SUMMS5	070262
N161#	051420	O#GNSW	000000	RBFCNT	050264 G	RUNCOM	112000	SUMMS6	070300
N162#	051464	O#POIN	000001	RBUFB	004100	RUNDIR	110246	SVCGBL	000000
N163#	051504	O#SETU	000000	RBUFNR	000151 G	RUNLUP	110700	SVCINS	000001
N164#	051524	PART	100542 G	RCVBUF	050300 G	RUNPAT	112342	SVCSUB	000001
N165#	051544	PASABT	060672	RCVERR	050276 G	SAVCOM	114260	SVCTAG	000001
N166#	051564	PATTRN	061426	RCVTGT	023772	SAVED	054324	SVCTST	000001
N167#	051604	PCCALL	050330 G	RDERR	116126	SAVTBL	002536	S#LSYM	010000
N168#	051614	PCEFLG	050254 G	RDESAL	002354 G	SCOLL	000004 G	S.BYTE	050040 G
N170#	051620	PCLKCT	001600 G	RDESC	021200	SELMG	116246	S.COMP	050036 G
N175#	051650	PCLKEN	000111 G	RDESSZ	000014 G	SETADD	072346	S.LEN	050034 G
N178#	051702	PCMSG	067542 G	RDLIN	115414	SETEXI	072450	S.NREC	050032 G
N18#	050604	PHYADR	004010 G	RECAST	061407	SETFIL	073006	S.REC	050030 G
N180#	051706	PLINE	116154	RECERR	052345	SETINI	072212	S.XFER	050042 G
N181#	051712	PNT	001000 G	RECEVE	073740	SETKLD	072410	TABCLR	054220
N182#	051736	PREG14	070656	REPLY	000001 G	SETUP	072126	TABEMT	053346
N183#	051754	PRI	002000 G	REQID	050340 G	SETVAL	072720	TABFUL	053274
N184#	051776	PRI00	000000 G	RESTOR	054333	SETWRT	072454	TASIST	050370 G
N185#	052014	PRI01	000040 G	RESTRT	103234	SFPTBL	002200 G	TBLLEN	000050 G
N186#	052020	PRI02	000100 G	RETRY	061462	SIADDR	000042 G	TEMP	050314 G
N190#	052044	PRI03	000140 G	RETRYS	050274 G	SIDEV	000053 G	TEMP1	050316 G
N20#	050622	PRI04	000200 G	RIFUNC	000020 G	SI'CO	000030 G	TEMP2	050320 G
N200#	052046	PRI05	000240 G	RLSTHI	003772 G	SIFNCT	000035 G	TEMP3	050322 G
N201#	052052	PRI06	000300 G	RLSTLO	003770 G	SIMSG1	067266	TENPWR	100456
N210#	052060	PRI07	000340 G	RNGSIZ	024256	SIMSG2	067325	TGTADR	023570
N212#	052064	PROTOT	000014 G	RNXPRV	077360 G	SIMSG3	067360	TGTSIZ	023770
N215#	052072	PROT00	050306 G	ROMAD1	003764 G	SIMSG4	067420	TIMERS	003762
N22#	050644	PROT02	050310 G	ROMSIZ	050302 G	SIMSG5	067441	TIMER1	003756
N23#	050664	PRTNOD	116734	RPKLEN	001100 G	SIMSG6	067467	TIMER2	003760
N24#	050700	P#ACT	003210	RRGCHN	024560 G	SIMSG7	067515	TIMMIN	003750
N25#	050704	P#AERR	003222	RRGCR	024222 G	SIRCPT	000022 G	TIMOUT	050272 G
N26#	050724	P#BUFA	003204	RRGLST	024232 G	SIUECO	000031 G	TIMSEC	003752
N28#	050750	P#CNT	003212	RRGNXT	024226 G	SIVERS	000027 G	TIMTCK	003754
N29#	050772	P#CPYS	002376	RRGPRV	024234 G	SIZLMT	053575	TRAST	061367

SYMBOL TABLE

TRVACT 100672	T\$LOLI= 000000	T\$SW = 010001	VINVRC= 000005 G	WRDCNT= 000006 G
TRVADR 101656	T\$LSYM= 010000	T\$TES= 010015	VINVXM= 000004 G	WRITES 100072
TRVALN 101472	T\$LTNO= 000001	T1 103544 G	VLOBYT= 000007 G	WRITEXI 072736
TRVALP 101426	T\$NEST= 177777	UAM = 000200 G	VLOSS = 000014 G	XC = 010000 G
TRVBIF 100776	T\$NSO = 000005	UCB12 050044	VNOCAR= 000013 G	XFER 050324 G
TRVBR 100766	T\$PTNU= 000000	UNBOND 053756	VNXM = 000002 G	XFLAG 050260 G
TRVBRC 100712	T\$SAVL= 177777	UNIT 050026 G	VQVF = 000000 G	XMIT 073222 G
TRVDEC 101072	T\$SEGL= 177777	UNMSG 054265	VRBL = 000000 G	XMTDAT= 000001 G
TRVERR 100730	T\$SUBN= 000000	V = 100000 G	VRCVEN= 000000 G	XMTSET= 000000 G
TRVEXI 100750	T\$TAGL= 177777	VABORT= 000013 G	VRCVIN= 000017 G	XPKLEN= 001100 G
TRVNMA 101112	T\$TAGN= 010020	VBOOT = 000003 G	VRERRS= 000016 G	XRGCUR 024220 G
TRVNOB 100722	T\$TEMP= 000005	VCARRI= 000015 G	VRESET= 000001 G	XRGINV 024252
TRVNUM 101104	T\$TEST= 000001	VCHADR= 000016 G	VRLAST= 000017 G	XRGLST 024230 G
TRVOCT 101104	T\$TSTM= 177777	VCOLLO= 000004 G	VRLONG= 000015 G	XRGNXT 024224 G
TRVSPA 101020	T\$TSTS= 000001	VCRC = 000001 G	VRSETP= 000015 G	XRGSRT 024214 G
TRVSTR 101560	T\$TAU = 010014	VDISC = 000014 G	VRUNT = 000013 G	XRG001 024574 G
T\$TMS1 060715	T\$TAUT= 010011	VECTOR= 000014 G	VRUSED= 000016 G	XRING 024236 G
T\$TMS2 060735	T\$TCLE= 010012	VELOOP= 000011 G	VSANIT= 000012 G	XRINGH 024252
T\$TMS3 060755	T\$TDU = 010013	VEOM = 000015 G	VSHORT= 000003 G	XRNGB 024212 G
T\$TMS4 060770	T\$THAR= 010016	VFAIL = 000010 G	VTDR = 000000 G	X\$ = 000171
T\$ARGC= 000002	T\$THM = 010000	VFERRS= 000016 G	VVALID= 000017 G	X\$ALWA= 000000
T\$CODE= 002032	T\$THINI= 010010	VFLAST= 000017 G	VXERRS= 000016 G	X\$FALS= 000040
T\$ERRN= 000105	T\$THMSG= 010004	VFRAM = 000002 G	VXLAST= 000015 G	X\$OFFS= 000400
T\$EXCP= 000000	T\$THPRO= 010007	VFUSED= 000016 G	VXMTIN= 000007 G	X\$TRUE= 000020
T\$FLAG= 000040	T\$THRPT= 010006	VHIBYT= 000006 G	VXSETP= 000014 G	ZEROS = 000002 G
T\$GMAN= 000000	T\$THSOF= 010017	VILoop= 000010 G	VXUSED= 000016 G	ZROALT= 000004 G
T\$HILI= 000007	T\$THSRV= 010005	VINTEN= 000006 G	WAIT 070720 G	\$PATCH 11274 G
T\$LAST= 000001				

. ABS. 117324 000  
 000000 001  
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 32432 WORDS ( 127 PAGES)  
 DYNAMIC MEMORY: 20060 WORDS ( 77 PAGES)  
 ELAPSED TIME: 00:09:03  
 CVNIAA.BIN,CVNIAA.LST/-SP=SVC34R.MLB,CVNIAA.P11