

The main body of the document consists of a grid of approximately 20 columns and 30 rows of small, illegible data tables. Each cell in the grid contains a miniature version of a data table, likely representing individual test results or component specifications. Due to the low resolution and dark background, the specific text within these tables is not legible.

JUL 1984  
digital  
MADE IN USA

1

.REM 8

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  
DEC

PDP  
DECUS

UNIBUS  
DECTAPE

MASSBUS

## TABLE OF CONTENTS

1.0	GENERAL INFORMATION
1.1	PROGRAM ABSTRACT
1.2	SYSTEM REQUIREMENTS
1.3	RELATED DOCUMENTS AND STANDARDS
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES
1.5	ASSUMPTIONS
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 FLAGS (SEE SECTION 2.3)

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

## 2.2 SWITCHES

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

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:DDDD	EXECUTE DDDD PASSES (DDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP:DDDD	REPORT END OF PASS MESSAGE AFTER EVERY DDDD PASSES ONLY. (DDDD = 1 TO 64000)
/UNITS:LIST	TEST/ADD/DROP ONLY THOSE UNITS SPECIFIED IN THE LIST. LST 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) ? SH C  
--

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

\*\*\*\*\*

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-F0/T  
-----

EXAMPLE OF ASSIST NODE ENTRY:

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

EXAMPLE:

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

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

TYPE ONE OF THE FOLLOWING MESSAGE TYPES:

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

SIZE THE SIZE OF THE MESSAGE BUFFER 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-FO  
-----

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-FO  
-----

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 HRD 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 HRD 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					

```

927          .SBTTL PROGRAM HEADER
947
948 000000          .ENABL ABS,AMA
949          002000          "      "      2000
950
951          .SBTTL PROGRAM MACROS
952
953          ;I$STACK MACRO
954          ;-----
955
956          ;***
957          ;THE I$STACK MACRO FACILITATES INITIALIZING THE R6 (HARDWARE) STACK
958          ;AND THE R5 (PARAMETER) STACK.  R5 IS SET TO THE STACK LOW LIMIT
959          ;(STAKLO) AND THE PARAMETER STACK GROWS UPWARD.  R6 IS SET TO THE
960          ;STACK HIGH LIMIT (STAKHI) AND THE HARDWARE STACK GROWS DOWNWARD.
961          ;IF THERE IS A STACK OVER-RUN, IT WILL BE DETECTED BY THE PREG14
962          ;ROUTINE.
963          ;---
964          .MACRO I$STACK STAKLO,STAKHI
965          .LIST
966          MOV STAKLO,R5          ;INITIALIZE THE PARAMETER STACK POINTER.
967          MOV STAKHI,SP        ;INITIALIZE THE HARDWARE STACK POINTER.
968          .NLIST
969          .ENDM I$STACK
970
971          ;PUSH MACRO
972          ;-----
973
974          ;***
975          ;THE "PUSH" MACRO FACILITATES PUSHING ITEMS ON THE HARDWARE STACK.
976          ;UP TO SEVEN ITEMS MAY BE PLACED ON THE STACK WITH ONE MACRO.
977          ;---
978          .MACRO PUSH A,B,C,D,E,F,G
979
980          .IF NB G
981          .LIST
982          MOV G,-(SP)
983          .NLIST
984          .ENDC
985
986          .IF NB F
987          .LIST
988          MOV F,-(SP)
989          .NLIST
990          .ENDC
991
992          .IF NB E
993          .LIST
994          MOV E,-(SP)
995          .NLIST
996          .ENDC
997
998          .IF NB D
999          .LIST
1000
1001
1002

```

```

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

```

```

MOV     D, -(SP)
MOV     C, -(SP)
MOV     B, -(SP)
MOV     A, -(SP)
MOV     (SP), .A
MOV     (SP), .B
MOV     (SP), .C
MOV     (SP), .D

```

```

1060
1061 .IF NB E
1062 .LIST
1063 MOV (SP)+,E
1064 .NLIST
1065 .ENDC
1066
1067 .IF NB F
1068 .LIST
1069 MOV (SP)+,F
1070 .NLIST
1071 .ENDC
1072
1073 .IF NB G
1074 .LIST
1075 MOV (SP)+,G
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 MOV G,(R5)+
1096 .NLIST
1097 .ENDC
1098
1099 .IF NB F
1100 .LIST
1101 MOV F,(R5)+
1102 .NLIST
1103 .ENDC
1104
1105 .IF NB E
1106 .LIST
1107 MOV E,(R5)+
1108 .NLIST
1109 .ENDC
1110
1111 .IF NB D
1112 .LIST
1113 MOV D,(R5)+
1114 .NLIST
1115 .ENDC
1116

```

```

1117 .IF NB C
1118 .LIST
1119 MOV C,(R5)+
1120 .NLIST
1121 .ENDC
1122 .JF NB B
1123 .LIST
1124 MOV B,(R5)+
1125 .NLIST
1126 .ENDC
1127 .IF NB A
1128 .LIST
1129 MOV A,(R5)+
1130 .NLIST
1131 .ENDC
1132 .LIST
1133 JSR R4,PREG14
1134 .WORD S-ANCHOR
1135 .NLIST
1136 .ENDM CALL
1137 ;RETURN MACRO
1138 ;-----
1139
1140 ;***
1141 ;THE RETURN MACRO FACILITATES PASSING PARAMETERS BACK TO A CALLING
1142 ;ROUTINE. UP TO 7 PARAMETERS MAY BE PASSED BACK ON THE PARAMETER
1143 ;STACK.
1144 ;---
1145 .MACRO RETURN A,B,C,D,E,F,G
1146
1147 .IF NB G
1148 .LIST
1149 MOV G,(R5)+
1150 .NLIST
1151 .ENDC
1152 .IF NB F
1153 .LIST
1154 MOV F,(R5)+
1155 .NLIST
1156 .ENDC
1157 .IF NB E
1158 .LIST
1159 MOV E,(R5)+
1160 .NLIST
1161 .ENDC
1162 .IF NB D
1163 .LIST
1164 MOV D,(R5)+
1165 .NLIST
1166 .ENDC
1167 .IF NB C
1168 .LIST
1169 MOV C,(R5)+
1170 .NLIST
1171 .ENDC
1172 .JF NB B
1173 .LIST

```

```

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      ;***
1257      ;THE P$POP MACRO FACILITATES RETRIEVING PARAMETERS FROM THE PARAMETER
1258      ;STACK.  UP TO 7 PARAMETERS MAY BE RETRIEVED.
1259      ;
1260      ;THE ROUTINE THAT RECEIVES THE PARAMETERS HAS THE RESPONSIBILITY OF
1261      ;CLEANING UP THE PARAMETER STACK.  THIS MACRO IS AN AID TO MAKING
1262      ;A LOCAL COPY OF PASSED PARAMETERS AND CLEANING UP THE PARAMETER STACK.
1263      ;---
1264
1265      .MACRO P$POP  A,B,C,D,E,F,G
1266
1267      .IF NB A
1268      .LIST
1269      MOV      -(R5),A
1270      .NLIST
1271      .ENDC
1272
1273      .IF NB B
1274      .LIST
1275      MOV      -(R5),B
1276      .NLIST
1277      .ENDC
1278
1279      .IF NB C
1280      .LIST
1281      MOV      -(R5),C
1282      .NLIST
1283      .ENDC
1284
1285      .IF NB D
1286      .LIST
1287      MOV      -(R5),D

```

1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348

```

.NLIST
.ENDC

.IF NB E
.LIST
MOV -(R5),E

.NLIST
.ENDC

.IF NB F
.LIST
MOV -(R5),F

.NLIST
.ENDC

.IF NB G
.LIST
MOV -(R5),G

.NLIST
.ENDC

.ENDM P$POP

.MACRO CLI CHAR,HITVAL,MISADR,CMPSTR
.NODCL CHAR,HITVAL,\X$,MISADR,CMPSTR ;:#### PARSE TREE ####
.ENDM

.MACRO NODCL CHAR,HITVAL,XY,MISADR,CMPSTR
.NOD'XY: .BYTE CHAR,HITVAL ;SPECIAL CHAR. CODE OR COMPARE CHAR.
; AND ACTION (HIT) VALUE FOR ACTION
; ROUTINES.

.IF NB MISADR
.WORD MISADR-NOD'XY ;DISPLACEMENT TO "MISS" NODE (BYTES)
.ENDC
.IF NB CMPSTR
.WORD 1$-NOD'XY ;DISPLACEMENT TO GET TO NEXT NODE
.ASCIZ CMPSTR ; (ONLY IF ITS A "CLISTR" NODE)
.EVEN

.NLIST
1$:
.LIST
.ENDC
.NLIST
X$=X$+1
.LIST
.ENDM

.MACRO NEXT A,B
.LIST
.WORD A'B ; SEGMENT BUFFER ADDRESS
.NLIST

.ENDM
.MACRO RNGFRM A,B,C ; MACRO TO FORM TRANSMIT AND RECEIVE
; DESCRIPTOR RINGS.

```



1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1384

002000  
002000  
002000 103  
002001 126  
002002 116  
002003 111  
002004 101  
002005 000  
002006 000  
002007 000  
002010  
002010 101  
002011  
002011 060  
002012 000000  
002014  
002014 000000  
002016  
002016 117114  
002020  
002020 000000  
002022  
002022 002170  
002024  
002024 000000  
002026  
002026 117312  
002030  
002030 000000  
002032  
002032 000000  
002034  
002034 000001  
002036  
002036 000000  
002040  
002040 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$REV:: .ASCII /A/
L$DEPO:: .ASCII /O/
L$UNIT:: .WORD 0
L$TIML:: .WORD 0
L$HPCP:: .WORD L$HARD
L$SPCP:: .WORD 0
L$HPTP:: .WORD L$HW
L$SPTP:: .WORD 0
L$LADP:: .WORD L$LAST
L$STA:: .WORD 0
L$CO:: .WORD 0
L$DTYP:: .WORD 1
L$APT:: .WORD 0
L$DTP:: .WORD L$DISPATCH
```

```

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
:
:   DEVTYP <DEQNA>

```

```

002122
002122 104 105 121
002125 116 101 000

```

```

L$DVTYP:: .ASCIZ /DEQNA/

```

.EVEN

1401				
1407				
1408				
1409				
1410	002130			
	002130			
	002130	103	126	116
CISER/				
	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		

L\$DESC:: .ASCIZ /CVNID DEQNA NI EXER

.EVEN

.EVEN

1411  
1412  
1419  
1420  
1421  
1422  
1423  
1434  
1435

:  
: TEST DESCRIPTION  
:  
: DESCRIPT <CVNID DEQNA NI EXERCISER>  
:  
:  
: FORMAT STATEMENTS USED IN PRINT CALLS  
:  
:

1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452

002162  
002162 000001  
002164  
002164 103544

.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  
1469  
  
1470  
1471  
1472  
1473  
1474  
1484  
1485

002166  
002166 000003  
002170  
002170  
  
002170 174440  
002172 000300  
002174 000240  
  
002176  
002176

.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.  
:--

BGNHW DFPTBL

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

.WORD 174440 : CSR  
.WORD 300 : VECTOR  
.WORD PRI05 : PRIORITY

ENDHW

L10000:

1487  
 1488  
 1489  
 1490  
 1491  
 1492  
 1493  
 1494  
 1495  
 1496 002176  
 002176 000000  
 002200  
 002200  
 1497  
 1505  
 1506 002200  
 002200  
 1507  
 1508  
 1509  
 1519  
 1520  
 1521  
 1522  
 1523  
 1524  
 1525  
 1540  
 1541 002200

.SBTTL SOFTWARE P-TABLE

```

; **
; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
; PROGRAM AS OPERATIONAL PARAMETERS. THESE PARAMETERS ARE
; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
; AT RUN TIME.
; --

```

BGNSW SFPTBL

.WORD L10001-L\$SW/2

L\$SW::  
SFPTBL::

ENDSW

L10001:

.SBTTL GLOBAL EQUATES SECTION

```

; **
; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
; ARE USED IN MORE THAN ONE TEST.
; --

```

EQUALS

; BIT DIFINITIONS

```

;
; BIT15== 100000
; BIT14== 40000
; BIT13== 20000
; BIT12== 10000
; BIT11== 4000
; BIT10== 2000
; BIT09== 1000
; BIT08== 400
; BIT07== 200
; BIT06== 100
; BIT05== 40
; BIT04== 20
; BIT03== 10
; BIT02== 4
; BIT01== 2
; BIT00== 1
;
; BIT9== BIT09
; BIT8== BIT08
; BIT7== BIT07
; BIT6== BIT06
; BIT5== BIT05
; BIT4== BIT04
; BIT3== BIT03
; BIT2== BIT02

```

```

000002      BIT1== BIT01
000001      BIT0== BIT00
;
; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
;
000040      EF.START==      32.      ; START COMMAND WAS ISSUED
000037      EF.RESTART==    31.      ; RESTART COMMAND WAS ISSUED
000036      EF.CONTINUE==   30.      ; CONTINUE COMMAND WAS ISSUED
000035      EF.NEW==        29.      ; A NEW PASS HAS BEEN STARTED
000034      EF.PWR==        28.      ; A POWER-FAIL/POWER-UP OCCURRED
;
; PRIORITY LEVEL DEFINITIONS
;
000340      PRI07== 340
000300      PRI06== 300
000240      PRI05== 240
000200      PRI04== 200
000140      PRI03== 140
000100      PRI02== 100
000040      PRI01== 40
000000      PRI00== 0
;
; OPERATOR FLAG BITS
;
000004      EVL==          4
000010      LOT==         10
000020      ADR==         20
000040      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      HOE==     100000

```

```

1543
1544
1545      ; CSR DEFINITIONS
1546      000010  BD      ==      BIT03      ;BOOT/DIAGNOSTIC ROM
1547      001000  EL      ==      BIT09      ;ELOOP
1548      010000  XC      ==      BIT12      ;TRANSCIEVER
1549
1550      ; DESCRIPTOR FIELDS
1551
1552      000000  FLAGWORD==      0      ;FLAG WORD
1553      000010  STAT3   ==      8.      ;STATUS WORD 1
1554      000012  STAT4   ==     10.      ;STATUS WORD 2
1555
1556      ; DESCRIPTOR BITS
1557
1558      100000  V       ==      BIT15      ;VALID BIT
1559
1560      000004  CITVECTOR==      4      ;OFFSET IN 4K ROM FOR VECTOR TO CITIZENSHIP TEST
1561
1562      ; TERMINAL CONTROL CHARACTERS
1563
1564      000015  CR       ==     15      ;
1565      000012  LF       ==     12      ;
1566
1567      :::EQUATES FOR FLAG WORD:::
1568
1569      000000  CTARGET==0
1570      000001  CASIST==1
1571      000002  CSHCTR==2      ;ARG TYPE FOR 'SHOW COUNTERS' CMD
1572      000004  CCLNAD==4      ;ARG TYPE FOR 'CLEAR NODE/ADR' CMD
1573      000010  CCLNAL==8.      ;ARG TYPE FOR 'CLEAR NODE/ALL' CMD
1574      000020  CEXIT==16.
1575
1576      :::CLOCK ENABLE VALUES TO BE LOADED IN CLK'S CSR:::
1577
1578      000100  LCLKEN==100      ; L-CLOCK CSR VALUE TO ENABLE THE CLOCK
1579      000111  PCLKEN==111      ; P-CLOCK CSR VALUE TO ENABLE THE CLOCK
1580      001600  PCLKCT==1600      ; P-CLOCK COUNT SET REGISTER FOR COUNTER
1581
1582      ; SPECIAL CLI CODES FOR "CHAR" ARGUMENT IN CLI CALLS
1583      ; (COMMAND LINE INTERPRETER DEFINITIONS)
1584      000000  CLIERR= 0
1585      000001  CLIEXI= 1
1586      000002  CLIBR = 2
1587      000003  CLIBIF= 3
1588      000004  CLISPA= 4
1589      000005  CLINUM= 5
1590      000006  CLIALP= 6
1591      000007  CLIALN= 7
1592      000010  CLIOCT= 8.
1593      000011  CLIDEC= 9.
1594      000012  CLISTR= 10.
1595
1596      ;DEFS FOR COMMAND LINE INTERPRETATION ACTION VALUES
1597
1598      000000  NULL=0
1599      000001  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	CNODAL=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	CDEFLT=44
1635	000045	CUNSVF=45
1636	000046	RSPOND=46
1637		:
1638		:
1639		:
1640	000000	MESSAGE TYPE VALUES
1641	000001	ALPHA==0
1642	000002	ONES==1
1643	000003	ZEROS==2
1644	000004	ONEALT==3
1645	000005	ZROALT==4
1646	000006	CCITT==5
1647		OPRSEL==6
1648		:
1649		:
		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 VINVXM         == 04          ; XMIT LIST INVALID BIT
1667 000020 MINVXM         == 20          ;
1668 000005 VINVRC         == 05          ; RECEIVE LIST INVALID BIT
1669 000040 MINVRC         == 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  
 1823 000014  
 1824 000016  
 1825 000020  
 1826 000024  
 1827 000030  
 1828 000032  
 1829 000034  
 1830 000040  
 1831 000044  
 1832 000050  
 1833 000054  
 1834 000060  
 1835 000064  
 1836 000070  
 1837 000072  
 1838 000074  
 1839  
 1840  
 1841  
 1842 000001  
 1843 000002  
 1844 000004  
 1845 000010  
 1846  
 1847  
 1848  
 1849 000001  
 1850 000002  
 1851 000004  
 1852 000010  
 1853  
 1854  
 1855  
 1856 000000  
 1857 000001  
 1858 000000  
 1859 000001  
 1860 000002  
 1861 000000  
 1862 000001  
 1863  
 1864  
 1865  
 1866 000002  
 1867 000001  
 1868 000005  
 1869  
 1870  
 1871  
 1872 000000  
 1873 000004  
 1874 000006  
 1875 000010  
 1876 000012  
 1877 000014  
 1878 000016

C.MREC == 10  
 C.RERB == 14  
 C.RERR == 16  
 C.RDAT == 20  
 C.RMDB == 24  
 C.RLIN == 30  
 C.RLEX == 32  
 C.PXMT == 34  
 C.MXMT == 40  
 C.PXM3 == 44  
 C.PXM2 == 50  
 C.PXMD == 54  
 C.XDAT == 60  
 C.XMDB == 64  
 C.XABB == 70  
 C.XABT == 72  
 C.COLL == 74

;  
 ; XMIT ABORT REASON MASK BIT DEFINITIONS  
 ;  
 C.RTRY == BIT00  
 C.LCAR == BIT01  
 C.SHRT == BIT02  
 C.OPEN == BIT03

;  
 ; RECEIVE ERROR REASON MASK BIT DEFINITIONS  
 ;  
 C.CRC == BIT00  
 C.FRAM == BIT01  
 C.MLEN == BIT02  
 C.RUNT == BIT03

;  
 ; ROUTINE FUNCTION CODES  
 ;  
 BMPRCV == 0 ; UPDATE RECEIVE COUNTERS  
 BMPXMT == 1 ; UPDATE TRANSMIT COUNTERS  
 INIADR == 0 ; INITIALIZE ADDRESS SETUP TABLE  
 ADDMUL == 1 ; ADD MULTICAST ADDRESS TO SETUP TABLE  
 KILMUL == 2 ; KILL MULTICAST ADDRESS IN SETUP TABLE  
 XMTSET == 0 ; XMIT AN ADDRESS SETUP PACKET  
 XMTDAT == 1 ; XMIT A DATA PACKET

;  
 ; FUNCTION CODES FOR LOOP-SERVER PACKETS  
 ;  
 FORWRD == 2  
 REPLY == 1  
 IDFUNC == 5

;  
 ; QNA DEVICE REGISTER OFFSETS  
 ;  
 NETADD == 0  
 LORCV == 4  
 HIRCV == 6  
 LOXMT == 10  
 HIXMT == 12  
 VECTOR == 14  
 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
1938 003224 055120  HLPTAB: .WORD HELP1 ;MESSAGE INPUT OCCURED, 0 FOR GOOD INPUT
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

;MESSAGE TYPE ASCII ADDRESS TABLE

; THIS SECTION DEFINES THE DATA PATTERNS USED BY THE EXERCISER

1977				
1978				
1979				
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

; THE NUMBER OF BYTES IN EACH MESSAGE

1988			MSGAD::	
1989	003360		.WORD	MSG00
1990	003360	003376	.WORD	MSG01
1991	003362	003526	.WORD	MSG02
1992	003364	003527	.WORD	MSG03
1993	003366	003530	.WORD	MSG04
1994	003370	003531	.WORD	MSG05
1995	003372	003532	.WORD	OPSLBF
1996	003374	003632	.WORD	

1997				
1998	003376	040	041	042
	003401	043	044	045
	003404	046	047	050
	003407	051	052	053
	003412	054	055	057
	003415	060	061	062
	003420	063	064	065

MSG00:: .ASCII \ !"#%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ\



	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				MSG05::	; CCITT 511 BIT TEST PATTERN
2010	003532	177603	157427	031011	.WORD 177603,157427,031011,047321,163715,105221	
	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 ; BUS ADDRESSES OF QNA HARDWARE REGISTERS
2038 ;
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 ADDRRES
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      ; XMIT RING DESCRIPTOR ENTRY FOR SETUP MODE
2093
2094 004060      ; LOXTGT:      ; LOW ADRS OF TARGET ADRS TBL DESCRPTR
2095 004060 000000      .WORD 0      ; AGAIN FOR LAST ENTRY
2096 004062 100000      .WORD 100000 ; SET VALID BIT
2097 004064 023570      .WORD TGTADR ; ADDRESS OF TARGET FIELD
2098 004066 000000      .WORD 0
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      ; CITIZENSHIP TEST SETUPS
2105
2106      ;RECEIVE BUFFERS
2107      SMLSIZ == 64.      ;SMALLEST SIZE IN BYTES OF ETHERNET PACKET
2108      RBUFNR == 105.    ;NUMBER OF RECEIVE BUFFERS
2109 004100      RBUFB: .BLKB RBUFNR*SMLSIZ ;100 RECEIVE BUFFERS, 64 BYTES EACH.
2110
2111      ;RECEIVE DESCRIPTORS
2112
2113      ;THERE ARE RBUFNR+1 DESCRIPTORS. ALL OF THEM EXCEPT THE LAST ONE
2114      ; ARE DATA DESCRIPTORS, I.E. THE ONLY CHAIN DESCRIPTOR IS THE LAST
2115      ; ONE TO CREATE A RING. COMMUNICATIONS WITH THE TESTER IS
2116      ; HALF-DUPLEX, I.E. PACKETS ARE RECEIVED ONE AT A TIME UNTIL THE
2117      ; COMPLETE MESSAGE IS RECEIVED.
2118
2119      RDESSZ == 12.      ;SIZE IN BYTES OF EACH RECEIVE DESCRIPTOR
2120      RDESAL == RDESSZ*RBUFNR ;SIZE IN BYTES OF ALL DESCRIPTORS
2121 021200      RDESC: .BLKB RDESAL ;STORAGE FOR RBUFNR RECEIVE DESCRIPTORS
2122 023554      .BLKB RDESSZ ; PLUS CHAIN DESCRIPTOR TO GIVE A RING
2123      .EVEN
2124
2125
2126      ;CITIZENSHIP WORK BUFFER
2127
2128      ;THE BOOT/DIAG ROM LOADS INTO THE FIRST 4KB OF THE RECEIVE BUFFERS. THE 4KW
2129      ; WORK BUFFER EXTENDS FROM THERE THROUGH THE TRANSMIT BUFFERS INTO THE
2130      ; STORAGE AREA THAT IS CREATED BELOW.
2131      CITWORK == RBUFB+4096. ;BEGINNING OF 4KW WORK BUFFER
2132      CITWEND == CITWORK+4096. ;END+1 OF WORK BUFFER
2133
2134
2135      ; TABLE OF TARGET ADDRESSES USED IN SETUP MODE
2136
2137 023570      ; TGTADR:      ; BYTE 0 OF ALL ADDRESSES 1-7
2138 023570      .BLKB 8.      ; BYTE 1 " " " "
2139 023600      .BLKB 8.      ; BYTE 2 " " " "
2140 023610      .BLKB 8.      ; BYTE 3 " " " "
2141 023620      .BLKB 8.      ; BYTE 4 " " " "
2142 023630      .BLKB 8.

```

```

2143 023640      .BLKB  8.      ; BYTE 5 " " " "
2144 023650      .BLKB  8.      ; BYTE 6 " " " "
2145 023660      .BLKB  8.      ; BYTE 7 " " " "
2146 023670      .BLKB  8.      ; BYTE 0 OF ALL ADDRESSES 8 - 14
2147 023700      .BLKB  8.      ; BYTE 1 " " " "
2148 023710      .BLKB  8.      ; BYTE 2 " " " "
2149 023720      .BLKB  8.      ; BYTE 3 " " " "
2150 023730      .BLKB  8.      ; BYTE 4 " " " "
2151 023740      .BLKB  8.      ; BYTE 5 " " " "
2152 023750      .BLKB  8.      ; BYTE 6 " " " "
2153 023760      .BLKB  8.      ; BYTE 7 " " " "
2154 023770      ;
2155 023770 000200  TGTSIZ:  .WORD  TGTSIZ - TGTADR      ; SIZE OF THE TARGET ADDRESS
2156      ;
2157      ; RECEIVE BUFFER FOR TARGET ADDRESSES LOOPED BACK IN SETUP MODE
2158      ;
2159 023772      ;
2160 023772      ; RCVTGT:
2161 024002      .BLKB  8.      ; BYTE 0 OF ALL ADDRESSES 1-7
2162 024012      .BLKB  8.      ; BYTE 1 " " " "
2163 024022      .BLKB  8.      ; BYTE 2 " " " "
2164 024032      .BLKB  8.      ; BYTE 3 " " " "
2165 024042      .BLKB  8.      ; BYTE 4 " " " "
2166 024052      .BLKB  8.      ; BYTE 5 " " " "
2167 024062      .BLKB  8.      ; BYTE 6 " " " "
2168 024072      .BLKB  8.      ; BYTE 7 " " " "
2169 024102      .BLKB  8.      ; BYTE 0 OF ALL ADDRESSES 8 - 14
2170 024112      .BLKB  8.      ; BYTE 1 " " " "
2171 024122      .BLKB  8.      ; BYTE 2 " " " "
2172 024132      .BLKB  8.      ; BYTE 3 " " " "
2173 024142      .BLKB  8.      ; BYTE 4 " " " "
2174 024152      .BLKB  8.      ; BYTE 5 " " " "
2175 024162      .BLKB  8.      ; BYTE 6 " " " "
2176      .BLKB  8.      ; BYTE 7 " " " "
2177      ;
2178      ; TABLE OF START ADDRESS OF RECEIVE RING BUFFERS
2179 024172      ;
2180 024172 025674  RRNGTB::
2181 024174 026774  .WORD  RRG001
2182 024176 030074  .WORD  RRG002
2183 024200 031174  .WORD  RRG003
2184 024202 032274  .WORD  RRG004
2185 024204 033374  .WORD  RRG005
2186 024206 034474  .WORD  RRG006
2187 024210 035574  .WORD  RRG007
2188      .WORD  RRG010
2189      ;
2190      ; TABLE OF START ADDRESS OF TRANSMIT RING BUFFERS
2191 024212      ;
2192 024212 024574  XRNGTB::
2193      .WORD  XRG001
2194      ;
2195 024214 024236  ;
2196 024216 024260  XRGSR:  .WORD  XRING      ; FIRST ENTRY IN TRANSMIT RING
2197 024220 024236  RRGSR:  .WORD  RRING      ; FIRST ENTRY IN RECEIVE RING
2198 024222 024260  XRGCR:  .WORD  XRING      ; CURRENT ENTRY IN TRANSMIT RING
2199 024224 024236  RRGCR:  .WORD  RRING      ; CURRENT ENTRY IN RECEIVE RING
                XRGNT:  .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 DESCRIPOTR POINTER
2204 ;
2208 024236 XRING::
2209 024236 RRGFRM XRG00,B,0 ; FLAG WORD
024236 000000 .WORD 0 ; SET VALID BIT (DSCR/HIGH ADDRESS WRD)
024240 100000 .WORD 100000 ; SEGMENT BUFFER ADDRESS
024242 024574 .WORD XRG001 ; BUFFER SIZE
024244 000000 .WORD 0 ; STATUS WORD 1
024246 000000 .WORD 0 ; STATUS WORD 2
024250 000000 .WORD 0
2210 024252 XRINGH:
2211 024252 XRGINV: .WORD 0 ; JUST ENOUGH OF THE NEXT DESCRIPTOR
2212 024252 000000 .WORD 0 ; TO INVALIDATE LIST
2213 024254 000000
2214 024256 RNGSIZ: .WORD . - XRINGH ; SIZE OF DESCRIPTORS
2215 024256 000004
2219 024260 RRING::
2220 000007 .REPT 7
2221 .NLIST
2222 RRGFRM RRG00,B,-RPKLEN/2 ; ENTER BUFF ADD, BUF OFFSET, BUF SIZE
2223 .LIST
2224 .ENDR
2225 ;
2226 024404 000000 .WORD 0 ; AGAIN FOR LAST ENTRY (FLAG WRD)
2227 024406 100000 .WORD 100000 ; DSCR/HIGH ADDRESS WORD
2228 024410 035574 .WORD RRG010 ; LOW ADDRESS WORD
2229 024412 177340 .WORD -RPKLEN/2 ; BUFFER SIZE (TWO'S COMPLEMENT)
2230 024414 000000 .WORD 0 ; STATUS WRD 1
2231 024416 000000 .WORD 0 ; STATUS WRD 2
2232 ;
2236 ;
2237 000007 .REPT 7
2238 .NLIST
2239 RRGFRM RRG01,B,-RPKLEN/2 ; ENTER BUFF ADD, BUF OFFSET, BUF SIZE
2240 .LIST
2241 .ENDR
2242 024544 RRINGH::
2243 024544 000000 .WORD 0 ; AGAIN FOR LAST ENTRY (FLAG WRD)
2244 024546 000000 .WORD 0 ; LEAVE THIS ONE INVALID
2245 024550 046574 .WORD RRG020 ; LOW ADDRESS WORD
2246 024552 177340 .WORD -RPKLEN/2 ; BUFFER SIZE (TWO'S COMPLEMENT)
2247 024554 000000 .WORD 0 ; STATUS WRD 1
2248 024556 000000 .WORD 0 ; STATUS WRD 2
2249 024560 RRGCHN::
2250 024560 000000 .WORD 0
2251 024562 140000 .WORD 140000 ; DESCRIPTOR TO CHAIN TO BEGINNING OF RING
2252 024564 024260 .WORD RRING ; ADDRESS OF START OF DESCRIPTOR RING
2253 024566 000000 .WORD 0 ; BYTE COUNT (UNNECESSARY)
2254 024570 000000 .WORD 0 ; STAT1
2255 024572 000000 .WORD 0 ; STAT2
2256 ;
2257 ;
2258 024574 XRG001:: .BLKB XPKLEN ; XMIT RING BUFFER
2259 ;

```

2260 025674  
 2261 026774  
 2262 030074  
 2263 031174  
 2264 032274  
 2265 033374  
 2266 034474  
 2267 035574  
 2268 036674  
 2269 037774  
 2270 041074  
 2271 042174  
 2272 043274  
 2273 044374  
 2274 045474  
 2275 046574  
 2276  
 2277  
 2278  
 2279 047674  
 2280 047674  
 2281 047702  
 2282 047710 001140  
 2283 047712 000034  
 2284 047714 007  
 2285 047715 000  
 2286 047716 046507  
 2287 047720 000000  
 2288 047722 000  
 2289 047723 000  
 2290 047724 000  
 2291 047725 000  
 2292 047726 000002  
 2293 047730 001  
 2294 047731 000  
 2295 047732 000  
 2296 047733 007  
 2297 047734 000  
 2298 047735 006  
 2299 047736  
 2300 047744 144  
 2301 047745 000  
 2302 047746 001  
 2303 047747 005  
 2304 047750  
 2305  
 2306 047770  
 2307 047770 000074  
 2308  
 2309  
 2310  
 2311  
 2312  
 2313 047772 000000  
 2314 047774 000000  
 2315 047776 000000  
 2316

RRG001:: .BLKB RPKLEN ; RECEIVE RING BUFFERS  
 RRG002:: .BLKB RPKLEN  
 RRG003:: .BLKB RPKLEN  
 RRG004:: .BLKB RPKLEN  
 RRG005:: .BLKB RPKLEN  
 RRG006:: .BLKB RPKLEN  
 RRG007:: .BLKB RPKLEN  
 RRG010:: .BLKB RPKLEN  
 RRG011:: .BLKB RPKLEN  
 RRG012:: .BLKB RPKLEN  
 RRG013:: .BLKB RPKLEN  
 RRG014:: .BLKB RPKLEN  
 RRG015:: .BLKB RPKLEN  
 RRG016:: .BLKB RPKLEN  
 RRG017:: .BLKB RPKLEN  
 RRG020:: .BLKB RPKLEN  
 ;  
 ; CONSOLE ID INFORMATION  
 ;  
 IDTDAT:  
 .BLKB 6 ; DESTINATION ADDRESS FIELD  
 .BLKB 6 ; SOURCE ADDRESS FIELD  
 .WORD 1140 ; CONSOLE ID TYPE CODE  
 .WORD 28. ; NUMBER OF VALID DATA BYTES TO FOLLOW  
 .BYTE 7 ; SYSTEM ID FUNCTION CODE  
 .BYTE 0 ; PAD BYTE  
 .WORD 46507 ; RECEIPT NUMBER ;; N.M. (CHANGED FROM 51115 TO 46507)  
 .WORD 0 ; MOP VERSION TYPE  
 .BYTE 0 ; MOP VERSION LENGTH  
 .BYTE 0 ; MOP VERSION VERSION  
 .BYTE 0 ; MOP VERSION ECO  
 .BYTE 0 ; MOP VERSION USER ECO  
 .WORD 2 ; FUNCTION TYPE  
 .BYTE 1 ; FUNCTION LENGTH  
 .BYTE 0 ; MAINTENANCE FUNCTIONS SUPPORTED 1  
 .BYTE 0 ; MAINTENANCE FUNCTIONS SUPPORTED 2  
 .BYTE 7 ; HARDWARE ADDRESS TYPE  
 .BYTE 0 ; SECOND BYTE OF ADDRESS TYPE  
 .BYTE 6 ; HARDWARE ADDRESS LENGTH  
 .BLKB 6 ; HARDWARE ADDRESS VALUE  
 .BYTE 100. ; DEVICE TYPE FIELD  
 .BYTE 0 ; DEVICE TYPE FIELD BYTE 2  
 .BYTE 1 ; LENGTH OF DEV TYPE FIELD  
 .BYTE 5 ; DEQNA DEVICE TYPE  
 .BLKB 65.-49. ; PAD THE REST OF THE PACKET ;; N.M. (CHANGED 43 TO 49)  
 ; ; N.M. (USED TO BE .EVEN HERE)  
 IDTSIZ:  
 .WORD . - IDTDAT  
 ;  
 ; INFORMATION ABOUT THE CURRENT UNIT AS OBTAINED FROM THE HARDWARE P-TABLE  
 ;  
 ; PTABLE INFO STORAGE FOR UNIT 0  
 ;  
 QNAADO:: .WORD 0 ; DEVICE ADDRESS FROM PTABLE  
 QNAVCO:: .WORD 0 ; VECTOR ADDRESS FROM PTABLE  
 QNAPRO:: .WORD 0 ; PRIORITY FROM PTABLE  
 ;

```

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 QNAVC1:: .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 CULLISON 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: .WORD 3      ; BYTE COUNT (=3 FOR REQUEST ID)
2417 050340 000003      .WORD IDFUNC      ; FUNCTION CODE FOR REQUEST ID
2418 050342 000005      .WORD "MR"      ; RECEIPT NUMBER
2419 050344 051115
2420      ;
2421      ; LOOP DIRECT MESSAGE
2422      ;
2423      .EVEN
2424 050346      LOPDIR: .WORD 0      ; SKIP COUNT
2425 050346 000000      .WORD FORWRD      ; FUNCTION = FORWARD DATA
2426 050350 000002      .WORD 0,0,0      ; LOCAL NODE ADDRESS
2427 050352 000000 000000 000000      .WORD 1      ; FUNCTION = REPLY
2428 050360 000001      .WORD 0,0,0      ; LOCAL NODE ADDRESS
2429 050362 000000 000000 000000
2430      ;

```



```

2431 ; TRANSMIT ASSIST MESSAGE
2432 ;
2433 ;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 ;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 ;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 050516 CLI CLISPA,0,N10$ ;SKIP ANY LEADING SPACES
2484 050522 N10$: CLI <'?'>,HELP,N12$ ;IS THE FIRST NON-SP CHAR. A "?"
2485 050526 CLI CLIEXI,0 ; IF YES DO "HLP" AND EXIT
2486 050530 N12$: CLI CLISTR,HELP,N14$,<'HELP'> ;ELSE IS FIRST WORD A "HELP"
2487 050544 CLI CLIEXI,0 ; IF YES DO "HLP" AND EXIT
2488 050546 N14$: CLI CLISTR,NOTNUF,N16$,<'NODE'> ;ELSE IS FIRST WORD A "NODE"
2489 050562 CLI CLIBR,0,N80$ ; IF YES, BR N80$
2490 050566 N16$: CLI CLISTR,BUILD,N18$,<'BUILD'> ;ELSE IS FIRST WORD A "BUILD"
2491 050602 CLI CLIEXI,0 ; IF YES DO "BUILD" AND EXIT
2492 050604 N18$: CLI CLISTR,NOTNUF,N20$,<'RUN'> ;ELSE IS FIRST WORD A "RUN"
2493 050616 CLI CLIBR,0,N180$ ; IF YES, BR N180$
2494 050622 N20$: CLI <'S'>,NOTNUF,N25$ ;ELSE IS FIRST CHAR. A "S"
2495 050626 CLI CLISTR,0,N22$,<'HOW'> ; IF YES IS REST OF WORD "HOW"
2496 050640 CLI CLIBR,0,N100$ ; IF YES, BR N100$
2497 050644 N22$: CLI CLISTR,SUMMARY,N23$,<'UMMARY'> ; ELSE IS REST OF WORD "UMMARY"
2498 050662 CLI CLIEXI,0 ; IF YES, DO "SUMM" AND EXIT
2499 050664 N23$: CLI CLISTR,CSAVE,N24$,<'AVE'> ; ELSE IS REST OF WORD "AVE"
2500 050676 CLI CLIEXI,0 ; IF YES, DO "SAVE" AND EXIT
2501 050700 N24$: CLI CLIERR,0 ; ELSE "ILL COMMAND"
2502 050702 CLI CLIEXI,0 ; EXIT
2503 050704 N25$: CLI CLISTR,NOTNUF,N26$,<'CLEAR'> ;ELSE IS FIRST WORD A "CLEAR"
2504 050720 CLI CLIBR,0,N120$ ; IF YES, BR N120$
2505 050724 N26$: CLI CLISTR,NOTNUF,N28$,<'IDENTIFY'> ;ELSE IS FIRST WORD "IDENTIFY"
2506 050744 CLI CLIBR,0,N140$ ; IF YES, GET ADDR, BR N140$
2507 050750 N28$: CLI CLISTR,NOTNUF,N29$,<'MESSAGE'> ;ELSE IS FIRST WORD "MESSAGE"
2508 050766 CLI CLIBR,0,N160$ ; IF YES, BR N160$
2509 050772 N29$: CLI CLISTR,CUNSAV,N30$,<'UNSAVE'> ;ELSE IS FIRST WORD "UNSAVE"
2510 051010 CLI CLIBR,0,N210$ ; IF YES, BR TO N210$
2511 051014 N30$: CLI CLISTR,EXIT,N31$,<'EXIT'> ;ELSE IS FIRST WORD "EXIT"
2512 051030 CLI CLIEXI,0 ; IF YES EXIT
2513 051032 N31$: CLI CLISTR,NOTNUF,N32$,<'FUNCTION'> ;ELSE IS FIRST WORD "FUNCTION"
2514 051052 CLI CLIBR,0,N200$ ; IF YES, BR N200$
2515 051056 N32$: CLI CLISTR,RSPOND,N33$,<'RESPOND'> ;ELSE IS FIRST WORD "RESPOND"
2516 051074 CLI CLIEXI,0 ; IF YES LEAVE
2517 051076 N33$: CLI CLIERR,0 ;OTHERWISE "ILL CMD".
2518 051100 CLI CLIEXI,0 ; EXIT
2519 ;
2520 ; SECOND KEYWORD (ADR/TYPE) FOR NODE COMMAND
2521 ;
2522 051102 N80$: 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 DELIMETER, 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$: CLI      CLIEXI,CRUN ; RUN TEST AND EXIT
2616 ;
2617 ; REMAINING COMMAND LINE FOR FUNCTION COMMAND
2618 ;
2619 052046      N200$: 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                                ERRTBL                                L$ERRTBL::
      052074                                .WORD 0
      052074 ERRRTYP::                                .WORD 0
      052076 ERRNBR::                                .WORD 0
      052100 ERRMSG::                                .WORD 0
      052102 ERRBLK::                                .WORD 0
2639 052104 CLI$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: .ASCII /%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 NULSTR: .ASCIZ /%N%AA ZERO LENGTH STRING WAS ENTERED./
2654 053144 NODADR: .ASCIZ /%N%S2%T/
2655 053154 LOGNAM: .ASCIZ /%S14%AN%D2/
2656 053167 NODTYP: .ASCIZ /%S14%T/
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%AHOWEVER, 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 HDMSG1: .ASCIZ /%N%A 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[SH]OW [N]ODES - PRINTS INFORMATION IN CURRENT NODE TABLE.\
2688 055536 HELP6: .ASCIZ \#N2#A[SH]OW [M]ESSAGE - PRINTS THE SELECTED MESSAGE TYPE, SIZE AND COPIES.\
2689 055651 HELP7: .ASCIZ \#N2#A[SH]OW [C]OUNTERS - PRINTS THE LOW LEVEL COUNTERS OF THE HOST 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/TYPE - 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]L - 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[S]AVE - 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 MESPAT: .ASCIZ /#A DATA PATTERN: #T/
2729 061315 ALLNOD: .ASCIZ /ALL NODE/
2730 061326 LUPAIR: .ASCIZ /LOOPPAIR/
2731 061337 DIRECT: .ASCIZ /LOOP DIRECT/
2732 061353 FULAST: .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#S5#ACONTENTS OF NODE #T#A INTERNAL COUNTERS:/
2774 062127  CNTR01: .ASCIZ /#N#2#ASECONDS SINCE LAST ZEROED:#S15#Z5/
2775 062176  CNTR02: .ASCIZ /#N#APACKETS RECEIVED:#S19#T/
2776 062232  CNTR03: .ASCIZ /#N#AMULTICAST PACKETS RECEIVED:#S9#T/
2777 062277  CNTR04: .ASCIZ /#N#APACKETS REC'D WITH ERROR - BITMAP:#S8#B4/
2778 062354  CNTR05: .ASCIZ /#N#APACKETS RECEIVED WITH ERROR:#S13#Z5/
2779 062424  CNTR06: .ASCIZ /#N#ADATA BYTES RECEIVED:#S16#T/
2780 062463  CNTR07: .ASCIZ /#N#AMULTICAST DATA BYTES RECEIVED:#S6#T/
2781 062533  CNTR08: .ASCIZ /#N#ARECEIVED PACKETS LOST-INTERNAL:#S10#Z5/
2782 062606  CNTR09: .ASCIZ /#N#ARECEIVED PACKETS LOST-LOCAL:#S12#Z5/
2783 062657  CNTR10: .ASCIZ /#N#APACKETS TRANSMITTED:#S16#T/
2784 062716  CNTR11: .ASCIZ /#N#AMULTICAST PACKETS TRANSMITTED:#S6#T/
2785 062766  CNTR12: .ASCIZ /#N#APACKETS TRANSMITTED 3+ TRYS:#S8#T/
2786 063034  CNTR13: .ASCIZ /#N#APACKETS TRANSMITTED 2 TRYS:#S9#T/
2787 063101  CNTR14: .ASCIZ /#N#APACKETS DEFERRED:#S19#T/
2788 063135  CNTR15: .ASCIZ /#N#ADATA BYTES TRANSMITTED:#S13#T/
2789 063177  CNTR16: .ASCIZ /#N#AMULTICAST BYTES TRANSMITTED:#S8#T/
2790 063245  CNTR17: .ASCIZ /#N#ATRANSMIT PACKETS ABORTED-BITMAP:#S8#B6/
2791 063320  CNTR18: .ASCIZ /#N#ATRANSMIT PACKETS ABORTED:#S16#Z5/
2792 063365  CNTR19: .ASCIZ /#N#AXMIT 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 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  \NAILLEGAL 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          BGNMSG  ERR1
2880          070306          MOV      #CEXIT,CFLAG
2881          070306          PRINTX   #PCMSG,PCCALL
2882          070314          012737  000020  003734
2883          070314          013746  050330
2884          070320          012746  067542
2885          070324          012746  000002
2886          070330          010600
2887          070332          104415
2888          070334          062706  000006
2889          070340          DOCLN
2890          070340          104444
2891          070342          ENDMSG
2892          070342          104423
2893          070344          BGNMSG  ERR2
2894          070344          MOV      R1,-(SP)
2895          070346          013701  002372          MOV      P$TYPE,R1
2896          070352          006301          ASL     R1
2897          070354          062701  003324          ADD     #MSGTAB,R1
2898          070360          PRINTX   #EMSG35,#STRBUF
2899          070360          012746  002324
2900          070364          012746  064367
2901          070370          012746  000002
2902          070374          010600
2903          070376          104415
2904          070400          062706  000006
2905          070404          PRINTX   #EMSG36,(R1)
2906          070404          011146
2907          070406          012746  064424
2908          070412          012746  000002
2909          070416          010600
2910          070420          104415
2911          070422          062706  000006
2912          070426          012601
2913          070430          ENDMSG          MOV      (SP)+,R1
2914          070430          104423
2915          070432          BGNMSG  ERR3
2916          070432          PRINTX   #EMSG37,#STRBUF
2917          070432          012746  002324
2918          070436          012746  064451
2919          070442          012746  000002
2920          070446          010600
2921          070450          104415
    
```

```

ERR1::
    MOV      PCCALL,-(SP)
    MOV      #PCMSG,-(SP)
    MOV      #2,-(SP)
    MOV      SP,R0
    TRAP    C$PNTX
    ADD     #6,SP

L10002:
    TRAP    C$MSG

ERR2::
    MOV      #STRBUF,-(SP)
    MOV      #EMSG35,-(SP)
    MOV      #2,-(SP)
    MOV      SP,R0
    TRAP    C$PNTX
    ADD     #6,SP

    MOV      (R1),-(SP)
    MOV      #EMSG36,-(SP)
    MOV      #2,-(SP)
    MOV      SP,R0
    TRAP    C$PNTX
    ADD     #6,SP

L10003:
    TRAP    C$MSG

ERR3::
    MOV      #STRBUF,-(SP)
    MOV      #EMSG37,-(SP)
    MOV      #2,-(SP)
    MOV      SP,R0
    TRAP    C$PNTX
    
```

```

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

```

PRINTX #EMSG38,#STRBU1

```

ADD #6,SP
MOV #STRBU1,-(SP)
MOV #EMSG38,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C#PNTX
ADD #6,SP

```

ENDMSG

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.
: *****
:

```

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

.SBTTL CLKSET CLOCK SETUP SUBROUTINE

```

:--*
: 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:
      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"
      RTS      PC

```

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  
3072  
3073  
3074  
3075  
3076  
3077  
3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087  
3088  
3089  
3090  
3091  
3092  
3093  
3094  
3095  
3096

070530  
070530  
070530 005077 113202  
070534 005337 003754  
070540 001017  
070542 013737 003744 003754  
070550 005237 050146  
070554 005237 003752  
070560 022737 000074 003752  
070566 001004  
070570 005237 003750  
070574 005037 003752  
070600  
070600 005737 003756  
070604 001402  
070606 005337 003756  
070612  
070612 005737 003760  
070616 001402  
070620 005337 003760  
070624  
070624 005737 003762  
070630 001406  
070632 023737 003744 003754  
070640 001002

.SBTTL CLKINT CLOCK INTERRUPT SERVICE ROUTINE

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

BGNSRV CLKINT

CLKINT::

: CLR @CLKCSR ; DISABLE THE CLOCK FROM INTERRUPTING  
: DEC TIMTCK ; DECREMENT THE NO. OF TICKS/SEC  
: BNE 10\$ ; GO CHECK TIMERS  
: MOV CLKHZ,TIMTCK ; RESET THE NO. OF TICKS/SEC.  
: INC CNTRS+C.SECONDS ; INCREMENT QNA'S NUMBER OF SECS  
: INC TIMSEC ; INC. NO OF SECS-SINCE-START  
: CMP #60.,TIMSEC ; SEE IF WE'VE COUNTED 60 SEC.S YET  
: BNE 10\$ ; IF NOT, GO CHECK TIMERS  
: INC TIMMIN ; ELSE, INC. MINUTES-SINCE-START  
: CLR TIMSEC ; AND RESTART SECOND COUNTER  
10\$: : TST TIMER1 ; SEE IF TIMER1 TIMING ANYTHING  
: BEQ 20\$ ; IF=0, NO, CHECK NEXT TIMER  
: DEC TIMER1 ; ELSE DECREMENT THE TIMER VALUE (BY 1 TICK)  
20\$: : TST TIMER2 ; SEE IF TIMER2 TIMING ANYTHING  
: BEQ 30\$ ; IF=0, NO, CHECK NEXT TIMER  
: DEC TIMER2 ; ELSE DECREMENT TIMER VALUE (BY 1 TICK)  
30\$: : TST TIMERS ; SEE IF TIMERS TIMING ANYTHING  
: BEQ 40\$ ; IF=0, NOTHING BE TIMED, LEAVE  
: CMP CLKHZ,TIMTCK ; SEE IF A SECOND HAS BEEN COUNTED OFF  
: BNE 40\$ ; BR IF NO

3097 070642 005337 003762  
3098 070646  
3099 070646 013777 003746 113062 40\$:  
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,(R4)+,R1 ;GET THE RELATIVE ADDRESS OF THE CALLED
3127 070670 012401 MOV    (R4)+,R1   ;ROUTINE.
3128          ;MAKE IT AN ABSOLUTE ADDRESS.
3129 070672 060701 ANCHOR: ADD    PC,R1
3130 070674          MOV    R4,-(SP)    ;SAVE THE RETURN TO THE CALLING ROUTINE.
3131 070674 010446 CMP    R5,SP      ;CHECK FOR STACK OVER-RUN.
3132 070676 020506 BLO   10$        ;
3133 070700 103401 HALT          ;HANDLE STACK OVER-RUN CONDITION.
3134 070702 000000
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)+,R2   ;
3139 070712 012602 MOV    (SP)+,R3   ;
3140 070714 012603 RTS    R4        ;BACK TO THE CALLING ROUTINE.
3141 070716 000204

```

```

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 ; INPUTS -
3155 ; NONE
3156 ;
3157 ; OUTPUTS-
3158 ; P1: SUCCESS/FAILURE 0=SUCCESS/-1=FAILURE
3159 ;
3160 ; CALLING SEQUENCE:
3161 ; CALL WAIT
3162 ; P$POP P1
3163 ;--
3164 WAIT::
3165 070720 012703 000012 MOV #10.,R3 ; MOVE NO. OF COUNTS TO R3
3166 070724 012704 003756 MOV #TIMER1,R4 ; AND TIMER TO BE USED TO R4
3167 070730 005002 CLR R2 ; LOCAL STATUS PARAMETER
3168 070732 010314 MOV R3,(R4) ; SET NUMBER OF TICKS. (GLOBAL)
3169 070734 10$: TST ERRFLG ; CHECK IF ERROR OCCURED
3170 070734 005737 050270 BNE 30$ ; BR IF YES
3171 070740 001011 TST DNIFLG ; CHECK FOR INTERRUPT
3172 070742 005737 050262 BEQ 20$ ; BR IF INTERRUPT RECEIVED
3173 070746 001403 CLR DNIFLG
3174 070750 005037 050262 BR 50$
3175 070754 000410
3176 070756 20$: TST (R4) ; HAS TIMER EXPIRED?
3177 070756 005714 BNE 10$ ; BR IF NO TO WAIT FOR INTERRUPT
3178 070760 001365 BR 40$ ; BR TO 40$
3179 070762 000403
3180 070764 30$: CALL ERROR ; CALL ERROR ROUTINE
3181 070764 004437 070656 JSR R4,PREG14
3182 070770 000106 .WORD ERROR-ANCHOR
3183 070772 012702 177777 40$: MOV #-1,R2 ; INDICATE FAILURE
3184 070776 50$: RETURN R2 ; RETURN WITH SUCCESS/FAILURE INDICATION
3185 070776 010225 MOV R2,(R5)+
071000 000207 RTS PC
    
```

3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197  
3198  
3199  
3200  
3201  
3202  
3203  
3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223  
3224

071002  
071002 005337 050270  
071006 005737 050254  
071012 001013  
071014 005737 050252  
071020 001017  
071022 005737 050266  
071026 001023  
071030 104455  
071032 000004  
071034 063563  
071036 070306  
071040 000424  
071042 104455  
071044 000001  
071046 063436  
071050 070306  
071052 005337 050254  
071056 000415  
071060 104455  
071062 000002  
071064 063467  
071066 070306  
071070 005337 050252  
071074 000406  
071076 104455  
071100 000003  
071102 063527  
071104 070306  
071106 005337 050266  
071112  
071112 000207

.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 QNAISR 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$ERDF
; .WORD 4
; .WORD EMSG04
; .WORD ERR1
10$: BR 40$ ; EXIT
ERRDF 1,EMSG01,ERR1 ;PORT COMMAND ERROR
;
; TRAP C$ERDF
; .WORD 1
; .WORD EMSG01
; .WORD ERR1
20$: DEC PCEFLG ; INDICATE THAT IT WAS HANDLED
BR 40$ ; EXIT
ERRDF 2,EMSG02,ERR1 ;QNA FATAL ERROR
;
; TRAP C$ERDF
; .WORD 2
; .WORD EMSG02
; .WORD ERR1
30$: DEC FATFLG ; KEEP UP ON BOOK KEEPING
BR 40$ ; EXIT
ERRDF 3,EMSG03,ERR1 ;UNEXPLAINED INTERRUPT
;
; TRAP C$ERDF
; .WORD 3
; .WORD EMSG03
; .WORD ERR1
40$: DEC BCOUNT ; BOOK KEEPING
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
:
: SIDEEFFECTS: 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     QNAIN1           ; TEST BEFORE? IF YES, BRANCH.
:   MOV     @1,FLAG1
:   MOV     @2,R1
:
:   MOV     @RDESC,R4         ;BUILD TWO RECEIVE DESCRIPTORS OF
:   MOV     @RBUF,R3         ; 2K BYTES EACH
:                               ;GET POINTER TO DESCRIPTOR
:                               ;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)+
3293 071252 005024      CLR     (R4)+
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     50$
3318
3319 071362 026427 000012 177777  CMP     STAT4(R4),#-1   ; NO, ERROR
3320 071370 001006      BNE     50$
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      SOB     50$:
3327 071406      ERRHRD 65,MSG65      ; NO, ERROR.
3328 071406 104456
3329 071410 000101      TRAP   C$ERRHRD
3330 071412 066705      .WORD 65
3331 071414 000000      .WORD MSG65
3332 071416 012700 177777      .WORD 0
3328 071416 012700 177777  MOV     #-1,R0          ; SET FAILURE STATUS
3329 071422 000137 072122  JMP     QNAEXI          ; AND EXIT
3330
3331 071426      SOB     70$:
3332 071426 013701 047772  MOV     QNAADO,R1      ; GET BASE ADDRESS
3333
3334 071432 010246      MOV     R2,-(SP)
3335 071434 010546      MOV     R5,-(SP)      ; SAVE R5 STACK POINTER

```



```

3381 071720 005303          DEC      R3
3382 071722 001354          BNE     10$
3383 071724 012703 000017  20$:    MOV     @15.,R3          ; ...AND RECEIVE RING
3384 071730                    ;
3385 071730 013701 024222  20$:    MOV     RRGCUR,R1      ; CLEAR FLAG WORD
3386 071734 005061 000000          CLR     FLAG(R1)      ; CLEAR STATUS 1
3387 071740 005061 000010          CLR     STAT1(R1)     ; CLEAR STATUS 2
3388 071744 005061 000012          CLR     STAT2(R1)     ; SET THE VALID BIT FOR THIS DESCRIPTOR
3389 071750 012761 100000 000002  MOV     @MVALID,DESC(R1) ; UPDATE, BUT DON'T VALDIATE PREVIOUS
3390 071756                    CALL    GETRNX @0,@RRGCUR ;
                                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     RRINGH+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     HIRCVR(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)+
                                072122 010025
                                072124 000207
                                RTS     PC

```

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
: . . . . . ENDOF
: . . . . 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  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524  
3525  
3526  
3527  
3528  
3529

072126

```

: . . . . . REPORT BYTE COUNT ERROR
: . . . . . ELSE
: . . . . . IF XMITTED SETUP DATA NOT = RECEIVED SETUP DATA
: . . . . . THEN
: . . . . . REPORT ADDRESS MISMATCH
: . . . . . ELSE
: . . . . . STATUS = SUCCESS
: . . . . . ENDFIF
: . . . . . ENDFIF
: . . . . . ENDFIF
: . . . . . ENDFIF
: . . . . . ENDFIF
: . . . . . ENDFIF
: . . . . . RETURN STATUS = STATUS
: . . . . . RETURN WITH RETURN STATUS
: . . . . . END
: . . . . . CALLING SEQUENCE
: . . . . . TBD
: . . . . . INPUT
: . . . . . NONE
: . . . . . IMPLICIT INPUT
: . . . . . NONE
: . . . . . OUTPUT
: . . . . . NONE
: . . . . . IMPLICIT OUTPUT
: . . . . . NONE
: . . . . . COMPLETION CODES
: . . . . . NONE
: . . . . . SIDE EFFECTS
: . . . . . NONE
: . . . . . REGISTERS USED
: . . . . . R0 - RETURN STATUS, SET BY CALLED ROUTINES
: . . . . . R1 - CASE OFFSET REG, ADDRESS GROUP POINTER
: . . . . . R2 - ADDRESS COLUMN POINTER
: . . . . . R3 - POINTS TO TABLE OF NI ADDRESSES
: . . . . . DEBUG
: . . . . . NONE
: . . . . . --
: . . . . . 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 014501              P$POP  R1          ; GET PARAMETER, BUT DON'T LOSE IT
                                MOV    -(R5),R1
3533 072140 100005              BPL    10$         ; BRANCH IF NOT UNDER RANGE
3534 072142 104454              ERRSF  50,EMSG50  ; REPORT RANGE UNDERFLOW
                                TRAP   C$ERSF
                                .WORD  50
                                .WORD  EMSG50
                                .WORD  0
3535 072152 000536              BR     SETEXI
3536 072154 000002              10$:  CMP    #2,R1   ; CHECK UPPER LIMIT OF CASE OFFSET
3537 072154 022701 000002      BGE    20$         ; BRANCH IF WITHIN RANG
3538 072160 002005              ERRSF  51,EMSG51  ; REPORT RANGE OVERFLOW
                                TRAP   C$ERSF
                                .WORD  51
                                .WORD  EMSG51
                                .WORD  0
3540 072172 000526              BR     SETEXI      ; LEAVE
3541 072174 006301              20$:  ASL    R1      ; TURN INTO WORD INDEX
3542 072174 062701 072204      ADD    #30$,R1    ; POINT TO WORD OFFSET
3543 072176 061107              ADD    (R1),PC    ; GO DO IT
3544 072202 000006              30$:  .WORD  SETINI - 30$ ; OFFSET TO INITIALIZE
3545 072204 000142              .WORD  SETADD - 30$ ; OFFSET TO SET DECNET MULTICAST
3546 072206 000204              .WORD  SETKLD - 30$ ; OFFSET TO KILL DECNET MULTICAST
3547 072210 012701 177700      SETINI: MOV    #-100,R1 ; GROUP POINTER
3548 072212 062701 000100      10$:  ADD    #100,R1   ; BUMP THE GROUP POINTER
3549 072216 001427              BEQ    20$         ; IF ZERO, DO THE FIRST GROUP
3550 072222 022701 000200      CMP    #200,R1   ; HAVE THE TWO GROUPS BEEN WRITTEN?
3551 072224 001024              BNE    20$         ; BRANCH IF NOT
3552 072232 013701 024220      ;
3553 072236 012761 177700 000006 ; MOV    XRGCUR,R1 ; GET XMIT PACKET ADDRESS
3554 072244 004437 070656      MOV    #-100,WRDCNT(R1) ; WRITE ALL ADDRESSES
3555 072252 001560              CALL   SETWRT     ; WRITE THEM
                                JSR    R4,PREG14
                                .WORD  SETWRT-ANCHOR
3560 072254 014500              P$POP  R0         ; ANY ERROR
3561 072256 001075              MOV    -(R5),R0  ; IF YES, GO WRITE THE ADDRESSES
3562 072262 013701 024220      BNE    SETEXI    ; ELSE, GET NXT XMIT DESCRIPTOR
3563 072270 012761 177705 000006 ; MOV    XRGCUR,R1 ; WORD COUNT TO INIT THE MODE BITS
3564 072274 004437 070656      MOV    #-73,WRDCNT(R1) ; WRITE THE PACKET
3565 072276 001560              CALL   SETWRT     ;
                                JSR    R4,PREG14
                                .WORD  SETWRT-ANCHOR
3566 072300 014500              P$POP  R0         ; GET STATUS
3567 072302 000463              BR     SETEXIT    ; LEAVE
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 005202              20$:  INC    R2      ; BUMP THE COLUMN POINTER
3571 072312 005202              30$:

```

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		; ELSE, 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	RO		; GET THE RETURNED STATUS
	072404	014500					MOV -(R5),RO	
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	RO		; GET THE RETURNED STATUS
	072446	014500					MOV -(R5),RO	
3602	072450			SETEXI:				
3603	072450				RETURN	RO		; RETURN
	072450	010025					MOV RO,(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

```

: * *
: 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,PREG14
:     .WORD  XMIT-ANCHOR
: NON EXISTENT MEMORY?
: BRANCH IF NOT
:
:     INC     R0
:     MOV     XRG NXT,R4
:     MOV     #TGTADR,LOADD(R4)
:     CALL    XMIT    #XMTSET
:
:     TST     FATFLG
:     BEQ     20$
:
:     10$:

```

```

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

```

```

3659 072510 005037 050252          CLR  FATFLG          ; CLEAR FATAL ERROR FLAG
3660 072514          ERRHRD 43,MSG43    ; REPORT TIMEOUT
      072514 104456          TRAP  C$ERHRD
      072516 000053          .WORD 43
      072520 065062          .WORD MSG43
      072522 000000          .WORD 0
3661 072524 000504          BR    WRTEXI        ; LEAVE
3662 072526          20$: 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
      072532 104456          TRAP  C$ERHRD
      072534 000055          .WORD 45
      072536 065237          .WORD MSG45
      072540 000000          .WORD 0
3666 072542 000475          BR    WRTEXI        ; EXIT SETUP, FAILURE OCCURRED
3667 072544          30$: CALL  RECEVE      ; PACKET FOUND?
3668 072544          004437 070656          JSR   R4,PREG14
      072550 003044          .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
      072572 104456          TRAP  C$ERHRD
      072574 000056          .WORD 46
      072576 065303          .WORD MSG46
      072600 000000          .WORD 0
3678 072602 000455          BR    WRTEXI
3679 072604          50$: MOV    RRGNXT,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
      072620 104456          TRAP  C$ERHRD
      072622 000057          .WORD 47
      072624 065366          .WORD MSG47
      072626 000000          .WORD 0
3684 072630 000433          BR    SETVAL        ; EXIT
3685 072632          60$: MOV    LOADD(R4),R1    ; GET XMITTED ADDRESS TABLE
3686 072632 016401 000004          MOV   WRDCNT(R4),R4  ; GET XMIT SIZE
3687 072636 016404 000006          NEG   R4           ; CONVERT FROM TWOS COMPLEMENT
3688 072642 005404          ASL  R4           ; TURN WORD COUNT TO BYTE COUNT
3689 072644 006304          ADD  #12,R4        ; COMPENSATE FOR GARBAGE BYTES
3690          ;; N.M. MOV   LOADD(R2),R3    ; GET RECEIVED PACKET
3691 072646 016203 000004          CMPB R4,STAT2(R2)  ; IS THE SIZE THE SAME?
3692 072652 120462 000012          BEQ  70$          ; BRANCH IF IT IS
3693 072656 001405          ERRHRD 48,MSG48    ; BYTE COUNT ERROR
3694 072660          104456          TRAP  C$ERHRD

```

```

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

70$: BR SETVAL ; EXIT
;; 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
ERRHRD 44,MSG44 ; COMPARE ERROR

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

90$: BR SETVAL ; LEAVE
DEC R4 ; DECREMENT THE BYTE COUNT
BNE 80$ ; 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 012704 023570      MOV      @TGTADR,R4      ; SET ROW POINTER TO DESTINATION TABLE
3786 073012 060204          ADD      R2,R4          ; ADD COLUMN TO IT
3787 073014 061604          ADD      (SP),R4        ; ADD SAVED GROUP TO IT
3788 073016          PUSH     R1          ; SAVE RETURN ADDRESS
3789 073020 010146          MOV      R4,R1          ; INIT LIMIT TO CURRENT ADDRESS
3790 073022 062701 000050    ADD      @40.,R1        ; UPPER LIMIT (5 ADRS BYTES * 8 BYTE OFFSET TO
3791      ; NEXT ROW)
3792 073026          10$:
3793 073026 112314          MOVB     (R3),.(R4)      ; STUFF THE ADDRESS BYTE IN
3794 073030 062704 000010    ADD      @8.,R4          ; ELSE, POINT TO NEXT ROW IN THE COLUMN
3795 073034 020401          CMP      R4,R1          ; HAVE ALL ADDRESSES BEEN WRITTEN
3796 073036 003001          BGT      20$          ; BRANCH IF YES
3797 073040 000772          BR       10$          ; AND STUFF IT IN
3798 073042          20$:
3799 073042 111314          MOVB     (R3),.(R4)      ; STASH LAST ADDRESS BYTE
3800 073044 112714 000000    MOVB     @0,(R4)        ; NEXT BYTE MBZ
3801 073050 062704 000010    ADD      @8.,R4          ; POINT TO NEXT BYTE
3802 073054 112714 000000    MOVB     @0,(R4)        ; NEXT BYTE MBZ
3803 073060          POP      R1          ; RESTORE RETURN ADDRESS
3804 073060 012601          MOV      (SP),.R1
3804 073062 000201          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

.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     @MNXM, 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
  
```

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

QNA INTERRUPT SERVICE ROUTINE

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	HIRCV(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
      073264 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$
3997 073550          CALL     BMPCNT R4,#BMPXMT
      073550 012725 000001
      073554 010425
      073556 004437 070656
      073562 003736
3998 073564          105$:
3999 073564 032764 040000 000010 BIT      #MXERRS,STAT1(R4)
4000 073572 001011          BNE      120$
4001 073574          110$:
4002 073574          CALL     GETXNX #0,#XRGCUR
      073574 012725 024220
      073600 012725 000000
      073604 004437 070656
      073610 006534
4003 073612 005003          CLR      R3
4004 073614 000437          BR       200$
4005 073616          120$:
4006 073616 032764 016000 000010 BIT      #MABORT!MLOSS!MNOCAR,STAT1(R4)
4007 073624 001763          BEQ     110$
4008 073626          140$:
4009 073626 005237 050274          INC     RETRYS
4010 073632 022737 000003 050274 CMP      #3,RETRYS
4011 073640 100001          BPL     150$
4012 073642 000754          BR       110$
4013 073644          150$:
4014 073644 000137 073370          JMP     50$
4015 073650          160$:
4016 073650          ERRDF  10,MSG10,ERR1
      073650 104455
      073652 000012
      073654 063670
      073656 070306
4017 073660 000413          BR       190$
4018 073662          170$:
4019 073662          ERRDF  12,MSG10,ERR1
      073662 104455
      073664 000014
      073666 063670
      073670 070306
4020 073672 000406          BR       190$
4021 073674          180$:
4022 073674 005237 050272          INC     TIMEOUT
4023 073700          ERRHRD  8,MSG08,ERR1
      073700 104456
      073702 000010
      073704 063605
      073706 070306

; BRANCH IF IT IS, WE DON'T COUNT SETUP STATISTICS
; ELSE, GO UPDATE THE COUNTERS
      MOV     #BMPXMT,(R5)
      MOV     R4,(R5)
      JSR     R4,PREG14
      .WORD  BMPCNT-ANCHOR

; SEE IF ANY ERRORS
; IF YES, BRANCH AND TAKE CARE OF THEM

; UPDATE "TRANSMIT RING CURRENT" POINTER
      MOV     #XRGCUR,(R5)
      MOV     #0,(R5)
      JSR     R4,PREG14
      .WORD  GETXNX-ANCHOR

; INDICATE SUCCESS
; RETURN

; WAS MESSAGE STILL SENT?
; BRANCH IF YES, DO NEXT ONE

; LET'S TRY IT AGAIN, KEEP COUNT OF THEM
; HAVE WE RETRIED THREE TIMES YET?
; IF NOT, THEN LET'S RETRANSMIT
; ELSE, CLEAN UP AND LEAVE

; TRY AGAIN

; TRANSMIT RING BOOKKEEPING ERROR
      TRAP   C$ERDF
      .WORD  10
      .WORD  MSG10
      .WORD  ERR1

; BOOKKEEPING ERROR, XRGNEXT SHOULD = XRGCUR
      TRAP   C$ERDF
      .WORD  12
      .WORD  MSG10
      .WORD  ERR1

; REPORT ERROR
      TRAP   C$ERHRD
      .WORD  8
      .WORD  MSG08
      .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$:
      MOV     @-1,R3
200$:
      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
4091 073774 032764 020000 000010 30$: BIT @MRSETP,STAT1(R4) ; WAS A SETUP PACKET RECEIVED?
4092 074002 001402 BEQ 32$ ; BRANCH IF NOT
4093 074004 000137 074560 JMP 150$ ; ELSE, LEAVE
4094 074010
4095 074010 32$: CALL BMPCNT R4,@BMPCV ; UPDATE COUNTERS FOR RECEIVE
      074010 012725 000000 MOV @BMPCV,(R5)+
      074014 010425 MOV R4,(R5)+
      074016 004437 070656 JSR R4,PREG14
      074022 003736 .WORD BMPCNT-ANCHOR
4096 074024 016403 000004 MOV LOADD(R4),R3 ; MOVE BUFFER ADDRESS INTO R3
4097 074030 005737 002200 TST BLDFLG ; ARE WE IN BUILD MODE?
4098 074034 001035 BNE 321$ ; 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$: BIT @MRUNT,R1 ;: N.M.
4107 074064 032701 004000 BNE 322$ ; RUNT PACKET
4108 074070 001037 ERRHRD 61,EMSG61 ; IF YES, THROW OUT THE PACKET.
4109 074072 TRAP C$ERHRD
      074072 104456 .WORD 61
      074074 000075 .WORD EMSG61
      074076 066412 .WORD 0
      074100 000000
4110 074102 000137 074544 JMP 130$ ; CONTINUE
4111
4112 074106 380$: BIT @MRUNT,R1 ;: N.M.
4113 074106 032701 004000 BNE 38$ ; RUNT PACKET
4114 074112 001451 ERRHRD 54,EMSG54 ; BRANCH IF NOT
4115 074114 TRAP C$ERHRD
      074114 104456 .WORD 54
      074116 000066 .WORD EMSG54
      074120 065773 .WORD 0
      074122 000000
4116 074124 000137 074544 JMP 130$ ; CONTINUE
4117
4118 074130 321$: CMP MCSTAD,(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4119 074130 023713 004032 BNE 35$ ; IF NOT, REPORT ERROR
4120 074134 001011 CMP MCSTAD+2,2(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4121 074136 023763 004034 000002 BNE 35$ ; IF NOT, REPORT ERROR
4122 074144 001005 CMP MCSTAD+4,4(R3) ; SEE IF CONSOLE ID MULTICAST IS DESTINATION
4123 074146 023763 004036 000004 BNE 35$ ; IF NOT, REPORT ERROR
4124 074154 001001 BR 38$ ; ITS A OKAY, MOVE ON
4125 074156 000427
4126 074160 35$: ERRHRD 68,EMSG68 ; ELSE, REPORT THE ERROR
4127 074160 TRAP C$ERHRD
      074160 104456 .WORD 68
      074162 000104 .WORD EMSG68
      074164 067121 .WORD 0
      074166 000000
4128
4129 074170 322$: MOV R4,R3 ; GET DESCRIPTOR ADDRESS
4130 074170 010403

```



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	RRGSRT,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.EMSG52				; ELSE REPORT ERROR
		074300	104456						TRAP	C\$ERHRD
		074302	000064						.WORD	52
		074304	065633						.WORD	EMSG52
		074306	000000						.WORD	0
4159	074310	005200			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!MOV!MRUNT,R1 ; WAS THE PACKET TOO LARGE
4177 074414 001005          BNE      80$ ; BRANCH IF NOT
4178 074416          ERRHRD  53,EMSG53 ; REPORT THE ERROR
      074416 104456          TRAP      C$ERHRD
      074420 000065          .WORD    53
      074422 065721          .WORD    EMSG53
      074424 000000          .WORD    0
4179 074426 000446          BR       130$ ; 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      #MOV!MRUNT,R1 ; OVERFLOW?
4185 074442 001405          BEQ     90$ ; BRANCH IF NOT
4186 074444          ERRHRD  55,EMSG55 ; REPORT OVERFLOW ERROR
      074444 104456          TRAP      C$ERHRD
      074446 000067          .WORD    55
      074450 066063          .WORD    EMSG55
      074452 000000          .WORD    0
4187 074454 000433          BR       130$ ; 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,EMSG56 ; ELSE REPORT IT
      074464 104456          TRAP      C$ERHRD
      074466 000070          .WORD    56
      074470 066116          .WORD    EMSG56
      074472 000000          .WORD    0
4192 074474          100$:
4193 074474 032701 000004          BIT      #MFRAM,R1 ; WAS FRAMING ERROR ALSO SET?
4194 074500 001405          BEQ     110$ ; BRANCH IF NOT
4195 074502          ERRHRD  57,EMSG57 ; REPORT THE ERROR
      074502 104456          TRAP      C$ERHRD
      074504 000071          .WORD    57
      074506 066152          .WORD    EMSG57
      074510 000000          .WORD    0
4196 074512 000414          BR       130$ ; 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,EMSG60 ; REPORT IT
      074522 104456          TRAP      C$ERHRD
      074524 000074          .WORD    60
      074526 066344          .WORD    EMSG60
      074530 000000          .WORD    0
4201 074532 000404          BR       130$
4202 074534          120$:
4203 074534          ERRHRD  61,EMSG61 ; REPORT UNKNOWN RECEIVER ERROR
      074534 104456          TRAP      C$ERHRD
      074536 000075          .WORD    61
      074540 066412          .WORD    EMSG61
      074542 000000          .WORD    0
4204 074544          130$:
4205 074544 005237 050276          INC      RCVERR ; ELSE, INCREMENT RECEIVE ERROR COUNTER
4206 074550 005200          INC      R0 ; SET STATUS TO FAILURE
4207 074552 000664          BR       50$ ; UPDATE POINTERS AND RETURN
4208 074554          140$:

```

```

4209 074554 005237 050300          INC    RCVBUF          ; INCREMENT GOOD BUFFERS RECEIVED COUNTER
4210 074560          150$:          INC    R2              ; KEEP COUNT OF HOW MANY BUFFERS RECEIVED
4211 074560 005202          DEC    NIRCNT          ; KEEP BOOKKEEPING IN ORDER
4212 074562 005337 050256          CALL   GETRNX #0,#RRGCR ; UPDATE "RECEIVE RING CURRENT" POINTER
4213 074566          MOV    #RRGCR,(R5)+
      074566 012725 024222          MOV    #0,(R5)+
      074572 012725 000000          JSR   R4,PREG14
      074576 004437 070656          .WORD GETRNX-ANCHOR
      074602 006450
4214 074604 000406          BR     170$
4215 074606          160$:          MOV    #2,R0          ; SET FAILURE STATUS IN R0
4216 074606 012700 000002          ERRDF 11,MSG11,ERR1 ; RECEIVE RING BOOKKEEPING ERROR
4217 074612          TRAP  C$ERDF
      074612 104455          .WORD 11
      074614 000013          .WORD MSG11
      074616 063735          .WORD ERR1
      074620 070306
4218 074622          170$:          CLR    DESC(R4)      ; INVALIDATE CURRENT DESCRIPTOR
4219 074622 005064 000002          180$:          RETURN R2           ; RETURN WITH NUMBER OF ENTRYS HANDLED
4220 074626
4221 074626          MOV    R2,(R5)+
      074626 010225          RTS    PC
      074630 000207
4222

```

.SBTTL BMPCNT - UPDATE STATISTICS

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

\*\*\*  
: FUNCTIONAL DESCRIPTION:  
: THIS ROUTINE IS USED TO UPDATE THE STATISTICS COUNTERS FROM THE  
: INFO IN STATUS WORDS 1 AND 2 IN BUFFER DESCRIPTOR 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

4272 074632  
4273 074632 012701 050144  
4274 074636 014502  
4275 074640 014503  
4276 074642 001504  
4277 074644 016204 000010  
4278 074650 032704 000400

---  
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
4292 074704 010446          MOV      R4,-(SP)
4293 074710 042704 177417  BIC      @177417,R4  ; CLEAR EVERYTHING BUT RETRY COUNT
4294 074712 001003          BNE      5$          ; BRANCH IF THE COUNT IS NOT ZERO
4295 074720 052761 000001 000070  BIS      @C.RTRY,C.XABB(R1) ; ELSE, SET RETRY FAILURE BIT IN BIT MASK
4296 074720          5$:
4297 074722 012604          POP      R4          ; RESTORE CONTENTS OF R4
4298 074722          10$:
4299 074726 032704 010000  BIT      @MLOSS,R4   ; WAS LOST CARRIER REASON FOR ABORT?
4300 074730 001406          BEQ      20$          ; BRANCH IF NOT
4301 074734 032704 002000  BIT      @MNOCAR,R4  ; TEST FOR NO CARRIER SENSED
4302 074736 001003          BNE      20$          ; BRANCH IF ZERO
4303 074744 052761 000002 000070  BIS      @C.LCAR,C.XABB(R1) ; SET CARRIER LOST BIT
4304 074744          20$:
4305 074750 032704 010000  BIT      @MLOSS,R4   ; WAS LOST CARRIER REASON FOR ABORT?
4306 074752 001006          BNE      30$          ; BRANCH IF NOT
4307 074756 032704 002000  BIT      @MNOCAR,R4  ; TEST FOR NO CARRIER SENSED
4308 074760 001403          BEQ      30$          ; BRANCH IF ZERO
4309 074766 032761 000004 000070  BIT      @C.SHRT,C.XABB(R1) ; YES, IT'S SET, SET SHORTED BIT
4310 074766          30$:
4311 074774 032761 000001 000070  BIT      @C.RTRY,C.XABB(R1) ; RETRY ABORT SET?
4312 074776 001407          BEQ      40$          ; BRANCH IF NOT
4313 075004 032761 000004 000070  BIT      @C.SHRT,C.XABB(R1) ; ELSE, IS THE SHORT BIT SET?
4314 075006 001003          BNE      40$          ; BRANCH IF NOT
4315 075014 052761 000010 000070  BIS      @C.OPEN,C.XABB(R1) ; ELSE, RETRY ABORT AND NOT A SHORT, SET OPEN BIT
4316 075014          40$:
4317 075016 000560          BR       BMPEXI      ; NO XMISSION OCCURRED, IGNORE BYTCNT
4318 075016          50$:
4319 075022 042704 177417  BIC      @177417,R4  ; CLEAR ALL BUT RETRY COUNT
4320 075024 001460          BEQ      BMPMUL      ; COUNTER IS ZERO, NO RETRIES OCCURRED
4321 075030 162704 000020  SUB      @20,R4      ; DECREMENT COUNT
4322 075032 001455          BEQ      BMPMUL      ; BRANCH IF ONLY ONE RETRY
4323 075036 162704 000020  SUB      @20,R4      ; ELSE, DECREMENT COUNT AGAIN
4324 075040 001003          BNE      60$          ; BRANCH IF MORE THAN TWO
4325 075044 005261 000050  INC      C.PXM2(R1)  ; ELSE, BUMP 2 RETRIES COUNT
4326 075046          BR       BMPMUL      ; FINISH COUNTERS
4327 075046          60$:
4328 075052 005261 000044  INC      C.PXM3(R1)  ; INCREMENT 3 OR MORE RETRIES COUNTER
4329 075054          BR       BMPMUL      ; FINISH COUNTERS
4330 075054 016204 000010  MOV      STAT1(R2),R4 ; GET STATUS WORD 1
4331 075060 032704 040000  BIT      @MRERRS,R4  ; TEST ERROR BIT
4332 075064 001001          BNE      5$          ; BRANCH IF ERROR BIT IS SET
4333 075066 000436          BR       BMPMUL
    
```

BMPCNT - UPDATE STATISTICS

```

4334 075070          5%:          BIT      @MDISC,R4          ; IS RUNT OR DISCARD SET?
4335 075070 032704 010000          BEQ      40%          ; BRANCH IF NOT
4336 075074 001424          10%:          INC      C.RERR(R1)          ; INCREMENT RECEIVE ERROR COUNT
4337 075076          BEQ      @MOVF,R4          ; OVERFLOW?
4338 075076 005261 000016          BEQ      20%          ; BRANCH IF NOT
4339 075102 032704 000001          INC      C.RLIN(R1)          ; SET INTERNAL BUFFER ERROR
4340 075106 001402          20%:          BIT      @MCRC,R4          ; CRC ERROR SET?
4341 075110 005261 000030          BEQ      35%          ; BRANCH IF NOT
4342 075114          BIS      @C.CRC,C.RERB(R1)          ; SET CRC ERROR BIT
4343 075114 032704 000002          30%:          BIT      @MFRAM,R4          ; FRAMING ERROR?
4344 075120 001411          BEQ      35%          ; BRANCH IF NOT
4345 075122 052761 000001 000014          BIS      @C.FRAM,C.RERB(R1)          ; ELSE, SET FRAMING ERROR BIT
4346 075130          BR      BMPEXI          ; EXIT
4347 075130 032704 000004          40%:          BIT      @MRUNT,R4          ; RUNT PACKET?
4348 075134 001403          BEQ      BMPMUL          ; BRANCH IF NOT
4349 075136 052761 000002 000014          BIS      @C.RUNT,C.RERB(R1)          ; SET RUNT PACKET BIT
4350 075144          BR      BMPEXI          ; EXIT
4351 075144 000504          5%:          P%PUSH  LOADD(R2)          ; SAVE ADDRESS OF DATA BUFFER
4352 075146          MOV      LOADD(R2),(R5)          ; XMIT OR RECEIVE?
4353 075146 032704 004000          TST      R3          ; BRANCH IF RECEIVE
4354 075152 001404          BEQ      5%          ; GET XMITTED WORD COUNT
4355 075154 052761 000010 000014          MOV      WRDCNT(R2),R4          ; COMPLEMENT IT
4356 075162 000475          NEG      R4          ; TURN INTO BYTE COUNT
4357 075164          ASL      R4          ; CONTINUE
4358 075164          BR      15%          ; EXTRACT RCV BYTE LENGTH 10:8
4359 075170 005703          BIC      @174377,R4          ; SAVE THE PACKET ADDRESS
4360 075172 001405          P%PUSH  R2          ; GET RCV BYTE LENGTH 7:0
4361 075174 016204 000006          MOV      STAT2(R2),R2          ; CLEAR COPY OF THE COUNT IN HIGH BYTE
4362 075200 005404          BIC      @C377,R2          ; MERGE THE TWO BYTE LENGTHS
4363 075202 006304          BIS      R2,R4          ; MAKE UP FOR BYTES LOST DURING ADDRESS CHECK
4364 075204 000413          ADD     @60.,R4          ; RESTORE THE PACKET ADDRESS
4365 075206          BR      15%          ; MOV      -(R5),R2
4366 075206 042704 174377          5%:          BIC      @174377,R4          ; SUBTRACT PROTOCOL BYTES. COUNT ONLY DATA BYTES
4367 075212          P%PUSH  R2          ; SEE IF XMIT OR RECEIVE
4368 075214 010225          MOV      STAT2(R2),R2          ; BRANCH IF RECEIVE
4369 075220 042702 177400          BIC      @C377,R2          ; INCREMENT NUMBER OF PACKETS XMITTED
4370 075224 050204          BIS      R2,R4          ; DID DATA START ON BYTE BOUNDARY?
4371 075226 062704 000074          ADD     @60.,R4          ; BRANCH IF NOT
4372 075232          P%POP   R2          ; DECREMENT COUNT BY ONE
4373 075234          MOV      -(R5),R2          ; SUBTRACT PROTOCOL BYTES. COUNT ONLY DATA BYTES
4374 075234 162704 000016          15%:          SUB     @16,R4          ; SEE IF XMIT OR RECEIVE
4375 075240 005703          TST      R3          ; BRANCH IF RECEIVE
4376 075242 001430          BEQ      50%          ; INCREMENT NUMBER OF PACKETS XMITTED
4377 075244 005261 000034          INC     C.PXMT(R1)          ; DID DATA START ON BYTE BOUNDARY?
4378 075250 032762 000100 000002          BIT     @MHIBYT,DESC(R2)          ; BRANCH IF NOT
4379 075256 001401          BEQ      10%          ; DECREMENT COUNT BY ONE
4380 075260 005304          DEC     R4          ; DID DATA START ON BYTE BOUNDARY?
4381 075262          10%:          BIT     @MLOBYT,DESC(R2)          ; BRANCH IF NOT
4382 075262 032762 000200 000002          BEQ     20%          ; DECREMENT COUNT BY ONE
4383 075270 001401          DEC     R4          ; GET DATA BUFFER ADDRESS
4384 075272 005304          20%:          P%POP   R2          ; MOV      (R5),R2
4385 075274          20%:          P%POP   R2          ; MOV      (R5),R2
4386 075274 014502          MOV     (R5),R2

```

```

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          40$:
4392 075316 060461 000060          ADD    R4,C.XDAT(R1)        ; ADD BYTE COUNT TO TOTAL DATA
4393 075322 000415          BR    BMPEXI                ; EXIT
4394 075324          50$:
4395 075324          P$POP  R2                    ; GET DATA BUFFER ADDRESS
      075324 014502          INC    C.PREC(R1)           ; BUMP PACKETS RECEIVED
4396 075326 005261 000004          BIT    #BIT00,DESTIN(R2)    ; MULTICAST ADDRESS?
4397 075332 032762 000001 000000          BEQ    60$                    ; BRANCH NOT SET
4398 075340 001404          INC    C.MREC(R1)           ; BUMP MULTICAST RECEIVE PACKET COUNT
4399 075342 005261 000010          ADD    R4,C.RMDB(R1)        ; BUMP MULTICAST RECEIVE BYTE COUNT
4400 075346 060461 000024          60$:
4401 075352          ADD    R4,C.RDAT(R1)      ; BUMP RECEIVED DATA COUNT
4402 075352 060461 000020          BMPEXI:
4403 075356          RETURN
4404 075356          ; ADIOS, AVOIR, AUWEIDERSEN
      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  
4442  
4443  
4444  
4445  
4446  
4447  
4448  
4449  
4450  
4451  
4452

\*\*\*  
: 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 FORMATTED 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

:  
: -  
: 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 REQD

075474 014537 075472  
075500 014504  
075502 014503  
075510 010325  
075512 012725 075360  
075516 013725 075472  
075522 004437 070656  
075526 004672

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

```

.SBTTL HXFORM - HEX FORMAT ROUTINE

```

4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484 075564 000000
4485 075566
      075566 014503
      075570 014502
      075572 014537 075564
4486
4487
4488
4489 075576 010204
4490 075600 063704 075564
4491
4492 075604 112301
4493 075606 142701 000200
4494
4495 075612 001414
4496 075614 120127 000060
4497 075620 100413
4498 075622 120127 000072
4499 075626 100406
4500 075630 120127 000101
4501 075634 100405
4502 075636 122701 000106
4503 075642 100402
4504
4505 075644 110122
4506 075646 001405
4507
4508
4509
4510 075650 020402
4511 075652 100354
4512 075654 012704 177777
4513 075660 000404
  
```

\*\*\*  
 : FUNCTIONAL DESCRIPTION:

```

: HXFORM HEX FORMAT ROUTINE
:
: THIS ROUTINE WILL ACCEPT A HEX STRING, AND STRIP OUT THE NON-HEX
: CHARACTERS (WITHOUT FLAGGING AN ERROR FOR THE NON-HEX CHARACTERS).
: A NULL WILL BE LEFT AS THE END OF STRING DELIMITER.
:
: INPUT ARGUMENTS P1- ADDRESS OF THE SOURCE ASCII STRING (NULL
: DELIMITER AT END OF STRING)
: P2- ADDRESS OF THE DESTINATION ASCII HEX STRING
: (STRIPPED OF NON-HEX AND RIGHT JUSTIFIED)
: P3- THE NUMBER OF HEX CHARACTERS DESIRED @R4
: P4- WILL CONTAIN THE ADDRESS OF THE LAST
: CHARACTER IN THE OUTPUT BUFFER.
: P5- THE SUCCESS/FAILURE (0/-1) INDICATOR
:
: IMPLICIT OUTPUTS THE BUFFER AT R4 WILL CONTAIN FORMATTED
: HEX (ASCII) STRING.
: CALLING PROCEDURE CALL HXFORM P1,P2,P3
: P$POP P4,P5
  
```

```

:---
HXN: .WORD
HXFORM: P$POP R3,R2,HXN
  
```

```

: ADDRESS OF SOURCE STRING
      MOV      -(R5),R3
      MOV      -(R5),P2
      MOV      -(R5),HXN
: ADDRESS OF DESTINATION STRING
: NUMBER OF HEX CHARACTERS DESIRED
: DESTINATION ADDRESS, R2: DESTINATION POINTER
: POINT TO END OF OUTPUT BUFFER (DESTINATION)
: DO WHILE NO NULL FOUND IN SOURCE STRING
: GET BYTE IN QUESTION (& POINT TO NEXT BYTE)
: ENSURE HI BIT IS LO (SEVEN BIT ASCII)
: CHECK FOR VALID HEX CHARACTER
: THE NULL IS A VALID CHARACTER (BUT THE LAST)
: IF GREATER THAN "0" THEN RANGE OK
: 40%-OUT OF RANGE, BYTE VALUE TOO SMALL
: IF BYTE IS LESS THAN 72 AND >=60 THEN RANGE OK
: RANGE OK IF >=60 AND <72 ELSE, CONTINUE CHECK
: BYTE MUST BE >=101 TO CONTINUE CHECK
: IF BYTE >71 AND <101 THEN BYTE OUT OF RANGE.
: BYTE MUST BE <=106 TO BE OK, ELSE: NOT OK.
: BYTE NOT OK, DON'T PLACE THIS BYTE IN OUTPUT.
: PLACE THE HEX BYTE IN THE OUTPUT BUFFER.
: BYTE IN RANGE, CONFIRMED. BYTE OK. POINT TO
: NEXT BYTE DEST ADDRESS. IF NULL, THEN EXIT.
: (NO ERRORS)
: IF NOT NULL, THEN CHECK FOR BUFFER OVERFLOW.
: R4 POINTS TO LAST CHARACTER POSITION (OUTPUT)
: R2 - PRESENT WRITE ADDRESS
: (SHOULD BE POSITIVE RESULT OR 0) (MORE TO DO)
: SET ERROR CONDITION (EXIT WITH ERROR)
: ERROR DETECTED EXIT PATH -> (TOO MANY CHARS)
  
```

```

      MOV      R2,R4
      ADD      HXN,R4
10$:  MOVB     (R3)+,R1
20$:  BICB     @200,R1
      BEQ     30$
      CMPB    R1,@60
      BMI     40$
      CMPB    R1,@72
      BMI     30$
      CMPB    R1,@101
      BMI     40$
      CMPB    @106,R1
      BMI     40$
30$:  MOVB     R1,(R2)+
      BEQ     60$
40$:  CMP      R4,R2
      BPL     10$
50$:  MOV      @-1,R4
      BR      70$
  
```

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 :
4526 :
4527 :
4528 :
4529 :
4530 :
4531 :
4532 :
4533 :
4534 :
4535 :
4536 :
4537 :
4538 :
4539 :
4540 :
4541 :
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
4546 075724 HEXBIN: .EVEN R1,R2,MN ;R1=SOURCE STRING ADDRESS
      075724 014501 MOV -(R5),R1
      075726 014502 MOV -(R5),R2
      075730 014537 075700 MOV -(R5),MN
4547 ;R2=DESTINATION STRING ADDRESS
4548 ;MN=NUMBER OF BYTES REQUIRED
4549 075734 060237 075700 ADD R2,MN ;MN NOW POINTS TO THE LAST_BYTE_POSITION+1
4550 075740 012704 075702 MOV @CMPSTR,R4 ;POINTER IN THE COMPARE STRING
4551 075744 121124 10%: CMPB (R1),(R4). ;COMPARE CURRENT CHAR WITH A CHAR IN CMPSTR
4552 075746 001376 20%: BNE 20% ;REPEAT UNTIL CHARACTER FOUND IN LIST
4553 075750 005201 INC R1 ;POINT TO THE NEXT ASCII BYTE
4554 075752 162704 075703 SUB @CMPSTR+1,R4 ;R4 NOW CONTAINS THE ACTUAL BINARY VALUE FOR
4555 ;THE NIBBLE DESCRIBED BY THE CURRENT BYTE.
4556 ;NOTE: NIBBLE IS THE HI PORTION OF THE BYTE
4557 075756 006304 ASL R4 ;MOVE NIBBLE TO THE HI END OF THE BYTE
4558 075760 006304 ASL R4
4559 075762 006304 ASL R4
4560 075764 006304 ASL R4
4561 075766 010403 MOV R4,R3 ;SAVE THE HI NIBBLE
4562 075770 012704 075702 MOV @CMPSTR,R4 ;POINTER INTO COMPARE STRING
4563 075774 121124 30%: CMPB (R1),(R4). ;COMPARE CURRENT CHAR WITH A CHAR IN CMPSTR
4564 075776 001376 BNE 30% ;REPEAT UNTIL MATCH FOUND IN CMPSTR LIST
4565 076000 005201 INC R1 ;POINT TO THE NEXT ASCII BYTE
4566 076002 162704 075703 SUB @CMPSTR+1,R4 ;R4 NOW CONTAINS THE ACTUAL BINARY VALUE FOR
4567 ;THE NIBBLE DESCRIBED BY THE CURRENT BYTE.
4568 ;NOTE: NIBBLE IS THE HI PORTION OF THE BYTE

```

```

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,MN
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 [MN] 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 :
4588 :              OUTPUTS - NONE
4589 :              IMPLICIT OUTPUTS      THE BUFFER AT P3 WILL CONTAIN THE HEX STRING
4590 :                                      FOLLOWED BY A NULL CHARACTER.
4591 :
4592 :              SUBORDINATE ROUTINES  NONE
4593 :              CALLING PROCEDURE     CALL BINHEX P1,P2,P3
4594 :
4595 :              HEXC:  .ASCII /0123456789ABCDEF/
4596 :
4597 :
4598 :
4599 :
4600 :
4601 :
4602 :
4603 :
4604 :
4605 :
4606 :
4607 :
4608 :
4609 :
4610 :
4611 :
4612 :
4613 :
4614 :
4615 :
076022 060 061 062
076025 063 064 065
076030 066 067 070
076033 071 101 102
076036 103 104 105
076041 106
076042 000000
076044 014501
076046 014537 076042
076052 014502
076054 060137 076042
076060 112103
076062 110304
076064 042703 177760
076070 006204
076072 006204
076074 006204
076076 006204
076100 042704 177760
076104 116422 076022
076110 116322 076022
076114 112722 000055
076120 020137 076042
076124 103755
076126 105042
076130
076130 000207

```

```

LST: .WORD
BINHEX: P$POP R1,LST,R2 ;R1 HAS THE INPUT BUFFER ADDRESS
MOV -(R5),R1
MOV -(R5),LST
MOV -(R5),R2
;LST: HAS THE NUMBER OF BYTES IN INPUT BUFFER
;R2 HAS THE OUTPUT BUFFER ADDRESS
;LST IS NOW ADDRESS OF LAST SOURCE BYTE + 1
;GET THE CURRENT BYTE AND POINT TO NEXT BYTE
;SEPARATE NIBBLES AND GET CHARACTERS SEPARATELY
;ONLY RIGHT BINARY NIBBLE REMAINS IN R3
;SHIFT OVER FOR LEFT BINARY NIBBLE IN R4
;ONLY LEFT BINARY NIBBLE REMAINS IN R4
;R4 IS THE MOST SIGNIFICANT NIBBLE (FIRST)
;R3 IS THE LEAST SIGNIFICANT NIBBLE (SECOND)
;PUT THE ASCII BYTE INTO THE BUFFER HI POSITION
;PUT THE ASCII BYTE INTO THE BUFFER LO POSITION
;PUT - BETWEEN HEX PAIRS
;RESULT IS NEGATIVE UNTIL R1=LST
;UNTIL R1=LST. (TRANSFER ALL SOURCE BYTES)
;TERMINATE OUTPUT BUFFER WITH A NULL

```

```

10$: ADD R1,LST
MOV (R1),R3
MOV R3,R4
BIC #177760,R3
ASR R4
ASR R4
ASR R4
ASR R4
BIC #177760,R4
MOV HEXC(R4),(R2)
MOV HEXC(R3),(R2)
MOV #'-',(R2)
CMP R1,LST
BLO 10$
CLRB -(R2)
RETURN

```

```

RTS PC

```

4617  
4618  
4619  
4620  
4621  
4622  
4623  
4624  
4625  
4626  
4627  
4628  
4629  
4630  
4631  
4632  
4633  
4634  
4635  
4636  
4637  
4638  
4639  
4640  
4641  
4642  
4643  
4644

.SBTTL BLDL - BUILD LOOP DIRECT DATA BUFFERS FOR TRANSMIT.

```

:--*
: FUNCTIONAL DESCRIPTION:
:   THIS SUBROUTINE BUILDS LOOP DIRECT PACKETS FOR TRANSMISSION
:   FROM THE QNA. SOURCE ADDRESS, DESTINATION ADDRESS,
:   PROT. TYPE, AND LOOP DIRECT HEADER INFO ARE ADDED
:   TO THE MESSAGE BUFFER. THE MESSAGE BUFFER IS BUILT
:   BY A CALL TO BLDBUF.
:
: INPUTS -
:   P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
:   IMPLICIT - P$SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
:             XRGXNT POINTS TO THE NEXT AVAILABLE RING ENTRY
:             PHYADR HOLDS THE CURRENT LOCAL QNA PHYSICAL ADDRESS
:
: OUTPUTS -
:   BUILT MESSAGE PACKET.
:
: CALLING PROCEDURE - CALL BLDL P1
:
: SIDE EFFECTS - THE MESSAGE PACKET IS BUILD AND CONTAINED IN THE
:                BUFFER POINTED TO BY XRGXNT WHEN THE ROUTINE WAS ENTERED.
:                XRGXNT IS UPDATED TO POINT TO THE NEXT RING ENTRY
:
: REGISTER USAGE - R1 HOLDS ADDRESS OF DESTINATION ADDRESS
:                  R2 IS A POINTER FOR THE LOOP DIRECT HEADER INFO
:                  R3 HOLDS THE PACKET LENGTH
:                  R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
:
:--*

```

```

4645 076132
4646 076132
4647 076132 014501
4648 076134 013704 024224
4649 076140 016404 000004
4650 076144 013703 002374
4651 076150 062703 000040
4652 076154 010337 050324
4653 076160 162737 000016 050324
4654 076166 022703 001100
4655 076172 002474
4656 076174 010337 050332
4657 076200 012164 000000
4658 076204 012164 000002
4659 076210 011164 000004
4660 076214 005064 000006
4661 076220 005064 000010
4662 076224 005064 000012
4663 076230 013764 050306 000014
4664 076236 012702 050346
4665 076242 012264 000016
4666 076246 011264 000020
4667 076252 013764 004010 000022
4668 076260 013764 004012 000024
4669 076266 013764 004014 000026
4670 076274 016264 000010 000030
4671 076302 013764 004010 000032
4672 076310 013764 004012 000034
4673 076316 013764 004014 000036

```

```

BLDL:
P$POP R1 ; PUT ADDRESS OF DEST. ADDRESS IN R1
MOV -(R5),R1
; MOVE NEXT PACKET ADDRESS TO R4
LOADD(R4),R4 ; POINT R4 TO DATA BLOCK
MOV P$SIZE,R3 ; PUT MESSAGE SIZE INTO R3
ADD #40,R3 ; ADD HEADER AND LOOP DIRECT INFO TO LENGTH
MOV R3,XFER ; PUT BYTES TRANSFERED INTO WORD
SUB #16,XFER ; AND CORRECT FOR HEADER
CMP #XPKLEN,R3 ; SEE IF LONGER THAN ONE PACKET
BLT 10$ ; IF YES, ERROR
MOV R3,BUFLEN ; PUT PACKET LENGTH IN BUFLEN
MOV (R1)+,DESTIN(R4) ; MOVE FIRST TWO BYTES OF ADDRESS
MOV (R1)+,DESTIN+2(R4) ; MOVE BYTES THREE AND FOUR
MOV (R1),DESTIN+4(R4) ; MOVE BYTES FIVE AND SIX
CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
CLR SOURCC+2(R4) ; SIX BYTES WORTH
CLR SOURCC+4(R4)
MOV PROTOO,PROTOT(R4) ; MOVE PROTOCALL TYPE INTO HEADER
MOV #LOPDIR,R2 ; MOVE LOOPDIRECT FORMAT HEADER LOC. TO R2
MOV (R2)+,LDSKIP(R4) ; SKIP COUNT
MOV (R2),LDFCT1(R4) ; FUNCTION CODE (FORWARD)
MOV PHYADR,LDADR1(R4) ; LOCAL NODE ADDRESS
MOV PHYADR+2,LDADR1+2(R4) ; SIX BYTES
MOV PHYADR+4,LDADR1+4(R4)
MOV 10(R2),LDFCT2(R4) ; FUNCTION CODE (REPLY)
MOV PHYADR,LDADR2(R4) ; LOCAL NODE ADDRESS
MOV PHYADR+2,LDADR2+2(R4) ; SIX BYTES
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$
4678 076364      10$:  ERRDF  14,MSG14,ERR1 ; EXIT
      076364 104455      ; MESSAGE SIZE TOO BIG
      076366 000016      TRAP  C$ERDF
      076370 063774      .WORD 14
      076372 070306      .WORD MSG14
      076374      .WORD ERR1
4679 076374      20$:  RETURN
      076374 000207      RTS   PC

```



4681  
 4682  
 4683  
 4684  
 4685  
 4686  
 4687  
 4688  
 4689  
 4690  
 4691  
 4692  
 4693  
 4694  
 4695  
 4696  
 4697  
 4698  
 4699  
 4700  
 4701  
 4702  
 4703  
 4704  
 4705  
 4706 076376  
 4707 076376  
       076376 014501  
       076400 014502  
 4708  
 4709 076402 013704 024224  
 4710 076406 016404 000004  
 4711 076412 013703 002374  
 4712 076416 062703 000060  
 4713 076422 010337 050324  
 4714 076426 162737 000016 050324  
 4715 076434 022703 001100  
 4716 076440 002524  
 4717 076442 010337 050332  
 4718 076446 011264 000000  
 4719 076452 016264 000002 000002  
 4720 076460 016264 000004 000004  
 4721 076466 005064 000006  
 4722 076472 005064 000010  
 4723 076476 005064 000012  
 4724 076502 013764 050306 000014  
 4725 076510 012764 000000 000016  
 4726 076516 012764 000002 000020  
 4727 076524 011164 000022  
 4728 076530 016164 000002 000024  
 4729 076536 016164 000004 000026  
 4730 076544 012764 000002 000030  
 4731 076552 011264 000032  
 4732 076556 016264 000002 000034  
 4733 076564 016264 000004 000036  
 4734 076572 012764 000002 000040  
 4735 076600 013764 004010 000042

```

.SBTTL          BLDFAS - BUILD PACKET FOR FULL ASSIST TRANSMISSION
:--*
: FUNCTIONAL DESCRIPTION:
:   THIS SUBROUTINE BUILDS FULL ASSIST PACKETS FOR TRANSMISSION
:   FROM THE QNA.  SOURCE ADDRESS, DESTINATION ADDRESS, PROT. TYPE
:   AND FULL ASSIST HEADER INFO ARE ADDED TO THE MESSAGE BUFFER.
:   THE MESSAGE BUFFER IS BUILT BY A CALL TO BLDBUF.
:
: INPUTS -      P1 - THE ADDRESS OF THE DESTINATION ADDRESS (FROM NODE TABLE)
:               IMPLICIT - P$SIZE CONTAINS THE SIZE OF THE MESSAGE BUFFER DATA
:               XRG NXT POINTS TO THE NEXT AVAILABLE RING ENTRY
:               PHYADR HOLDS THE CURRENT LOCAL NODE ADDRESS
:
: OUTPUTS -     THE BUILT BUFFER
:
: CALLING PROCEDURE - CALL BLDFAS P1
:
: SIDE EFFECTS - XRG NXT SI UPDATED TO POINT TO THE NEXT RING ENTRY
:
: REGISTER USAGE - R1 HOLDS ADDRESS OF TARGET NODE ADDRESS
:                  R2 HOLDS ADDRESS OF ASSIST NODE ADDRESS
:                  R3 HOLDS THE PACKET LENGTH
:                  R4 HOLDS ADDRESS OF NEXT RING ENTRY DATA BUFFER
:--*
BLDFAS::
      P$POP      R1,R2          ; PUT ADDRESS OF TARGET ADDRESS INTO R1
                          MOV      -(R5),R1
                          MOV      -(R5),R2
:
:   AND ADDRESS OF ASSIST ADDRESS INTO R2
:   MOVE NEXT PACKET ADDRESS TO R4
:   POINT R4 TO DATA BLOCK
:   PUT MESSAGE SIZE INTO R3
:   ADD HEADER INFO TO LENGTH
:   PUT 'BYTES TRANSFERED' INTO WORD
:   AND CORRECT FOR HEADER
:   SEE IF LONGER THAN ONE PACKET
:   IF YES, ERROR
:   PUT PACKET LENGTH IN BUFLN
:   MOVE FIRST TWO BYTES OF ADDRESS
:   MOVE BYTES THREE AND FOUR
:   MOVE BYTES FIVE AND SIX
:   LEAVE BLANK SPACE FOR SOURCE ADDRESS
:   SIX BYTES WORTH
:
: MOVE PROTOCOL TYPE INTO HEADER
: SKIP COUNT
: FUNCTION CODE (FORWARD)
: TARGET NODE ADDRESS
: SIX BYTES
:
: FUNCTION CODE (FORWARD)
: ASSIST NODE ADDRESS
: SIX BYTES
:
: FUNCTION CODE (FORWARD)
: 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
                                MOV     R4,(R5)+
                                JSR     R4,PREG14
                                .WORD  BLDBUF-ANCHOR
4745 076672              CALL    GETXNX #0,#XRGNXT          ; UPDATE POINTER TO NEXT RING ENTRY
                                MOV     #XRGNXT,(R5)+
                                MOV     #0,(R5)+
                                JSR     R4,PREG14
                                .WORD  GETXNX-ANCHOR
4746 076710 000404              BR      20$
4747 076712 104455 10$:  ERRDF  29,MSG14,ERR1            ; EXIT
                                ; MESSAGE SIZE TOO BIG
                                TRAP   C$ERDF
                                .WORD  29
                                .WORD  MSG14
                                .WORD  ERR1
4748 076722 000207 20$:  RETURN
                                RTS    PC
  
```

```

4750 .SBTTL BLDAST - BUILD TRANSMIT AND RECEIVE ASSIST PACKETS
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
076724 014501 MOV -(R5),R1
076726 014502 MOV -(R5),R2
4758 ; ASSIST ADDRESS INTO R2
4759 076730 013704 024224 MOV XRGXNT,R4 ; MOVE NEXT PACKET ADDRESS TO R4
4760 076734 016404 000004 MOV LOADD(R4),R4 ; POINT R4 TO DATA BLOCK
4761 076740 013703 002374 MOV P%SIZE,R3 ; PUT MESSAGE SIZE INTO R3
4762 076744 062703 000050 ADD #50,R3 ; ADD HEADER INFO INTO LENGTH
4763 076750 010337 050324 MOV R3,XFER ; PUT 'BYTES TRANSFERED' INTO WORD
4764 076754 162737 000016 050324 SUB #16,XFER ; AND CORRECT FOR HEADER
4765 076762 022703 001100 CMP #XPKLEN,R3 ; SEE IF LONGER THAN ONE PACKET
4766 076766 002511 BLT 10# ; IF YES, ERROR
4767 076770 010337 050332 MOV R3,BUFLEN ; PUT PACKET LENGTH INTO BUFLEN
4768 076774 011164 000000 MOV (R1),DESTIN(R4) ; MOVE DESTINATION ADDRESS INTO HEADER
4769 077000 016164 000002 000002 MOV 2(R1),DESTIN+2(R4) ; SIX BYTES WORTH
4770 077006 016164 000004 000004 MOV 4(R1),DESTIN+4(R4)
4771 077014 005064 000006 CLR SOURCC(R4) ; LEAVE BLANK SPACE FOR SOURCE ADDRESS
4772 077020 005064 000010 CLR SOURCC+2(R4) ; SIX BYTES WORTH
4773 077024 005064 000012 CLR SOURCC+4(R4)
4774 077030 013764 050306 000014 MOV PROTOO,PROTOT(R4) ; MOVE PROTOCOL TYPE INTO HEADER
4775 077036 012764 000000 000016 MOV #0,FASKIP(R4) ; SKIP COUNT
4776 077044 012764 000002 000020 MOV #2,FAFCT1(R4) ; FUNCTION CODE (FORWARD)
4777 077052 011264 000022 MOV (R2),FAADR1(R4) ; TARGET NODE ADDRESS
4778 077056 016264 000002 000024 MOV 2(R2),FAADR1+2(R4) ; SIX BYTES
4779 077064 016264 000004 000026 MOV 4(R2),FAADR1+4(R4)
4780 077072 012764 000002 000030 MOV #2,FAFCT2(R4) ; FUNCTION CODE (FORWARD)
4781 077100 013764 004010 000032 MOV PHYADR,FAADR2(R4) ; LOCAL NODE ADDRESS
4782 077106 013764 004012 000034 MOV PHYADR+2,FAADR2+2(R4) ; SIX BYTES WORTH
4783 077114 013764 004014 000036 MOV PHYADR+4,FAADR2+4(R4)
4784 077122 012764 000001 000040 MOV #1,FAFCT3(R4) ; FUNCTION CODE (REPLY)
4785 077130 013764 004010 000042 MOV PHYADR,FAADR3(R4) ; LOCAL NODE ADDRESS
4786 077136 013764 004012 000044 MOV PHYADR+2,FAADR3+2(R4)
4787 077144 013764 004014 000046 MOV PHYADR+4,FAADR3+4(R4)
4788 077152 062704 000050 ADD #FAADR3+6,R4 ; POINT R4 TO FIRST DATA BYTE
4789 077156 010437 050334 MOV R4,CMPBUF ; STORE BUFFER LOCATION FOR DATA COMPARE
4790 077162 CALL BLDBUF R4 ; BUILD DATA BUFFER
077162 010425 MOV R4,(R5)+
077164 004437 070656 JSR R4,PREG14
077170 006756 .WORD BLDBUF-ANCHOR

```

```

4791 077172          CALL  GETXNX  #0,#XRG NXT      ; UPDATE RING POINTER
      077172 012725 024224          MOV      #XRG NXT,(R5).
      077176 012725 000000          MOV      #0,(R5).
      077202 004437 070656          JSR      R4,PREG14
      077206 006534          .WORD  GETXNX-ANCHOR
4792 077210 000404          BR      20$
4793 077212          10$:  ERRDF  36,EMSG14,ERR1      ; MESSAGE SIZE TOO BIG ERROR
      077212 104455          TRAP   C$ERDF
      077214 000044          .WORD  36
      077216 063774          .WORD  EMSG14
      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 013704 024224  
077224 016404 000004  
077234 012737 000100 050332  
077242 012701 002316  
077246 012164 000000  
077252 012164 000002  
077256 011164 000004  
077262 005064 000006  
077266 005064 000010  
077272 005064 000012  
077276 013764 050310 000014  
077304 012702 050340  
077310 012264 000016  
077314 012264 000020  
077320 011264 000022  
077324 012725 024224  
077330 012725 000000  
077334 004437 070656  
077340 006534  
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
MOV LOADD(R4),R4 ; POINT R4 TO DATA BLOCK
MOV #100,BUFLEN ; MOVE BUFFER SIZE TO BUFLEN
MOV #ADRFUF,R1 ; MOVE ADDRESS OF DECT. 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 : FUNCTIONAL DESCRIPTION
4841 : THIS SUBROUTINE GETS THE NEXT TRANSMIT OR RECEIVE RING
4842 : ENTRY. IT IS ENTERED AT SEPERATE POINTS DEPENDING ON
4843 : WHICH RING IS BEING USED.
4844 :
4845 : INPUTS - P1 - FLAG INDICATING IF THE PREVIOUS DESCRIPTOR NEEDS REVALIDATING
4846 : P2 - THE ADDRESS OF THE RING POINTER TO BE UPDATED.
4847 :
4848 : OUTPUTS - THE RING POINTER IS UPDATED TO POINT TO THE NEXT AVAILABLE
4849 : ENTRY.
4850 :
4851 : CALLING PROCEDURE - CALL GETXNX @P1,@P2 ; FOR TRANSMIT UPDATES
4852 : CALL GETRXN @P1,@P2 ; FOR RECEIVE UPDATES
4853 :
4854 : SIDE EFFECTS - NONE
4855 :
4856 : REGISTER USAGE - R1 POINTS TO THE FIRST ENTRY IN THE RING
4857 : R2 POINTS TO THE LAST ENTRY IN THE RING
4858 : R3 IS THE ADDRESS OF THE RING POINTER TO BE UPDATED
4859 :
4860 :---*
4861 077344 GETRXN::
4862 077344 013701 024216 MOV RRGSR, R1 ; MOVE FIRST RING ENTRY TO R1
4863 077350 013702 024232 MOV RRGLST, R2 ; MOVE LAST RING ENTRY TO R2
4864 077354 P$POP R3 ; SHOULD WE VALIDATE PREVIOUS DESCRIPTOR
4865 077356 014503 BEQ GETCOM ; BRANCH IF NOT
4866 077360 001431
4867 077360 013703 024234 RNXPRV:: MOV RRGPRV, R3 ; ELSE, GET PREVIOUS DESCRIPTOR
4868 077364 005063 000000 CLR FLAG(R3) ; REINIT THE FLAG
4869 077370 005063 000010 CLR STAT1(R3) ; REINIT THE STATUS
4870 077374 005063 000012 CLR STAT2(R3) ; REINIT STATUS 2
4871 077400 012763 100000 000002 MOV #MVALID, DESC(R3) ; VALIDATE IT
4872 077406 020203 CMP R2, R3 ; IS THIS THE LAST DATA DESCRIPTOR?
4873 077410 001404 BEQ 10$ ; BRANCH IF YES
4874 077412 062737 000014 024234 ADD #12., RRGPRV ; ELSE UPDATE PREVIOUS POINTER
4875 077420 000410 BR GETCOM ; AND DO COMMON STUFF
4876 077422
4877 077422 010137 024234 10$: MOV R1, RRGPRV ; LAST ENTRY, POINT TO START OF LIST
4878 077426 000405 BR GETCOM ; GO TO COMMON MODE
4879 077430
4880 077430 GETXNX:: P$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 XRGLST, R2 ; MOVE LAST RING ENTRY TO R2
4884 077442 P$POP R3 ; GET ADDRESS OF RING POINTER IN R3
4885 077442 014503 MOV -(R5), R3
4886 077444 021302 CMP (R3), R2 ; SEE IF POINTER POINTS TO LAST RING
4887 077450 062713 000014 BEQ 10$ ; IF YES, BRANCH
4888 077454 020327 024226 ADD #12., (R3) ; ELSE, ADD ENTRY LENGTH TO POINTER
4889 077460 001073 CMP R3, @RRGNXT ; IS THIS THE NEXT RECEIVE POINTER?
4890 077462 012704 024226 BNE 20$ ; BRANCH IF NOT
4891 077466 062704 000014 MOV @RRGNXT, R4 ; ELSE GET NEXT RING DESCRIPTOR
ADD #14, R4 ; POINT TO NEXT DESCRIPTOR
    
```

```

4892 077472 005764 000002          TST    DESC(R4)
4893 077476 100407                BMI    5$
4894 077500 001406                BEQ    5$
4895 077502 062737 000014 024226  ADD    @14,RRGNXT
4896 077510 012725 024226          P$PUSH @RRGNXT
                                MOV    @RRGNXT,(R5)
4897 077514 000721                BR     RNXPV
                                ; AND UPDATE RECEIVE RING POINTERS AGAIN
4898 077516 023737 024222 024226 5$:  CMP    RRGCUR,RRGNXT
4899 077524 002051                BGE    20$
4900 077526 005737 050336          TST    LAPCNT
4901 077532 001046                BNE    20$
4902 077534 104456                ERRHRD 63,MSG63
                                ; DID NEXT POINTER PASS BY CURRENT POINTER?
                                ; IF NO, THEN EXIT
                                ; ELSE, IS CURRENT POINTER A LAP(S) AHEAD?
                                ; IF YES, THEN EXIT
                                ; ELSE, REPORT ERROR
                                TRAP   C$ERHRD
                                .WORD 63
                                .WORD MSG63
                                .WORD 0
4903 077534 000441                BR     20$
                                ; EXIT
4904 077544 022713 024226          CMP    @RRGNXT,(R3)
4905 077546 001007                BNE    12$
4906 077546 005337 050336          DEC    LAPCNT
4907 077552 100004                BPL    12$
4908 077562 104456                ERRHRD 63,MSG63
                                ; IS THIS THE "NEXT RECEIVE DESCR" POINTER?
                                ; BRANCH IF NOT
                                ; ELSE, DECREMENT THE LAPCOUNT
                                ; BRANCH IF "CURRENT RCV DESCR" POINTER NOT PASSED
                                ; PRINT "NEXT POINTER PASSED BY CURRENT PONTER"
                                TRAP   C$ERHRD
                                .WORD 63
                                .WORD MSG63
                                .WORD 0
4909 077562 022713 024222          CMP    @RRGCUR,(R3)
4910 077564 001002                BNE    18$
4911 077566 005237 050336          INC    LAPCNT
                                ; IS THIS THE "CURRENT RCV DESC" POINTER?
                                ; BRANCH IF NOT
                                ; ELSE, INCREMENT THE LAP COUNTER
4912 077572 010113                MOV    R1,(R3)
4913 077572 022703 024226          CMP    @RRGNXT,R3
4914 077576 001016                BNE    20$
4915 077600 012704 024226          MOV    @RRGNXT,R4
4916 077604 062704 000014          ADD    @14,R4
4917 077606 005764 000002          TST    DESC(R4)
4918 077606 100407                BMI    20$
4919 077612 001406                BEQ    20$
4920 077614 062737 000014 024226  ADD    @14,RRGNXT
4921 077620 005764 000002          P$PUSH @RRGNXT
4922 077624 012725 024226          MOV    @RRGNXT,(R5)
4923 077630 100407                BR     RNXPV
                                ; GO UPDATE POINTERS AGAIN
4924 077632 001406                BR     RNXPV
4925 077634 062737 000014 024226  BR     RNXPV
4926 077642 012725 024226          BR     RNXPV
4927 077646 000644                BR     RNXPV
4928 077650 000207                RETURN
                                RTS    PC

```

4930  
4931  
4932  
4933  
4934  
4935  
4936  
4937  
4938  
4939  
4940  
4941  
4942  
4943  
4944  
4945  
4946  
4947  
4948  
4949  
4950  
4951  
4952  
4953  
4954 077652  
4955 077652  
4956 077654 014503  
4957 077660 013702 002372  
4958 077662 006302  
4959 077662 013704 002374  
4960 077670 016201 003360  
4961 077674 005037 050304  
4962 077700 005237 050304  
4963 077704 112123  
4964 077706 026237 003342 050304  
4965 077714 001004  
4966 077716 016201 003360  
4967 077722 005037 050304  
4968 077726 020304  
4969 077730 001363  
4970 077732  
077732 000207

```

.SBTTL          BLDBUF - BUILD MESSAGE BUFFERS
:--*
: FUNCTIONAL DESCRIPTION
: THIS SUBROUTINE CREATES A MESSAGE BUFFER TO BE USED
: FOR TRANSMISSION.
:
: INPUTS -      P1 - BUFFER LOCATION TO PUT BUILT MESSAGE
: IMPLICIT - P$SIZE CONTAINS THE SIZE THE BUFFER IS TO BE
:             P$TYPE CONTAINS THE MESSAGE TYPE
:
: OUTPUTS -     IMPLICIT - BUFFER STARTING AT LOCATION P1 CONTAINS A
: MESSAGE P$SIZE BYTES LONG USING THE MESSAGE
: TYPE SPECIFIED BY P$TYPE.
:
: CALLING PROCEEDURE - CALL BLDBUF P1
:
: SIDE EFFECTS - NONE
:
: REGISTER USAGE - R1 POINTS TO THE NEXT BYTE OF STORED MESSAGE TO BE BUILT
:                   IN THE MESSAGE BUFFER.
:                   R2 = (MESSAGE TYPE X 2), USED AS OFFSET FOR POINTERS
:                   R3 POINTS TO THE NEXT BYTE OF THE BUFFER UNDER CONSTRUCTION
:                   R4 POINTS TO THE LAST BYTE OF THE BUFFER UNDER CONSTRUCTION
:--*
BLDBUF::
      P$POP      R3          ; PUT BUFFER ADDRESS INTO R3
                        MOV      -(R5),R3
      MOV        P$TYPE,R2  ; PUT MESSAGE TYPE INTO R2
      ASL        R2          ; MULTIPLY BY 2
      MOV        P$SIZE,R4  ; PUT SIZE INTO R4
      ADD        R3,R4      ; MAKE R4 = LAST BYTE OF BUFFER
      MOV        MSGAD(R2),R1 ; POINT R1 TO FIRST BYTE OF STORED MESSAGE
      CLR        COUNT      ; CLEAR BYTE COUNTER
10$:  INC        COUNT      ; COUNT NO. OF BYTES COPIED
      MOVB       (R1)+,(R3)+ ; PUT BYTE IN BUFFER
      CMP        MSGCNT(R2),COUNT ; ARE WE AT END OF STORED MESSAGE
      BNE        20$        ; IF NO, CHECK IF DONE
      MOV        MSGAD(R2),R1 ; ELSE, POINT R1 TO BEGINING
      CLR        COUNT      ; AND CLEAR COUNTER
20$:  CMP        R3,R4      ; IS BUFFER FILLED?
      BNE        10$        ; IF NO, LOOP
      RETURN              ; ELSE, RETURN
                        RTS          PC

```



4972  
4973  
4974  
4975  
4976  
4977  
4978  
4979  
4980  
4981  
4982  
4983  
4984  
4985  
4986  
4987  
4988  
4989  
4990  
4991  
4992  
4993  
4994  
4995  
4996  
4997  
4998  
4999  
5000  
5001  
5002  
5003  
5004  
5005  
5006  
5007  
5008  
5009  
5010  
5011  
5012  
5013  
5014  
5015  
5016

077734  
077734  
077734  
077736  
077740  
  
077742  
077744  
077750  
077752  
077754  
077756  
077762  
077770  
077772  
077772  
077774  
077776  
100000  
100004  
100010  
100012  
100014  
100020  
100022  
100024  
100026  
100030  
100036  
100040  
100040  
013746

014501  
014502  
014503  
  
005004  
005037  
005204  
121213  
001421  
005237  
002737  
002413  
  
011346  
011246  
010446  
067602  
000004  
010600  
104415  
062706  
000012  
  
005202  
005203  
005301  
001350  
002737  
001412  
050314  
050314

```
.SBTTL          DATCMP - COMPARE DATA BUFFERS
:---*
: FUNCTIONAL DESCRIPTION
:           THIS SUBROUTINE COMPARES TWO DATA BUFFERS BYTE BY BYTE.
:           IF COMPARISON ERRORS OCCURED, LOCATION, EXPECTED DATA
:           AND RECEIVED DATA ARE PRINTED OUT FOR THE FIRST FIVE
:           ERRORS.  THE TOTAL NUMBER OF ERRORS IS ALSO PRINTED.
:
: INPUTS -   P1 - THE SIZE (IN BYTES) OF THE BUFFER TO BE COMPARED.
:           P2 - THE ADDRESS OF BUFFER TO COMPARE OTHER BUFFER AGAINST.
:           P3 - THE ADDRESS OF THE SECOND BUFFER.
:
: OUTPUTS -  P4 - THE NUMBER OF COMPARISON ERRORS.
:
: CALLING PROCEDURE - CALL DATCMP P1,P2,P3
:                   P$POP P4
:
: SIDE EFFECTS - NONE.
:
: REGISTER USAGE - R1 CONTAINS THE COMPARE SIZE
:                   R2 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 1
:                   R3 CONTAINS THE ADDRESS OF THE BYTE BEING COMPARED IN BUFFER 2
:                   R4 CONTAINS THE BYTE OFFSET (BYTES FROM BEGINING OF BUFFER
:---*
```

```
DATCMP::
P$POP      R1,R2,R3          ; PUT COMPARE SIZE IN R1
                                MOV      -(R5),R1
                                MOV      -(R5),R2
                                MOV      -(R5),R3
                                ; BUFFER 1 ADDRESS IN R2 AND
                                ; BUFFER 2 ADDRESS IN R3
                                ; INITIALIZE BYTE OFFSET
                                ; AND ERROR COUNTER
10$:      CLR      R4          ; INCREMENT OFFSET COUNTER
                                INC      R4          ; INCREMENT OFFSET COUNTER
                                CMPB    (R2),(R3)   ; COMPARE BUFFERS
                                BEQ      20$         ; IF SAME, BRANCH
                                INC      TEMP        ; INCREMENT ERROR COUNTER
                                CMP      #5,TEMP    ; IF MORE THAN 5 ERRORS,
                                BLT      20$         ; DON'T PRINT MESSAGE
                                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$PNTX
                                ADD     #12,SP
20$:      INC      R2          ; INCREMENT BUFFER 1 POINTER
                                INC      R3          ; INCREMENT BUFFER 2 POINTER
                                DEC      R1          ; DECREMENT COMPARE SIZE
                                BNE     10$         ; IF NOT FINISHED, GO BACK FOR MORE
                                CMP      #0,TEMP    ; WERE THERE ANY ERRORS?
                                BEQ     30$         ; IF NO, EXIT
                                PRINTX  @CMPE2,TEMP
                                MOV      TEMP, -(SP)
```

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

30\$: RETURN TEMP

```

; RETURN WITH ERROR COUNT ON STACK
MOV    TEMP,(R5).
RTS    PC

```

```

MOV    #CMPER2,-(SP)
MOV    #2,-(SP)
MOV    SP,R0
TRAP   C$PNTX
ADD    #6,SP

```

5019  
5020  
5021  
5022  
5023  
5024  
5025  
5026  
5027  
5028  
5029  
5030  
5031  
5032  
5033  
5034  
5035  
5036  
5037  
5038  
5039  
5040  
5041  
5042  
5043  
5044  
5045  
5046  
5047  
5048  
5049  
5050  
5051  
5052  
5053  
5054  
5055  
5056  
5057  
5058  
5059  
5060  
5061  
5062  
5063  
5064  
5065  
5066

100072  
100072 014537 050314  
100076 023727 050314 000001  
100104 001002  
100106 014502  
100110 000402  
100112 014502  
100114 014504  
100116 012701 002714  
100122 005711  
100124 001415  
100126 021127 177777  
100132 001454  
100134 010225  
100136 010125  
100140 004437 070656  
100144 026004  
100146 014503  
100150 001412  
100152 062701 000026  
100156 000761  
100160 011211  
100162 016261 000002 000002  
100170 016261 000004 000004

```
.SBTTL          WRITES - WRITE DATA ONTO SUMMARY TABLE
:--*
: FUNCTIONAL DESCRIPTION:
:   THIS SUBROUTINE UPDATES THE SUMMARY TABLE DATA FOR
:   THE NODES SPECIFIED IN THE CALL STATEMENT.  EITHER ONE
:   OR TWO NODES CAN BE UPDATED PER CALL.  AFTER THE CALL,
:   THE SUMMARY DATA COUNTERS ARE CLEARED.  THE SUMMARY TABLE
:   IS CHECKED FOR A MATCHING NODE ADDRESS AND UPDATES THE
:   DATE FOR THAT NODE, OR ADDS THE NODE TO THE TABLE IF IT
:   DOESN'T EXIST.  AN ERROR IS REPORTED IF THE END OF THE TABLE
:   IS REACHED.
:
: INPUTS -      P1 - THE NUMBER OF NODES TO UPDATE (1 OR 2).
:               P2 - THE ADDRESS OF THE FIRST NODE ADDRESS.
:               P3 - THE ADDRESS OF THE SECOND NODE ADDRESS IF P1 = 2 OR
:                   BLANK IF P1 = 1.
:
: OUTPUTS -     THE SUMMARY TABLE IS UPDATED.
:
: CALLING PROCEDURE - CALL WRITES P1,P2(,P3)
:
: SIDE EFFECTS - THE SUMMARY COUNTERS ARE CLEARED.
:
: REGISTER USAGE - R1 POINTS TO THE CURRENT LOCATION IN THE SUMMARY TABLE.
:                  R2 POINTS TO THE NODE TO BE UPDATED'S ADDRESS.
:                  R3 IS SCRATCH
:                  R4 HOLDS THE SECOND NODE TO BE UPDATED ADDRESS.
:--*
WRITES: P$POP   TEMP          ; SEE HOW MANY NODES TO WRITE
                                MOV      -(R5),TEMP
                                ; IF ONLY ONE, GET ADDRESS
                                CMP      TEMP,#1
                                BNE     10$
                                P$POP   R2
                                MOV      -(R5),R2
                                BR 20$
10$:  P$POP     R2,R4          ; IF TWO, GET BOTH ADDRESSES
                                MOV      -(R5),R2
                                MOV      -(R5),R4
20$:  MOV      #STATBL,R1     ; MOVE STATISTICAL TABLE ADDRESS INTO R1
30$:  TST      (R1)           ; SEE IF SLOT IS EMPTY
                                BEQ     40$
                                ; IF YES, BR
                                CMP      (R1),#-1
                                BEQ     80$
                                ; IF YES, ERROR
                                CALL    CMPADR R1,R2
                                ; LOOK FOR MATCHING ADDRESS
                                MOV      R2,(R5)+
                                MOV      R1,(R5)+
                                JSR      R4,PREG14
                                .WORD   CMPADR-ANCHOR
                                P$POP   R3
                                MOV      -(R5),R3
                                BEQ     50$
                                ; IF YES, BR
                                ADD     #26,R1
                                ; ELSE, POINT R1 TO NEXT ENTRY
                                BR      30$
                                ; AND CHECK AGAIN
40$:  MOV      (R2),(R1)      ; ADD NEW ADDRESS TO TABLE
                                MOV     2(R2),2(R1)
                                ; SIX BYTES WORTH
                                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              80$:  PRINTF  #TABFUL,#SUMM ; PRINT TABLE FULL MESSAGE
      100264 012746 053422              MOV    #SUMM,-(SP)
      100270 012746 053274              MOV    #TABFUL,-(SP)
      100274 012746 000002              MOV    #2,-(SP)
      100300 010600              MOV    SP,R0
      100302 104417              TRAP  C$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              RETURN
      100340 000207

```

RTS PC

5093  
5094  
5095  
5096  
5097  
5098  
5099  
5100  
5101  
5102  
5103  
5104  
5105  
5106  
5107  
5108  
5109  
5110  
5111  
5112

```
.SBTTL          BINDEC  CONVERT A 32 BIT BINARY NUMBER TO DECIMAL
:---+
: FUNCTIONAL DESCRIPTION:
: THIS SUBROUTINE CONVERTS A 32 BIT BINARY NUMBER TO
: A DECIMAL NUMBER REPRESENTED AS AN ASCIZ STRING.
:
: INPUTS -      P1 - THE ADDRESS OF THE FIRST WORD OF BINARY DATA
:               BITS 0-15.  THE SECOND WORD, BITS 16-31, IS
:               EXPECTED TO IMMEDIATELY FOLLOW THE FIRST WORD.
:
: OUTPUTS -     THE ASCII STRING WILL BE LOCATED STARTING AT DECSTR
:
: SIDE EFFECTS - NONE
:
: REGISTER USAGE - R1 POINTS TO BITS 0-15 OF BINARY DATA
:                  R2 POINTS TO BITS 16-31 OF BINARY DATA
:                  R3 POINTS TO THE OUTPUT STRING
:                  R4 POINTS TO THE POWERS OF 10 TABLE
:---+
BINDEC::
```

5113 100342  
5114 100342  
5115 100344  
5116 100346  
5117 100350  
5118 100354  
5119 100360  
5120 100364  
5121 100370  
5122 100376  
5123 100402  
5124 100404  
5125 100406  
5126 100410  
5127 100412  
5128 100416  
5129 100420  
5130 100422  
5131 100424  
5132 100426  
5133 100430  
5134 100436  
5135 100442  
5136 100444  
5137 100446  
5138 100450  
5139 100452  
5140 100454  
5141 100456  
5142 100456  
5143 100460  
5144 100462  
5145 100464  
5146 100466  
5147 100470

100342 014501  
100342 010546  
100344 010102  
100350 062702 000002  
100354 012703 100526  
100360 012704 100456  
100364 012705 100460  
100370 012737 000012 100444  
100376 005037 100542  
100402 161411  
100404 005612  
100406 161512  
100410 002403  
100412 005237 100542  
100416 000771  
100420 062411  
100422 005512  
100424 062412  
100426 022525  
100430 052737 000060 100542  
100436 113723 100542  
100442 005327  
100444 000000  
100446 001353  
100450 105023  
100452 012605  
100454 000207  
100456 145000  
100460 035632  
100462 160400  
100464 002765  
100466 113200  
100470 000230

```
      P$POP  R1          ; PUT ADDRESS OF BINARY WORD INTO R1
      MOV    -(R5),R1
5115  MOV    R5, -(SP)
5116  MOV    R1,R2      ; PUT ADDRESS OF SECOND WORD INTO R2
5117  ADD    #2,R2
5118  MOV    #DECSTR,R3 ; PUT ADDRESS OF OUPUT STRING INTO R3
5119  MOV    #TENPWR,R4 ; ADDRESS OF TEN POWER TABLE
5120  MOV    #TENPWR+2,R5
5121  MOV    #10.,40$
5122  CLR    PART      ; CLEAR PARTIAL COUNTER
5123  SUB    (R4),(R1) ; SUBTRACT 10 POWER
5124  SBC    (R2)
5125  SUB    (R5),(R2)
5126  BLT    30$      ; BRANCH IF 10 POWER TOO LARGE
5127  INC    PART      ; ELSE ADD 1 TO PARTIAL
5128  BR     20$      ; LOOP
5129  ADD    (R4)+,(R1) ; RESTORE BINARY WORDS
5130  ADC    (R2)      ; AND POINT R4 TO NEXT TABLE ENTRIES
5131  ADD    (R4)+,(R2)
5132  CMP    (R5)+,(R5)+
5133  BIS    #'0,PART  ; CHANGE PARTIAL TO ASCII
5134  MOVB  PART,(R3)+ ; AND PUT INTO OUTPUT STRING
5135  DEC    (PC)+     ; HAVE WE DONE ALL 10 DIGITS
5136  .WORD 0
5137  BNE    10$      ; IF NO, BRANCH
5138  CLRB  (R3)+     ; IF YES, TERMINATE WITH ZERO
5139  MOV    (SP)+,R5
5140  RETURN
                    RTS    PC
TENPWR:
      145000      ; 1.0 E09
      35632      ;
      160400      ; 1.0 E08
      2765       ;
      113200     ; 1.0 E07
      230
```

5148	100472	041100	041100	:	1.0 E06
5149	100474	000017	17	:	
5150	100476	103240	103240	:	1.0 E05
5151	100500	000001	1	:	
5152	100502	023420	23420	:	1.0 E04
5153	100504	000000	0	:	
5154	100506	001750	1750	:	1.0 E03
5155	100510	000000	0	:	
5156	100512	000144	144	:	1.0 E02
5157	100514	000000	0	:	
5158	100516	000012	12	:	1.0 E01
5159	100520	000000	0	:	
5160	100522	000001	1	:	1.0 E00
5161	100524	000000	0	:	
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 ; P$TRV SUBROUTINE
5170 ;
5171 ; PARSE THE COMMAND LINE SUBROUTINE
5172 ; TAKE ACTIONS (VIA ACTION TREE) AS PARSING LINE
5173 ; PARSING DIRECTIONS FROM "CLI PARSING NODES"
5174 ; REGS USED:
5175 ;
5176 ; R1,R5=SCRATCH P$NUM=NUMERIC CODE FROM DATA
5177 ; R2=ACTION CODE PARAMETER FROM TREE
5178 ; R3=PARSE TREE POINTER
5179 ; R4=INPUT STRING POINTER
5180 ; CALLING SEQUENCE:
5181 ; JSR PC,P$TRV
5182 ;--
5183
5184 100544 P$TRV: MOV P$BUFA,R4
5185 100544 013704 003204 MOV P$TREE,R3
5186 100550 013703 003206 P$TR5: CMPB (R3),#3 ;SEE IF ONE OF FIRST THREE SPECIAL CODES
5187 100554 121327 000003 BLE 10$ ;IF YES, DON'T CHECK INPUT STRING
5188 100560 003405 TSTB (R4) ;SEE IF ANY CHARS LEFT IN INPUT STRING
5189 100562 105714 BEQ P$EXIT ;BR IF NO
5190 100564 001441 CMPB (R3),#11. ;SEE IF SPECIAL CLI CHAR CODE OR ASCII
5191 100566 121327 000013 BGT 30$ ;BR IF REGULAR ASCII CHAR.
5192 100572 003023 10$: MOVB (R3),R1 ;GET SPECIAL CHAR CODE INTO R5
5193 100574 111301 ASL R1
5194 100576 006301 MOV 20$(R1),R1 ;BUILD TRAVERSE ROUTINE ADDRESS
5195 100600 016101 100614 ADD #20$,R1
5196 100604 062701 100614 JSR PC,(R1) ;JSR TO SPECIAL CLI TRAVERSE ROUTINE
5197 100610 004711 BR P$TR5 ;GO SEE IF MORE OF STRING LEFT
5198 100612 000760
5199
5200
5201 100614 000114 20$: .WORD TRVERR-20$ ;TRAVERSE TABLE FOR "CLI FUNCTIONS"
5202 100616 000134 .WORD TRVEXI-20$ ;1
5203 100620 000152 .WORD TRVBR-20$ ;2
5204 100622 000162 .WORD TRVP1F-20$ ;3
5205 100624 000204 .WORD TRVSPA-20$ ;4
5206 100626 000270 .WORD TRVNUM-20$ ;5
5207 100630 000612 .WORD TRVALP-20$ ;6
5208 100632 000656 .WORD TRVALN-20$ ;7
5209 100634 000270 .WORD TRVOCT-20$ ;8
5210 100636 000256 .WORD TRVDEC-20$ ;9
5211 100640 000744 .WORD TRVSTR-20$ ;10
5212
5213 ;NOT A SPECIAL CODE
5214
5215 100642 121314 30$: CMPB (R3),(R4) ;SEE IF FIRST CHAR OF STRING IS A MATCH
5216 100644 001403 BEQ 40$ ;BR IF A MATCH
5217 100646 004737 100712 JSR PC,TRVBRC ;IF NOT A MATCH, GO TAKE MISS BRANCH
5218 100652 000740 BR P$TR5 ; THEN GO BACK PT'G TO MISS NODE
5219 100654 005204 40$: INC R4 ;IF A MATCH, INCR. CHAR POINTER
5220 100656 004737 100672 JSR PC,TRVACT ; GO DO ACTION DEFINED BY
5221 100662 062703 000004 ADD #4,R3 ; ACTION CODE IN CLI NODE, THEN
5222 ; ADJUST PTR TO NEXT CLI NODE

```

```

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: MOV      1(R3),R2      ;GET ACTION CODE FROM CLI NODE
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$TRV
5244
5245          ;-----
5246          ;ERROR HANDLING
5247 100730 004737 100672    TRVERR: JSR      PC,TRVACT      ;TAKE ERROR ACTION
5248 100734 112737 177777 003221    MOV      #-1,P$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$:    JMP      TRVNOB
5268
5269          ;SPACE ACTION CODE
5270 101020 005001          TRVSPA: CLR      R1          ;CLEAR "SPACE OR TAB FOUND" FLAG
5271 101022 121427 000011    10$:    CMPB     (R4),#11          ;SEE IF CHAR. IN CMD LINE = TAB
5272 101026 001003          BNE     20$
5273 101030 005204          INC     R4          ;BR IF NO, NOT A TAB
5274 101032 005201          INC     R1          ;INC INPUT STRING POINTER
5275 101034 000772          BR      10$          ;INDICATE A TAB FOUND
5276
5277 101036 121427 000040    20$:    CMPB     (R4),#40          ;GO CHECK NEXT CHAR
5278 101042 001003          BNE     30$
5279 101044 005204          INC     R4          ;SEE IF CHAR. IN CMD LINE = SPACE
5279 101044 005204          INC     R4          ;BR IF NO, NON-SPACE OR NON-TAB CHAR.
5279 101044 005204          INC     R4          ;INC INPUT STRING POINTER

```



```

COMMAND LINE TRAVERSE ROUTINES
5280 101046 005201          INC      R1          ;INDICATE A SPACE FOUND
5281 101050 000764          BR       10$         ;GO CHECK NEXT CHAR
5282 101052 005701          30$:    TST      R1          ;SEE IF ANY SPACES OR TABS FOUND
5283 101054 001404          BEQ     40$         ;BR IF NO, TAKE NO ACTION
5284 101056 004737 100672  JSR     PC,TRVACT   ;GO TAKE ACTION IF ANY FOUND
5285 101062 000137 100722  JMP     TRVNOB      ;JUST GO UPDATE R3 TO NEXT NODE IF OK
5286 101066 000137 100712  40$:    JMP     TRVBRC      ;TAKE BRANCH (MISS) IF NONE FOUND
5287
5288
5289 101072 012737 000012 003216 TRVDEC: MOV     @10.,P$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$RADX ;USE OCTAL AS RADIX AND ASSUME *
5293 101112          TRVNMA: PUSH    R5
5294 101112 010546          MOV     R5,-(SP)
5294 101114 005001          CLR     R1          ;CLEAR DIGIT COUNTER
5295 101116 121427 000053          CMPB   (R4),@' *   ;SEE IF THERE'S A * SIGN THERE
5296 101122 001001          BNE    10$         ; BR IF NO
5297 101124 000406          BR     20$         ; ELSE P$RADX ALREADY SAYS *, JUST BR
5298 101126 121427 000055          10$:    CMPB   (R4),@' - ;SEE IF THERE'S A - SIGN THERE
5299 101132 001004          BNE    30$         ; BR IF NO
5300 101134 112737 177777 003217 MOV     @-1,P$RADX+1 ;SET "MINUS FLAG" (HI BYTE OF P$RADX)
5301 101142 005204          20$:    INC     R4          ;BUMP R4 TO POINT TO FIRST CHAR
5302
5303 101144 121427 000060          30$:    CMPB   (R4),@60   ;SEE IF CHAR. LESS THAN A "0"
5304 101150 002434          BLT    60$         ;BR IF YES (NOT NUMERIC)
5305 101152 121427 000067          CMPB   (R4),@67   ;SEE IF CHAR. GREATER THAN A "7"
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
5311 101176 012746 052222          MOV     @CLIBRX,-(SP)
5311 101202 012746 000001          MOV     @1,-(SP)
5311 101206 010600          MOV     SP,R0
5311 101210 104417          TRAP   C$PNTF
5311 101212 062706 000004          ADD     @4,SP
5312 101216 112737 177777 003221 MOV     @-1,P$GDBD ;SET ERROR RETURN FLAG
5313 101224 000475          BR     110$        ; PRINT ERROR AND TAKE MISS
5314
5315 101226 121427 000071          40$:    CMPB   (R4),@71   ;SEE IF CHAR. GREATER THAN A "9"
5316 101232 003003          BGT    60$         ;BR IF YES (NOT NUMERIC)
5317 101234 005204          50$:    INC     R4          ;UPDATE CMD LINE PTR TO NEXT CHAR.
5318 101236 005201          INC     R1          ;INDICATE A NUMERIC FOUND
5319 101240 000741          BR     30$         ;GO LOOK AT NEXT CHAR.
5320
5321 101242 005701          60$:    TST     R1          ;SEE IF FOUND ANY NUMERICS
5322 101244 001465          BEQ    110$        ;BR IF NO, TAKE "MISS" BRANCH
5323 101246 010405          MOV     R4,R5      ;GET POINTER TO START OF NUMERIC STRING
5324 101250 160105          SUB     R1,R5
5325 101252 005037 003214          CLR     P$NUM      ;CLEAR LOC. WHERE VALUE WILL BE STORED
5326 101256 112502          70$:    MOV     (R5),R2    ;GET ASCII CHAR AND CONVERT IT TO A @
5327 101260 162702 060060          SUB     @60,R2
5328 101264 006337 003214          ASL    P$NUM
5329 101270 103440          BCS    100$
5330 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
	101360	012605						MOV (SP),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	101506	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$
5382 101532 005204          INC      R4
5383 101534 005201          INC      R1
5384 101536 000756          BR       10$
5385 101540 005701          30$:   TST      R1
5386 101542 001404          BEQ      40$
5387 101544 004737 100672   JSR      PC,TRVACT
5388 101550 000137 100722   JMP      TRVNOB
5389 101554 000137 100712   40$:   JMP      TRVBRC
5390 101560          TRVSTR: PUSH    R5
          MOV      R5,-(SP)
          ;POINT R1 TO CMD STRING
          ;BR IF YES (NOT ALPHA)
          ;UPDATE CMD LINE PTR TO NEXT CHAR
          ;INDICATE AN ALPHA FOUND
          ;GO LOOK AT NEXT CHAR.
          ;SEE IF ANY ALPHANUM'S WERE FOUND
          ;BR IF NO
          ;IF ANY FOUND TAKE ACTION
          ;THEN UPDATE R3 TO NEXT NODE -NO BRANCH
          ;NONE FOUND, TAKE MISS BRANCH
          ;SAVE R5
5391 101562 010546          MOV      R4,R1
5392 101564 010305          MOV      R3,R5
5393 101566 062705 000006   ADD      #6,R5
5394 101572 005037 003212   CLR      P%CNT
5395 101576 105715          10$:   TSTB     (R5)
5396 101600 001411          BEQ      20$
5397 101602 105711          TSTB     (R1)
5398 101604 001407          BEQ      20$
5399 101606 121115          CMPB     (R1),(R5)
5400 101610 001005          BNE      20$
5401 101612 005237 003212   INC      P%CNT
5402 101616 005201          INC      R1
5403 101620 005205          INC      R5
5404 101622 000765          BR       10$
          ;BR TO CONTINUE CHECKING CHARS.
5405
5406 101624 005737 003212   20$:   TST      P%CNT
5407 101630 001407          BEQ      30$
5408 101632 010104          MOV      R1,R4
5409 101634          POP      R5
          ;WHEN DONE SEE IF ANY MATCHES FOUND
          ;BR IF NO, GO TAKE THE MISS BRANCH
          ;POINT CMD POINTER TO END OF STRING &
          ;RESTORE R5
          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
          ; DISPLACEMENT DUE TO MATCH STRING)
5413
5414 101650          30$:   POP      R5
          ;RESTORE R5
          MOV      (SP),R5
5415 101652 000137 100712   JMP      TRVBRC
          ; GO TAKE BRANCH
          ; (PARSED OK), -1 IF ILL CMD.....
5416

```

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

.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				
ADR./	0	END OF LINE	CTARGET	N/A
ADR.				
ADR./CHAR. OR "OPR SEL/CHAR. OTHER THAN "A" "T" OR BLANK	-1	/	CTARGET	N/A
"	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$: CMPB (R4),#0 ;SEE IF NUL CHAR.
; IF YES, RETURN
CMPB (R4),#40 ;SEE IF ILLEGAL CHARACTER
BLT 20$ ;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.
MOV B, #1, P$GDBD ;SET ERROR FLAG
BR 90$ ;RETURN
30$: TST R1 ;SEE IF VALID CHARACTERS FOUND

```

5475	101736	001772				BEG	20\$				; IF NO, ILLEGAL CHAR.
5476	101740	012737	000000	003734	40\$:	MOV	#CTARGT,CFLAG				;SET TARGET FLAG
5477	101746	000456				BR	90\$				;RETURN
5478	101750	005701			50\$:	TST	R1				;SEE IF VALID CHARACTERS FOUND
5479	101752	001764				BEG	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				BEG	40\$				; IF YES, TAKE DEFAULT OF TARGET
5484	101770	121427	000101			CMPB	(R4),#'A				;IS NEXT CHAR. "A"
5485	101774	001412				BEG	60\$				; IF YES, BR 60\$
5486	101776	121427	000124			CMPB	(R4),#'T				;IS NEXT CHAR. "T"
5487	102002	001756				BEG	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				BEG	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  
5514  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5526  
5527  
5528  
5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5540  
5541  
5542  
5543

102106  
102106  
  
102106 004737 105170  
102112  
102112 000167  
102114 000000  
  
  
  
  
  
  
  
  
  
  
  
102116  
102116  
102116 104425

.SBTTL REPORT CODING SECTION

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

BGNRPT

L\$RPT::

\*\*\*\*\*  
: 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.  
\*\*\*\*\*

JSR PC,ACTSUM  
EXIT RPT

.WORD JSJMP  
.WORD L10006-2-

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

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

.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 PTABLE
: ENTRY TO THE ITEM IN QUESTION.) IF THE PARTICULAR
: ITEM DOES NOT APPLY, LEAVE ENTRY AS -1. WHEN THE RUNTIME
: SERVICES EXECUTES A GPHARD, IT USES THESE OFFSETS (IF NOT
: SET TO -1) TO GET THE ITEMS AND COMPARE WITH THOSE SAVED
: IN THE XXDP+ MONITOR. IF THE UNIT BEING REQUESTED MATCHES THE
: LOAD DEVICE, THE RUNTIME SERVICES RETURN AN INCOMPLETE FLAG ON
: THE GPHARD.
: *****

```

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$REFG
5626 102162          BCOMPLETE      START
      102162 103424          ;IF HERE BECAUSE OF "RESTART", DO SOME INIT
5627 102164          READEF  #EF.RESTART      ;IF HERE BECAUSE OF "RESTART", DO SOME INIT
      102164 012700 000037          MOV      #EF.RESTART,RO
      102170 104447          TRAP      C$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$:
5635 102214          READEF  #EF.NEW          ;IF HERE ON NEW PASS, SKIP SOME INIT
      102220 012700 000035          MOV      #EF.NEW,RO
      102220 104447          TRAP      C$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 001600 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$:
5661 102366 012737 000100 003742      MOV      #100,CLKVEC   ; SET UP INTERRUPT VECTOR
5662 102374 012737 003746 003736      MOV      #CLKEN,CLKCSR ; FAKE A CSR LOCATION
5663 102402 012737 000001 003756      MOV      #1,TIMER1    ; SET TIMER 1 VALUE
5664 102410                      SETVEC  CLKVEC,#CLKINT,#0 ; ENABLE CLOCK INTERRUPTS
102410 012746 000000                      MOV      #0,-(SP)
102414 012746 070530                      MOV      #CLKINT,-(SP)
102420 013746 003742                      MOV      CLKVEC,-(SP)
102424 012746 000003                      MOV      #3,-(SP)
102430 104437                      TRAP    C$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$
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)+,QNAPRO ;SAVE PRIORITY

```

```

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      #STBLEN,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
      103030 000220          .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      @IDTDAT,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      100$: CALL      BINHEX @DEPADR,@6,@STRBUF ; PUT ADDRESS INTO HEX FORMAT
                                MOV      @STRBUF,(R5).
                                MOV      @6,(R5).
                                MOV      @DEPADR,(R5).
                                JSR      R4,PREG14
                                .WORD   BINHEX-ANCHOR
103126 012725 002324
103132 012725 000006
103136 012725 003774
103142 004437 070656
103146 005150
5737 103150      PRINTS  @HMSG1,@STRBUF ; PRINT ADDRESS
103150 012746 002324      MOV      @STRBUF,-(SP)
103154 012746 055046      MOV      @HMSG1,-(SP)
103160 012746 000002      MOV      @2,-(SP)
103164 010600      MOV      SP,R0
103166 104416      TRAP    C:PNTS
103170 062706 000006      ADD     @6,SP
5738 103174 012737 000000 002372  MOV      @ALPHA,P$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
5745 103234      RESTRT:
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      CLKMZ,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   C$SVEC
103274 062706 000010      ADD     @10,SP
5750 103300 013777 003746 100430  MOV      CLKEN,@CLKCSR ; SET ENABLE BITS IN THE CLOCK TO START
5751 103306      SETPRI  @PRI00 ; SET PRIORITY=0 SO CLOCK CAN INTERRUPT
103306 012700 000000      MOV     @PRI00,R0
103312 104441      TRAP   C$SPRI
5752 103314      CALL   QNAINI ; INITIALIZE THE QNA
103314 004437 070656      JSR    R4,PREG14
103320 000220      .WORD  QNAINI-ANCHOR
5753 103322      P$POP  R0 ; GET RETURN STATUS
103322 014500      MOV     -(R5),R0
5754 103324 001401      BEQ    10$ ; BRANCH IF NO ERRORS
5755 103326 000406      BR     INICLN
5756 103330      10$:

```

```

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 012602          MOV      (SP)+,R3
103346 012603          MOV      (SP)+,R4
103350 012604          ; ABORT PASS
5763 103352          DOCLN      TRAP      C#DCLN
103352 104444          INIEXI:  POP      R2,R3,R4      ; RESTORE THE REGISTERS
5764 103354          MOV      (SP)+,R2
5765 103354 012602          MOV      (SP)+,R3
103356 012603          MOV      (SP)+,R4
103360 012604          ; EXIT INIT SECTION
5766 103362          EXIT      INIT      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          ;*****
5774          ;      INSERT MESSAGES THAT ARE USED ONLY
5775          ;      DURING THE INITIALIZE SECTION.
5776          ;*****
5777          ;*****
5779          .EVEN
5780          ENDINIT
5781
5782 103366          L10010:  TRAP      C#INIT
103366
103366 104411

```

5784  
5785  
5786  
5787  
5788  
5789  
5790  
5791  
5792  
5793 103370  
103370  
5794  
5796  
5797  
5798  
5799  
5801  
5802 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\$AUTO

5804  
5805  
5806  
5807  
5808  
5809  
5810  
5811  
5812  
5813  
5814  
5815  
5816  
5817  
5818  
5819  
5820 103372  
103372  
5821  
5823  
5824  
5825  
5826  
5827  
5828  
5830 103372  
103372 010225  
5831 103374  
5832 103374  
103374 013700 000000  
103400 104441  
5833 103402 005737 050256  
5834 103406 001417  
5835 103410  
103410 004437 070656  
103414 003044  
5836 103416  
103416 014502  
5837 103420 001765  
5838 103422 013702 024226  
5839 103426  
103426 012725 024226  
103432 012725 000001  
103436 004437 070656  
103442 006450  
5840 103444 000753  
5841 103446  
5842 103446 013702 047772  
5843 103452 012737 000001 003756  
5844 103460 012762 000002 000016  
5845 103466 012737 000001 003756  
5846 103474  
5847 103474 005737 003756  
5848 103500 001375  
5849 103502 005062 000016  
5850 103506 005077 100224  
5851 103512

.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

: --

BGNCLN

L\$CLEAN::

\*\*\*\*\*  
: 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 "+C".  
\*\*\*\*\*

P\$PUSH R2 ; PUT R2 ON THE STACK  
MOV R2,(R5)+  
10\$: SETPRI PRI00 ; MAKE SURE CLOCKS CAN INTERRUPT  
MOV PRI00,R0  
TRAP C\$SPRI  
TST NIRCNT ; SEE IF RECEVE COUNT ZERO  
BEQ 20\$ ; IF YES, EXIT  
CALL RECEVE ; ELSE, CLEAR OUT UNWANTED ENTRIES  
JSR R4,PREG14  
.WORD RECEVE-ANCHOR  
P\$POP R2  
MOV -(R5),R2  
BEQ 10\$ ; IF ANY WERE FOUND, UPDATE NEXT POINTER  
MOV RRGNXTR2 ; GET THE DESCRIPTOR  
CALL GETRNX @1,RRGNXT ; UPDATE NEXT AND PREVIOUS DESCRIPTOR POINTERS  
MOV @RRGNXT,(R5)+  
MOV @1,(R5)+  
JSR R4,PREG14  
.WORD GETRNX-ANCHOR  
BR 10\$ ;SEE IF ANY MORE ENTRIES  
20\$: MOV QNAADO,R2 ; GET DEVICE ADDRESS  
MOV @1,TIMER1 ; TIMER FOR INITIIALIZE  
MOV @MRESET,CSR(R2) ; RESET THE DEVICE  
MOV @1,TIMER1 ; TIMER FOR INITIIALIZE  
30\$: TST TIMER1 ; INIT DONE?  
BNE 30\$ ; BRANCH IF NOT  
CLR CSR(R2) ; FINISH THE RESET  
CLR @CLKCSR ; DISABLE CLOCK  
SETPRI @PRI07 ; SET PROCESSOR PRIORITY BACK TO 7

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



DROP UNIT SECTION

.SBTTL DROP UNIT SECTION

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

BGNDU

L\$DU: :

\*\*\*\*\*  
: 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 DU

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

\*\*\*\*\*  
: 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

5871  
5872  
5873  
5874  
5875  
5876  
5877  
5878 103530  
103530  
5879  
5881  
5882  
5883  
5884  
5885  
5886  
5888  
5889 103530 000167  
103530 000000  
103532 000000  
5890  
5892  
5893  
5894  
5895  
5896  
5897  
5898  
5899  
5900  
5902  
5903  
5904  
5905 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 000167  
103536 000000  
103540  
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

L\$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

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  
103554  
103555  
103556  
103560  
103562  
103564  
103566  
103570  
103572  
103574  
6025  
6026

103544  
103544 105037 003221  
103550 105037 003220  
103554 104443  
103556 000406  
103560 002202  
103562 000142  
103564 052104  
103566 000000  
103570 000001  
103572 000110  
103574  
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\$LOLIM  
.WORD T\$HILIM  
10000\$:  
MOV #CMDBUF,P\$BUFA  
MOV #CLITRE,P\$TREE

6027	103610	012737	103726	003210	MOV	#CLIACT,P\$ACT		
6028	103616	005037	003734		CLR	CFLAG		;CLEAR QUALIFIER FLAG
6029	103622	004737	100544		JSR	PC,P\$TRV		;GO PARSE COMMAND TREE
6030	103626	105737	003221		TSTB	P\$GDBD		;SEE IF PARSED OK, OR AN ERROR
6031	103632	001412			BEQ	10\$		
6032	103634				PRINTF	#CLIERM		;IF NOT PRINT ERROR MESSAGE
	103634	012746	052113				MOV	#CLIERM,-(SP)
	103640	012746	000001				MOV	#1,-(SP)
	103644	010600					MOV	SP,RO
	103646	104417					TRAP	C\$PNTF
	103650	062706	000004				ADD	#4,SP
6033	103654	000137	103544		JMP	GETCL		
6034	103660	105737	003220	10\$:	TSTB	P\$NNUF		;SEE IF INCOMPLETE COMMAND TYPED
6035	103664	001412			BEQ	20\$		
6036	103666				PRINTF	#CLINUF		;IF NOT PRINT ERROR MESSAGE
	103666	012746	052144				MOV	#CLINUF,-(SP)
	103672	012746	000001				MOV	#1,-(SP)
	103676	010600					MOV	SP,RO
	103700	104417					TRAP	C\$PNTF
	103702	062706	000004				ADD	#4,SP
6037	103706	000137	103544		JMP	GETCL		
6038	103712	022737	000020	003734	20\$:	CMP	#CEXIT,CFLAG	;WAS EXIT COMMAND TYPED?
6039	103720	001311			BNE	GETCL		; IF NOT GET NEW COMMAND LINE
6040	103722				EXIT	TST		; ELSE EXIT
	103722	104432					TRAP	C\$EXIT
	103724	013164					.WORD	L10015-

6042  
6043  
6044  
6045  
6046  
6047  
6048  
6049  
6050  
6051  
6052  
6053  
6054  
6055  
6056  
6057  
6058  
6059  
6060  
6061  
6062  
6063  
6064  
6065  
6066  
6067  
6068  
6069  
6070  
6071  
6072  
6073  
6074  
6075  
6076  
6077  
6078  
6079  
6080  
6081  
6082  
6083  
6084  
6085  
6086  
6087  
6088  
6089  
6090  
6091  
6092  
6093  
6094  
6095  
6096  
6097  
6098

103726  
103726 006302 103744  
103730 016202 103744  
103734 062702 103744  
103740 004712  
103742 000207  
  
103744 000124  
103746 000126  
103750 000164  
103752 000474

```

.SBTTL          CLI ACTION TABLE AND ROUTINES
:
: **
: FUNCTIONAL DESCRIPTION:
:
:   THESE ARE THE ACTION TABLES TO DEFING WHAT TO DO WITH A COMMAND
:   RECEIVED FROM A USER
:   USER MUST CLEAR/SET P$GDBD IF USE "CLIBIF" IN CONNECTION WITH ACTION
:
: CALLING SEQUENCE
:
:   JMP      CL
:
: INPUT
:
:   COMMAND FROM THE USER
:
: IMPLICIT INPUT
:
:   NONE
:
: OUTPUT
:
:   NONE
:
: IMPLICIT OUTPUT
:
:   NONE
:
: COMPLETION CODES
:
:   NONE
:
: SIDE EFFECTS
:
:   NONE
:
: REGISTERS USED
:
:   R2 - HOLDS ACTION CODE FROM THE PARSING (CLI) NODE
:
: DEBUG
:
:   NONE
:
:--
: CLIACT:
:
:   ASL      R2          ;MULTIPLY ACTION CODE BY 2
:   MOV      10$(R2),R2 ;OFFSET VALUE
:   ADD      #10$,R2    ;ADD BASE VALUE
:   JSR      PC,(R2)    ;GO DO ACTION
:   RTS      PC         ;RETURN TO TRVACT
:
:   ;BRIEF DESCRIPTION OF ACTION TAKEN
:
: 10$: .WORD  ACTNUL-10$ ;0-NULL
:      .WORD  ACTHLP-10$ ;1-HELP
:      .WORD  ACTNOD-10$ ;2-NODE
:      .WORD  ACTBLD-10$ ;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	ACT1AL-10\$	;21-SET '1ALT' 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	ACTSIZE-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 MODE

```

6135 ;
6136 ;ACTION ROUTINE TO INDICATE THAT NOT ENOUGH COMMAND
6137 ;INFORMATION HAS BEEN ENTERED
6138 ;
6139 ;
6140 104062 ACTNUF:
6141 104062 112737 177777 003220      MOVB    #-1,P$NNUF          ;SET FLAG TO SAY NEED MORE OF COMMAND
6142 ;
6143 ;
6144 ;ACTION ROUTINE TO DO NOTHING
6145 ;
6146 ;
6147 104070 ACTNUL:
6148 104070 000207      RTS     PC                ;RETURN TO PARSER
6149 ;
6150 ;
6151 ;ACTION ROUTINE TO PRINT OUT HELP FILE
6152 ;
6153 ;
6154 ;
6155 104072 ACTHLP:
6156 104072      P$PUSH  R2                ;SAVE R2
6157 104072 010225      MOV     R2,(R5)+          ;MOVE POINTER TO BEGINING OF HELP FILE
6158 104074 012702 003224      MOV     #HLPTAB,R2        ;PRINT LINE AND INCREMENT POINTER
6159 104100 012246      PRINTF  (R2)+          ;PRINT LINE AND INCREMENT POINTER
6160 104100 012746 000001      MOV     (R2)+,-(SP)
6161 104102 012746 000001      MOV     #1,-(SP)
6162 104106 010600      MOV     SP,R0
6163 104110 104417      TRAP   C$PNTF
6164 104112 062706 000004      ADD     #4,SP
6165 104116 020227 003322      CMP     R2,#HLPEND        ;SEE IF ENTIRE FILE PRINTED
6166 104122 001366      BNE    10$                ;IF NOT, PRINT MORE
6167 104124 014502      P$POP  R2                ;ELSE, RESTORE R2 AND RETURN
6168 104124 000207      RTS     PC                ;ELSE, RESTORE R2 AND RETURN
6169 ;
6170 ;ACTION ROUTINE TO READ IN NODE PHY. ADDRESS, STORE IT IN ADRBUF
6171 ;AND ENTER IT INTO THE NODE TABLE
6172 ;
6173 ;
6174 104130 ACTNOD:
6175 104130 105037 003220      CLRB   P$NNUF            ;CLEAR NOTNUF FLAG
6176 104134 004737 101656      JSR    PC,TRVADR         ;TRAVERSE ADDRESS, CHECK IF TARGET OR ASSIST
6177 104140 105737 003221      TSTB  P$GDBD            ;CHECK IF RESULTS OK
6178 104144 001134      BNE    60$                ;IF NOT, RETURN WITH -1 IN P$GDBD
6179 104146 012725 000006      CALL  EDPACK CBOADR,#ADRBUF,#6 ;GET ADDRESS INTO BUFFER
6180 104146 012725 002316      MOV     #6,(R5)+
6181 104152 012725 002316      MOV     #ADRBUF,(R5)+
6182 104156 013725 002370      MOV     CBOADR,(R5)+
6183 104162 004437 070656      JSR    R4,PREG14
6184 104166 004600      .WORD  EDPACK-ANCHOR
6185 104170 014501      P$POP  R1                ;CHECK RESULTS FOR NUMBER OF CHAR.S
6186 104170 001411      BEQ    20$                ;IF OK, BRANCH TO 20$
6187 104172 012746 052746      PRINTF #CADRER          ;ELSE PRINT ERROR MESSAGE
6188 104174 012746 052746      MOV     #CADRER,-(SP)

```





```

6201
6202
6203
6204
6205
6206 104440
6207 104440
104440 012746 054415
104444 012746 000001
104450 010600
104452 104416
104454 062706 000004
6208 104460
104460 012746 054530
104464 012746 000001
104470 010600
104472 104416
104474 062706 000004
6209 104500
104500 012746 054643
104504 012746 000001
104510 010600
104512 104416
104514 062706 000004
6210 104520
104520 010425
104522 010325
104524 010225
6211 104526
104526 012725 000001
104532 004437 070656
104536 001232
6212 104540
104540 014502
6213 104542 001406
6214 104544
104544 104456
104546 000031
104550 064157
104552 000000
6215 104554 000137 105154
6216 104560
6217 104560 005237 002200
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
6226 104624
6227 104624
104624 104422
6228 104626 005714
6229 104630 001503
6230 104632

```

```

;
; ACTION ROUTINE TO BUILD NODE TABLE USING THE PERIODIC ADDRESS
; IDENTIFICATION MESSAGES SENT OUT BY NODE ON THE NI
;
ACTBLD:
PRINTS @MSG1 ; PRINT 'BUILD' COMMAND MESSAGE
MOV @MSG1,-(SP)
MOV @1,-(SP)
MOV SP,R0
TRAP C$PNTS
ADD @4,SP
PRINTS @MSG11
MOV @MSG11,-(SP)
MOV @1,-(SP)
MOV SP,R0
TRAP C$PNTS
ADD @4,SP
PRINTS @MSG12
MOV @MSG12,-(SP)
MOV @1,-(SP)
MOV SP,R0
TRAP C$PNTS
ADD @4,SP
P$PUSH R2,R3,R4 ; SAVE REGISTERS
MOV R4,(R5)
MOV R3,(R5)
MOV R2,(R5)
CALL SETUP @ADDMUL ; WRITE MULTICAST ADDRESS
MOV @ADDMUL,(R5)
JSR R4,PREG14
.WORD SETUP-ANCHOR
P$POP R2 ; CHECK FOR ERROR
MOV -(R5),R2
BEQ 10$ ; IF OK, CONTINUE
ERRHRD 25,EMSG25 ; ELSE REPORT ERROR
TRAP C$ERHRD
.WORD 25
.WORD EMSG25
.WORD 0
JMP 100$ ; LEAVE
10$:
INC BLDFLG ; INDICATE THAT WE ARE IN BUILD MODE
CLR TEMP ; CLEAR 'NO. NODES IN LAST MIN.' COUNTER
CLR TEMP1 ; CLEAR NODE TYPE ARGUMENT (SET TO TARGET)
CLR TEMP2 ; SET INTERVAL COUNTER
MOV @13,TEMP3 ; SET 'MINS. SINCE LAST NEW NODE' COUNTER
MOV @NODTBL,SLOT ; SET SLOT TO BEGINING OF NODE TABLE
MOV @TIMERS,R4 ; SET UP FOR 1 MINUTE LOOP
20$:
MOV @60.,(R4)
30$:
BREAK ; ALLOW FOR CONTROL C INTERRUPTION
TRAP C$BRK
TST (R4) ; SEE IF INTERVAL IS UP
BEQ 70$ ; IF YES, BRANCH
CALL RECEVE ; ELSE, CHECK FOR RECEPTION OF ID MESSAGE

```

6231	104632	004437	070656				JSR R4,PREG14 .WORD RECEIVE-ANCHOR
	104636	003044					MOV -(R5),R2
	104640	014502		P\$POP	R2		: R2 HOLDS NO OF MESSAGES RECEIVED
6232	104642	001770		BEQ	30\$		: IF NONE, KEEP LOOKING
6233	104644	013703	024226	MOV	RRGNXT,R3		: IF ONE, GET RECEIVE RING POINTER
6234	104650	012725	024226	CALL	GETRNX #1,@RRGNXT		: UPDATE POINTER, VALIDATE PREVIOUS DESCRIPTOR
	104654	012725	000001				MOV @RRGNXT,(R5)
	104660	004437	070656				MOV #1,(R5)
	104664	006450					JSR R4,PREG14
6235	104666	016303	000004	MOV	LOADD(R3),R3		.WORD GETRNX-ANCHOR
6236	104672	062703	000006	ADD	@SOURCC,R3		: POINT R3 TO MESSAGE BUFFER
6237	104676	012702	002412	MOV	@NODTBL,R2		: POINT R3 TO NODE ADDRESS
6238	104702						: POINT R2 TO NODE TABLE
6239	104702			40\$: CALL	CMPADR R2,R3		: SEE IF NODE ALREADY ON TABLE
	104702	010325					MOV R3,(R5)
	104704	010225					MOV R2,(R5)
	104706	004437	070656				JSR R4,PREG14
	104712	026004					.WORD CMPADR-ANCHOR
6240	104714	014501		P\$POP	R1		: GET RESULTS OF COMPARE
	104716	001742		BEQ	30\$		MOV -(R5),R1
6241	104720	062702	000010	50\$: ADD	#10,R2		: DON'T ADD TO TABLE IF ALREADY THERE
6242	104724	022712	177777	CMP	#-1,(R2)		: ELSE CHECK NEXT TABLE ENTRY
6243	104730	001364		BNE	40\$		: SEE IF AT END OF TABLE
6244	104732	004437	070656	CALL	FINDSL		: IF NO, COMPARE NEXT ENTRY
6245	104736	025466					: IF YES, GET NEXT TABLE SLOT
6246	104740	014502		P\$POP	R2		JSR R4,PREG14
6247	104742	001023					.WORD FINDSL-ANCHOR
6248	104744	013702	002410	BNE	60\$		: SEE IF TABLE FULL
6249	104750	012322		MOV	SLOT,R2		MOV -(R5),R2
6250	104752	012322		MOV	(R3),,(R2)		: IF YES, BRANCH
6251	104754	011312		MOV	(R3),,(R2)		: IF NO, ADD NODE TO TABLE
6252	104756	113762	050316 000003	MOV	(R3),,(R2)		: SIX BYTES WORTH
6253	104764	105137	050316	MOV	TEMP1,3(R2)		: SET NODE TYPE (TARGET OR ASSIST)
6254	104770	142737	000376 050316	COMB	TEMP1		: SWITCH TYPE FOR NEXT TIME
6255	104776	005237	050314	BICB	#376,TEMP1		: INCREMENT 'NODES IN LAST MIN.' COUNTER
6256	105002	012737	000013 050322	INC	TEMP		: RESET 'MINS. SINCE LAST NODE' COUNTER
6257	105010	000705		MOV	#13,TEMP3		
6258	105012			BR	30\$		
6259	105012			60\$: PRINTB	@TABFUL,@NOD		: PRINT "TABLE FULL" MESSAGE
6260	105016	012746	053415				MOV @NOD,-(SP)
	105022	012746	053274				MOV @TABFUL,-(SP)
	105026	010600	000002				MOV #2,-(SP)
	105030	104414					MOV SP,R0
	105032	062706	000006				TRAP C\$PNTB
6261	105036	000430					ADD #6,SP
6262	105040			BR	80\$		
6263	105044	005337	050322	70\$: DEC	TEMP3		: SEE IF 10 MINS SINCE LAST NODE
6264	105046	001425		BEQ	80\$		: IF YES, EXIT
6265	105052	023727	050320 000050	INC	TEMP2		: SEE IF TIME IS UP
6266	105052	023727	050320 000050	CMP	TEMP2,#40		

```

6267 105060 001417          BEQ      80$          ; IF YES, EXIT
6268 105062          PRINTS  #BLDMSG,TEMP,TEMP2 ; ELSE, PRINT "STILL WORKING" MESSAGE
        105062 013746 050320          MOV      TEMP2,-(SP)
        105066 013746 050314          MOV      TEMP,-(SP)
        105072 012746 052473          MOV      #BLDMSG,-(SP)
        105076 012746 000003          MOV      #3,-(SP)
        105102 010600          MOV      SP,R0
        105104 104416          TRAP    C:PNTS
        105106 062706 000010          ADD     #10,SP
6269 105112 005037 050314      CLR     TEMP
6270 105116 000640          BR      20$
6271 105120          80$:
6272 105120          CALL    SETUP #KILMUL          ; KILL THE MULTICAST ADDRESS
        105120 012725 000002          MOV      #KILMUL,(R5)+
        105124 004437 070656          JSR     R4,PREG14
        105130 001232          .WORD  SETUP-ANCHOR
6273 105132          P$POP  R2          ; CHECK FOR ERROR
        105132 014502          MOV      -(R5),R2
6274 105134 001405          BEQ     90$          ; IF OK CONTINUE
6275 105136          ERRHRD 34,EMSG25      ; ELSE, REPORT ERROR
        105136 104456          TRAP    C:ERHRD
        105140 000042          .WORD  34
        105142 064157          .WORD  EMSG25
        105144 000000          .WORD  0
6276 105146 000402          BR      100$         ; AND LEAVE
6277 105150          90$:
6278 105150 004737 113676      JSR     PC,ACTSND     ; PRINT NODE TABLE
6279 105154          100$:
6280 105154 005037 002200      CLR     BLDFLG
6281 105160          P$POP  R2,R3,R4    ; INDICATE THAT WE ARE NO LONGER BUILDING
        105160 014502          MOV      -(R5),R2
        105162 014503          MOV      -(R5),R3
        105164 014504          MOV      -(R5),R4
6282 105166 000207          RTS     PC
6283
6284 ; ACTION ROUTINE TO PRINT OUT THE SUMMARY DATA
6285 ;
6286 105170 105037 003220      ACTSUM: CLR B P$NNUF          ; CLEAR NOTNUF FLAG
6287 105174          P$PUSH R2,R3,R4
        105174 010425          MOV      R4,(R5)+
        105176 010325          MOV      R3,(R5)+
        105200 010225          MOV      R2,(R5)+
6288 105202 012701 002714      MOV     #STATBL,R1    ; MOVE ADDRESS OF TABLE TO R1
6289 105206 005711          TST     (R1)          ; SEE IF TABLE EMPTY
6290 105210 001013          BNE     10$          ; IF NOT, CONT.
6291 105212          PRINTF #TABEMT,#SUMM      ; ELSE PRINT 'TABLE EMPTY' MESSAGE
        105212 012746 053422          MOV      #SUMM,-(SP)
        105216 012746 053346          MOV      #TABEMT,-(SP)
        105222 012746 000002          MOV      #2,-(SP)
        105226 010600          MOV      SP,R0
        105230 104417          TRAP    C:PNTF
        105232 062706 000006          ADD     #6,SP
6292 105236 000541          BR      20$
6293 105240 021127 177777      10$:  CMP     (R1),#-1
6294 105244 001536          BEQ     20$
6295 105246 005711          TST     (R1)
6296 105250 001534          BEQ     20$          ; EXIT
                          ; SEE IF AT END OF TABLE
                          ; IF YES, EXIT
                          ; SEE IF REST OF TABLE EMPTY
                          ; IF YES, EXIT

```

6297	105252			CALL	BINHEX R1,#6,#STRBUF	; PRINT SUMMARY DATA	
	105252	012725	002324				MOV #STRBUF,(R5),
	105256	012725	000006				MOV #6,(R5),
	105262	010125					MOV R1,(R5),
	105264	004437	070656				JSR R4,PREG14
	105270	005150					.WORD BINHEX-ANCHOR
6298	105272			PRINTF	#SUMMS1,#STRBUF	; NODE ADDRESS	
	105272	012746	002324				MOV #STRBUF,-(SP)
	105276	012746	070043				MOV #SUMMS1,-(SP)
	105302	012746	000002				MOV #2,-(SP)
	105306	010600					MOV SP,R0
	105310	104417					TRAP C#PNTF
	105312	062706	000006				ADD #6,SP
6299	105316			PRINTF	#SUMMS2	; FIRST HEADER	
	105316	012746	070064				MOV #SUMMS2,-(SP)
	105322	012746	000001				MOV #1,-(SP)
	105326	010600					MOV SP,R0
	105330	104417					TRAP C#PNTF
	105332	062706	000004				ADD #4,SP
6300	105336	012702	000026	MOV	#26,R2	; GET STATISTICS SIZE FOR THIS NODE	
6301	105342	012703	002664	MOV	#STATBF,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	#SUMMS3,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 #SUMMS3,-(SP)
	105402	012746	000004				MOV #4,-(SP)
	105406	010600					MOV SP,R0
	105410	104417					TRAP C#PNTF
	105412	062706	000012				ADD #12,SP
6310	105416			PRINTF	#SUMMS4	; SECOND HEADER	
	105416	012746	070177				MOV #SUMMS4,-(SP)
	105422	012746	000001				MOV #1,-(SP)
	105426	010600					MOV SP,R0
	105430	104417					TRAP C#PNTF
	105432	062706	000004				ADD #4,SP
6311	105436	013702	002700	MOV	STATBF+14,R2	; COMPARE ERRORS	
6312	105442			CALL	BINDEC #STATBF+16	; PUT BYTES COMPARED INTO ASCII STRING	
	105442	012725	002702				MOV #STATBF+16,(R5),
	105446	004437	070656				JSR R4,PREG14
	105452	007446					.WORD BINDEC-ANCHOR
6313	105454			PRINTF	#SUMMS5,R2,#DECSTR	; PRINT THEM OUT	
	105454	012746	100526				MOV #DECSTR,-(SP)
	105460	010246					MOV R2,-(SP)
	105462	012746	070262				MOV #SUMMS5,-(SP)
	105466	012746	000003				MOV #3,-(SP)
	105472	010600					MOV SP,R0
	105474	104417					TRAP C#PNTF
	105476	062706	000010				ADD #10,SP
6314	105502			CALL	BINDEC #STATEF+22	; PUT BYTES TRANSFERED INTO ASCII STRING	

15\$:

105502	012725	002706				MOV	#STATBF+22,(R5)+
105506	004437	070656				JSR	R4,PREG14
105512	007446					.WORD	BINDEC-ANCHOR
6315	105514			PRINTF	#SUMMS6,#DECSTR		; PRINT
	105514	012746	100526				MOV #DECSTR,-(SP)
	105520	012746	070300				MOV #SUMMS6,-(SP)
	105524	012746	000002				MOV #2,-(SP)
	105530	010600					MOV SP,R0
	105532	104417					TRAP C\$PNTF
	105534	062706	000006				ADD #6,SP
6316	105540	000637		BR	10\$		; PLAY IT AGAIN, SAM
6317	105542			P\$POP	R2,R3,R4		
	105542	014502					MOV -(R5),R2
	105544	014503					MOV -(R5),R3
	105546	014504					MOV -(R5),R4
6318	105550	000207		RTS	PC		
6319							
6320							
6321							
6322							
6323							
6324							
6325							
6326							
6327							
6328							
6329							
6330							
6331							
6332							
6333							
6334							
6335							
6336							
6337							
6338							
6339							
6340							
6341							
6342							
6343							
6344	105552						
6345	105552	105737	003222	ACTIDT:			
6346	105556	001402		TSTB	P\$AERR		;SEE IF ADDRESS ENTERED WAS VALID
6347	105560	000137	106466	BEQ	10\$		; IF NOT, EXIT ACTION ROUTINE
6348	105564			JMP	130\$		
6349	105564	105037	003220	10\$:			
6350	105570			CLRB	P\$NNUF		;CLEAR NOTNUF FLAG
	105570	012725	002656	CALL	CMPADR #ADRBUF,#ILLADR		; SEE IF ILLEGAL ADDRESS
	105574	012725	002316				MOV #ILLADR,(R5)+
	105600	004437	070656				MOV #ADRBUF,(R5)+
	105604	026004					JSR R4,PREG14
6351	105606						.WORD CMPADR-ANCHOR
	105606	014501		P\$POP	R1		MOV -(R5),R1
6352	105610	001012		BNE	20\$		; IF NO, CONTINUE
6353	105612			15\$:			
6354	105612			PRINTF	#ILADMS		; ELSE PRINT ILLEGAL ADDRESS MESSAGE

```

; ACTION ROUTINE TO INITIATE THE REQUEST ID TEST TO THE SPECIFIED NODE
;
; ---
; FUNCTIONAL DESCRIPTION
; THIS SUBROUTINE BUILDS AND TRANSMITS REQUEST ID PACKETS
; TO THE NODE SPECIFIED BY THE OPERATOR IN THE COMMAND LINE.
; THE SYSTEM ID INFO OF THE SPECIFIED NODE IS THEN DISPLAYED.
; IF THE NODE DOES NOT RESPOND BEFORE 60 SECONDS HAVE PASSED
; AN ERROR IS REPORTED TO THE OPERATOR.
;
; INPUTS - IMPLICIT - THE SPECIFIED NODE ADDRESS IS LOCATED IN ADRBUF.
;
; OUTPUTS - SYSTEM ID INFO OR ERROR MESSAGE PRINTED TO OPERATOR.
;
; CALLING PROCEDURE - JSR PC, ACTIDT
;
; SIDE EFFECTS - XRGXNT POINTER IS UPDATED BY A CALL TO BLDREQ SUB.
;
; REGISTER USAGE - R1 POINTS TO $WDMO FOR WRITE MODE OPERATIONS.
; R2 IS SCRATCH.
; R3 POINTS TO THE RECEIVED MESSAGE BUFFER.
; R4 POINTS TO TIMEOUT TIMER
;
; ---

```



6382	106012			CALL	RECEVE				: CHECK FOR ANSWER
	106012	004437	070656						JSR R4,PREG14
	106016	003044							.WORD RECEVE-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	CMP	SIRCPT(R3),#"MR	051115			: 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					60\$:			
6393	106070			PRINTF	#EMSG22				: ELSE, REPORT ERROR
	106070	012746	064050						MOV #EMSG22,-(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					70\$:			
6397	106120	005237	050030	INC	S.REC				: INCREMENT 'RECEIVED MESSAGES' COUNTER
6398	106124	062737	000056	ADD	#46.,S.XFER	050042			: 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 FUNCTION 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,MSG24,ERR1	; ERROR -- CAN'T TRANSMIT PACKETS		
106430	104455					TRAP	C\$ERDF
106432	000030					.WORD	24
106434	064103					.WORD	MSG24
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						
6426	106466	000207			130\$:	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\$:	MOV	#CMDTY6,KEYWD1
6437	106504	013701	002372			MOV	P\$TYPE,R1
6438	106510	006301				ASL	R1
6439	106512	062701	003324			ADD	#MSGTAB,R1
6440	106516					PRINTF	#MSGPRM
	106516	012746	054345				
	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\$:	RTS	PC
6445							
6446							
6447							
6448	106576						
6449	106576	012737	000020	003734		ACTEXT:	
6450	106604	000207				MOV	#CEXIT,CFLAG
6451						RTS	PC
6452							
6453							
6454							
6455	106606						
6456	106606					CALL	EDPACK CBOADR,#ADRBUF,#6
	106606	012725	000006				
	106612	012725	002316				
	106616	013725	002370				
	106622	004437	070656				
	106626	004600					
6457	106630					P\$POP	P\$AERR
	106630	014537	003222				
6458	106634	105737	003222			TSTB	P\$AERR

```

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, R0
      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$TYPE
6513 107020          CALL    SELMSG CBOADR
        107020 013725 002370
        107024 004437 070656
        107030 025352
6514 107032 000423          BR       40$
6515 107034 022737 000000 003734 20$:  CMP     #CTARGET,CFLAG
6516 107042 001011          BNE     30$
6517 107044          PRINTF  #UNBOND
        107044 012746 053756
        107050 012746 000001
        107054 010600
        107056 104417
        107060 062706 000004
6518 107064 000406          BR       40$
6519 107066 105737 003223          TSTB   P$MERR
6520 107072 001003          BNE     40$
6521 107074 112737 177777 003221  MOVB   #-1,P$GDBD
6522 107102 000207          RTS     PC
6523
6524          ;ACTION ROUTINE TO CHECK FOR MORE INPUT AFTER MESSAGE TYPE HAS BEEN
6525          ;ALTERED
6526
6527 107104 004737 106470  ACTTYP: JSR     PC,ACTMSG
6528 107110 000207          RTS     PC
6529
6530          ;ACTION ROUTINE TO INPUT MESSAGE SIZE PARAMETER, CHECK TO SEE IF
6531          ;IT IS WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO
6532          ;SEE IF MORE INPUT EXISTA
6533
6534 107112 023727 003214 000040 ACTSIZE: CMP     P$NUM,#32.
6535 107120 003410          BLE     10$
6536 107122 022737 001001 003214  CMP     #513.,P$NUM
6537 107130 003404          BLE     10$
6538 107132 013737 003214 002374  MOV     P$NUM,P$SIZE
6539 107140 000410          BR      20$
6540 107142          10$:  PRINTF  #SIZLMT
        107142 012746 053575
        107146 012746 000001
        107152 010600
        107154 104417
        107156 062706 000004
6541 107162 004737 106470  20$:  JSR     PC,ACTMSG
6542 107166 000207          RTS     PC
6543
6544          ;ACTION ROUTINE TO INPUT COPIES PARAMETER, CHECK TO SEE IF IT IS
6545          ;WITHIN LEGAL LIMITS, CHANGE PARAMETER AND THEN RETURN TO SEE IF
6546          ;MORE INPUT PARAMETERS EXIST
6547
6548 107170 023727 003214 000000 ACTCPY: CMP     P$NUM,#0
6549 107176 003410          BLE     10$
6550 107200 022737 000400 003214  CMP     #256.,P$NUM
6551 107206 003404          BLE     10$
6552 107210 013737 003214 002376  MOV     P$NUM,P$CPYS
6553 107216 000410          BR      20$
6554 107220          10$:  PRINTF  #CPYLMT

```

```

;IF NOT OK, RETURN WITH ERROR FLAG SET
;SET MESSAGE TYPE
;PUT OPERATOR SELECTED STRING INTO BUFFER
        MOV     CBUADR,(R5)+
        JSR     R4,PREG14
        .WORD   SELMSG-ANCHOR
;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,RO
        TRAP   C$PNTF
        ADD    #4,SP
;RETURN
;IF P$MERR FLAG SET, UNBOUNDED STRING
;WAS ENTERED, ERROR ALREADY HANDLED
;SET ERROR FLAG AND RETURN
;RETURN
;CHECK FOR ADDITIONAL COMMANDS
;CHECK FOR VALID SIZE RANGE
;IF VALID CONTINUE
;SET MESSAGE SIZE
;PRINT SIZE LIMITS EXCEEDED MESSAGE
        MOV     #SIZLMT,-(SP)
        MOV     #1,-(SP)
        MOV     SP,RO
        TRAP   C$PNTF
        ADD    #4,SP
;CHECK FOR ADDITIONAL COMMANDS
;CHECK FOR VALID COPIES RANGE
;IF VALID, CONTINUE
;SET MESSAGE COPIES
;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			RTS	PC		
6557								
6558								
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					
	107330	012746	000001					
	107334	010600						
	107336	104417						
	107340	062706	000004					
6573	107344	000414				BR	30\$	;RETURN
6574	107346	005023		20\$:	CLR	(R3)+		;ELSE, CLEAR NODE FROM TABLE
6575	107350	005023			CLR	(R3)+		
6576	107352	005023			CLR	(R3)+		
6577	107354	005013			CLR	(R3)		
6578	107356				PRINTF	#ADRDEL		;PRINT NODE DELETED FROM TABLE MESSAGE
	107356	012746	054044					
	107362	012746	000001					
	107366	010600						
	107370	104417						
	107372	062706	000004					
6579	107376			30\$:	P\$POP	R2,R3		;RESTORE R2 AND R3
	107376	014502						
	107400	014503						
6580	107402	000207		40\$:	RTS	PC		;RETURN
6581								
6582								
6583								
6584	107404							
	107404	010325						
	107406	010225						
6585	107410	012703	000050			MOV	#TBLLN,R3	;SET INCR. COUNTER TO 40

```

6586 107414 012702 002412          MOV    @NODTBL,R2          ;MOVE NODE TABLE ADDRESS INTO R2
6587 107420 005022          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$PNTF
        107446 062706 000006          ADD    @6,SP
6591 107452 105037 003220          CLRB  P$NUF              ;CLEAR NOTNUF FLAG
6592 107456          P$POP R2,R3            ;RESTORE R2 AND R3
        107456 014502          MOV    -(R5),R2
        107460 014503          MOV    -(R5),R3
6593 107462 000207          RTS    PC
6594
6595          ;ACTION ROUTINE TO RUN SPECIFIED TEST
6596          ;
6597 107464          ;ACTRUN:
6598 107464 042737 100000 000000          BIC    @BIT15,FLAG      ; CLEAR OPERATOR FLAG
6599 107472 105037 003220          CLRB  P$NUF              ; 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          ;ACTION ROUTINE TO SET 'RUN ALL' FLAG
6621          ;
6622 107612 012737 000032 002312  ;ACTRNA: MOV    @CRNALL,KEYWD1   ; SET FLAG
6623 107620 000207          RTS    PC
6624 107622          RUNALL: CALL  DIRCOM    ; RUN LOOPDIRECT TEST
        107622 004437 070656          JSR    R4,PREG14
        107626 017372          .WORD DIRCOM-ANCHOR

```

6625	107630					P\$POP	R1			:	CHECK RESULTS	
	107630	014501									MOV	-(R5),R1
6626	107632	001415				BEQ	20\$			:	IF OK, BRANCH	
6627	107634	022701	000001			CMP	#1,R1			:	ELSE, WAS TABLE EMPTY?	
6628	107640	001410				BEQ	10\$			:	IF YES, DON'T PRINT ABORT MESSAGE	
6629	107642					PRINTS	#PASABT			:	ELSE ABORT TEST AND PRINT MESSAGE	
	107642	012746	060672								MOV	#PASABT,-(SP)
	107646	012746	000001								MOV	#1,-(SP)
	107652	010600									MOV	SP,RO
	107654	104416									TRAP	C\$PNTS
	107656	062706	000004								ADD	#4,SP
6630	107662	000137	110234		10\$:	JMP	100\$					
6631	107666	012737	002412	002410	20\$:	MOV	#NODTBL,SLOT			:	MOVE NODE TABLE ADDRESS TO SLOT	
6632	107674					CALL	FULSLT			:	FIND FIRST ENTRY	
	107674	004437	070656								JSR	R4,PREG14
	107700	025556									.WORD	FULSLT-ANCHOR
6633	107702	013701	002410			MOV	SLOT,R1			:	AND PUT TARGET ADDRESS INTO R1	
6634	107706	013737	002376	050326	30\$:	MOV	P\$CPYS,CPYCNT			:	SET UP LOOP FOR NO. OF COPIES	
6635	107714	062737	000010	002410		ADD	#10,SLOT			:	UPDATE SLOT	
6636	107722					CALL	FULSLT			:	GET NEXT ASSIST NODE FROM TABLE	
	107722	004437	070656								JSR	R4,PREG14
	107726	025556									.WORD	FULSLT-ANCHOR
6637	107730	013702	002410			MOV	SLOT,R2					
6638	107734	022737	177777	002410		CMP	#-1,SLOT			:	SEE IF AT END OF TABLE	
6639	107742	001517				BEQ	90\$			:	IF YES, BR	
6640	107744				40\$:	CALL	BLDFAS R1,SLOT			:	BUILD FULL ASSIST MESSAGE	
	107744	013725	002410								MOV	SLOT,(R5)
	107750	010125									MOV	R1,(R5)
	107752	004437	070656								JSR	R4,PREG14
	107756	005502									.WORD	BLDFAS-ANCHOR
6641	107760					CALL	XMIT #XMTDAT			:	TRANSMIT MESSAGE	
	107760	012725	000001								MOV	#XMTDAT,(R5)
	107764	004437	070656								JSR	R4,PREG14
	107770	002326									.WORD	XMIT-ANCHOR
6642	107772					P\$POP	R3			:	CHECK RESULTS	
	107772	014503									MOV	-(R5),R3
6643	107774	001404				BEQ	50\$			:	IF OK, CONTINUE	
6644	107776					ERRHRD	37,EMSG24			:	PRINT ERROR MESSAGE	
	107776	104456									TRAP	C\$ERRHRD
	110000	000045									.WORD	37
	110002	064103									.WORD	EMSG24
	110004	000000									.WORD	0
6645	110006				50\$:	CALL	BINHEX R1,#6,#STRBUF			:	PRINT ERROR MESSAGE	
	110006	012725	002324								MOV	#STRBUF,(R5)
	110012	012725	000006								MOV	#6,(R5)
	110016	010125									MOV	R1,(R5)
	110020	004437	070656								JSR	R4,PREG14
	110024	005150									.WORD	BINHEX-ANCHOR
6646	110026					CALL	BINHEX R2,#6,#STRBU1			:		
	110026	012725	002346								MOV	#STRBU1,(R5)
	110032	012725	000006								MOV	#6,(R5)
	110036	010225									MOV	R2,(R5)
	110040	004437	070656								JSR	R4,PREG14
	110044	005150									.WORD	BINHEX-ANCHOR
6647	110046					PRINTB	#TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1			:	ASSIST NODE	
	110046	012746	002346								MOV	#STRBU1,-(SP)
	110052	012746	061747								MOV	#ARGTY6,-(SP)

110056	012746	002324				MOV	#STRBUF, -(SP)
110062	012746	061757				MOV	#ARGTY7, -(SP)
110066	012746	060770				MOV	#TSTMS4, -(SP)
110072	012746	000005				MOV	#5, -(SP)
110076	010600					MOV	SP, R0
110100	104414					TRAP	C#PNTB
110102	062706	000014				ADD	#14, SP
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\$:	ERRHRD	28, MSG42, ERR3		; ELSE PRINT ERROR MESSAGE
	110120	104456				TRAP	C#ERRHRD
	110122	000034				.WORD	28
	110124	064764				.WORD	MSG42
	110126	070432				.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	C#PNTB
	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		; SEE IF END OF TABLE
6662	110224	022737	177777 002410	CMP	#-1, SLOT		; IF NO, CONTINUE ELSE, FINISHED
6663	110232	001225		BNE	30\$		
6664	110234		100\$:	RETURN			
	110234	000207				RTS	PC
6665							
6666							
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$: RETURN
        110264 000207          RTS      PC
6678 110266 005001          DIRCOM: CLR      R1          ; CLEAR RESULTS REGISTER
6679 110270 012737 002412 002410  MOV      #NODTBL,SLOT ; MOVE NODE TABLE ADDRESS TO SLOT
6680 110276          CALL     FULSLT        ; SEE IF TABLE EMPTY
        110276 004437 070656          JSR      R4,PREG14
        110302 025556          .WORD   FULSLT-ANCHOR
6681 110304 022737 177777 002410  CMP      #-1,SLOT      ; IF NO CONTINUE
6682 110312 001015          BNE      10$           ; ELSE, PRINT "TABLE EMPTY" MESSAGE
6683 110314          PRINTF #TABEMT,#NOD
        110314 012746 053415          MOV      #NOD,-(SP)
        110320 012746 053346          MOV      #TABEMT,-(SP)
        110324 012746 000002          MOV      #2,-(SP)
        110330 010600          MOV      SP,RO
        110332 104417          TRAP    C$PNTF
        110334 062706 000006          ADD      #6,SP
6684 110340 012701 000001          MOV      #1,R1          ; PUT 'TABLE EMPTY' INDICATOR IN R1
6685 110344 000547          BR      100$
6686 110346 012737 002412 002410 10$: MOV      #NODTBL,SLOT
6687 110354 013737 002376 050326 20$: MOV      P$CPYS,CPYCNT ; SET UP FOR NO. OF COPIES
6688 110362          CALL     FULSLT        ; GET NEXT NODE IN TABLE
        110362 004437 070656          JSR      R4,PREG14
        110366 025556          .WORD   FULSLT-ANCHOR
6689 110370 022737 177777 002410  CMP      #-1,SLOT      ; SEE IF AT END OF TABLE
6690 110376 001532          BEQ      100$         ; IF YES, EXIT
6691 110400          CALL     BINHEX SLOT,#6,#STRBUF ; PRINT ADDRESS BEING TESTED
        110400 012725 002324          MOV      #STRBUF,(R5)
        110404 012725 000006          MOV      #6,(R5)
        110410 013725 002410          MOV      SLOT,(R5)
        110414 004437 070656          JSR      R4,PREG14
        110420 005150          .WORD   BINHEX-ANCHOR
6692 110422          30$: PRINTB #TSTMS2,#DIRECT,#STRBUF ; NODE ADDRESS
        110422 012746 002324          MOV      #STRBUF,-(SP)
        110426 012746 061337          MOV      #DIRECT,-(SP)
        110432 012746 060735          MOV      #TSTMS2,-(SP)
        110436 012746 000003          MOV      #3,-(SP)
        110442 010600          MOV      SP,RO
        110444 104414          TRAP    C$PNTB
        110446 062706 000010          ADD      #10,SP
6693 110452 022737 000005 002312  CMP      #CPATRN,KEYWD1
6694 110460 001016          BNE      40$
6695 110462 013701 002372          MOV      P$TYPE,R1
6696 110466 006301          ASL     R1
6697 110470 062701 003324          ADD     #MSGTAB,R1
6698 110474          PRINTB #MESPAL,(R1)
        110474 011146          MOV      (R1),-(SP)
        110476 012746 061271          MOV      #MESPAL,-(SP)
        110502 012746 000002          MOV      #2,-(SP)
        110506 010600          MOV      SP,RO
        110510 104414          TRAP    C$PNTB
        110512 062706 000006          ADD     #6,SP
6699 110516          40$: CALL     BLDLD SLOT ; CALL BUILD LOOPDIRECT SUBROUTINE
        110516 013725 002410          MOV      SLOT,(R5)
        110522 004437 070656          JSR      R4,PREG14
        110526 005236          .WORD   BLDLD-ANCHOR

```

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			ERRHRD	26,MSG24	50\$:			; ELSE PRINT ERROR MESSAGE
	110546	104456							TRAP C\$ERHRD
	110550	000032							.WORD 26
	110552	064103							.WORD MSG24
	110554	000000							.WORD 0
6704	110556	000676		BR	20\$				; DO RECEIVE LOOP
6705	110560			CALL	RUNCOM	60\$:			JSR R4,PREG14
	110560	004437	070656						.WORD RUNCOM-ANCHOR
	110564	021104							
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,MSG34,ERR2				
	110572	104456							TRAP C\$ERHRD
	110574	000033							.WORD 27
	110576	064322							.WORD MSG34
	110600	070344							.WORD ERR2
6709	110602	012701	177777	MOV	#-1,R1				; PUT ERROR INDICATOR INTO R1
6710	110606	000410		BR	80\$				
6711	110610			PRINTB	#OK	70\$:			; 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	DEC	CPYCNT	80\$:			; 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	ADD	#10,SLOT	90\$:			; INCREMENT TO NEXT NODE TABLE ENTRY
6716	110662	000634		BR	20\$				
6717	110664			RETURN	R1	100\$:			
	110664	010125							MOV R1,(R5)+
	110666	000207							RTS PC
6718									
6719									; ACTION ROUTINE TO SET 'RUN LOOPPAIR' FLAG
6720									
6721	110670	012737	000033	ACTRNL:	MOV #CLUPPR,KEYWD1				; SET FLAG
6722	110676	000207	002312	RTS	PC				
6723									
6724	110700								
6725	110700								
	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	MOV	#NODTBL,SLOT				; MOVE NODE TABLE ADDRESS TO SLOT
6728	110720			CALL	FULSLT				; SEE IF TABLE EMPTY

	110720	004437	070656				JSR	R4,PREG14		
	110724	025556					.WORD	FULSLT-ANCHOR		
6729	110726	022737	177777	002410		CMP	#-1,SLOT		:	
6730	110734	001014				BNE	9%		: IF NO, CONTINUE	
6731	110736					PRINTF	@TABEMT,#NOD		: ELSE, PRINT "TABLE EMPTY" MESSAGE	
	110736	012746	053415						MOV	#NOD,-(SP)
	110742	012746	053346						MOV	@TABEMT,-(SP)
	110746	012746	000002						MOV	#2,-(SP)
	110752	010600							MOV	SP,RO
	110754	104417							TRAP	C\$PNTF
	110756	062706	000006						ADD	#6,SP
6732	110762	000137	111776			JMP	30%		:	
6733	110766	012737	002412	002410	9%:	MOV	#NODTBL,SLOT		: MOVE NODE TABLE ADDRESS TO SLOT	
6734	110774	013737	002376	050326	10%:	MOV	P\$CPYS,CPYCNT		: SET UP FOR NO. OF COPIES	
6735	111002					CALL	FULSLT		: GET NEXT NODE IN TABLE	
	111002	004437	070656						JSR	R4,PREG14
	111006	025556							.WORD	FULSLT-ANCHOR
6736	111010	022737	177777	002410		CMP	#-1,SLOT		: SEE IF AT END OF TABLE	
6737	111016	001002				BNE	11%		: IF YES, EXIT	
6738	111020	000137	111776			JMP	30%		:	
6739	111024	013701	002410		11%:	MOV	SLOT,R1		: MOVE SLOT TO R1	
6740	111030	126127	000007	000000		CMPB	7(R1),#CTARGET		: SEE IF TARGET NODE	
6741	111036	001422				BEQ	18%		: IF YES, BRANCH	
6742	111040				17%:	PRINTF	#EMSG32		: ELSE PRINT ERROR MESSAGE	
	111040	012746	064245						MOV	#EMSG32,-(SP)
	111044	012746	000001						MOV	#1,-(SP)
	111050	010600							MOV	SP,RO
	111052	104417							TRAP	C\$PNTF
	111054	062706	000004						ADD	#4,SP
6743	111060					PRINTF	#PASABT		:	
	111060	012746	060672						MOV	#PASABT,-(SP)
	111064	012746	000001						MOV	#1,-(SP)
	111070	010600							MOV	SP,RO
	111072	104417							TRAP	C\$PNTF
	111074	062706	000004						ADD	#4,SP
6744	111100	000137	111776			JMP	30%		: EXIT	
6745	111104	010102			18%:	MOV	R1,R2		: POINT R1 TO TARGET NODE	
6746	111106	062702	000010		112%:	ADD	#10,R2		: AND R2 TO ASSIST NODE	
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		: IS R2 POINTING TO AN ASSIST NODE?	
6753	111140	001337				BNE	17%		: IF NOT, ERROR, ELSE CONTINUE	
6754	111142				20%:	CALL	BLDAST R2,R1		: BUILD TRANSMIT ASSIST MESSAGE	
	111142	010125							MOV	R1,(R5)+
	111144	010225							MOV	R2,(R5)+
	111146	004437	070656						JSR	R4,PREG14
	111152	006030							.WORD	BLDAST-ANCHOR
6755	111154					CALL	XMIT		: TRANSMIT MESSAGE	
	111154	004437	070656						JSR	R4,PREG14
	111160	002326							.WORD	XMIT-ANCHOR
6756	111162					P\$POP	R4		: GET RESULTS, R2 = SUCCESS/FAILURE	
	111162	014504							MOV	-(R5),R4
6757	111164	001406				BEQ	22%		: IF OK, EXIT	
6758	111166				21%:	ERRHRD	26,EMSG24		: ELSE PRINT ERROR MESSAGE	

	111166	104456						TRAP	C\$ERHRD
	111170	000032						.WORD	26
	111172	064103						.WORD	EMSG24
	111174	000000						.WORD	0
6759	111176	000137	111712		JMP	28\$			
6760	111202			22\$:	CALL	BINHEX R1,#6,#STRBUF	; PRINT ERROR MESSAGE		
	111202	012725	002324					MOV	#STRBUF,(R5)+
	111206	012725	000006					MOV	#6,(R5)+
	111212	010125						MOV	R1,(R5)+
	111214	004437	070656					JSR	R4,PREG14
	111220	005150						.WORD	BINHEX-ANCHOR
6761	111222				CALL	BINHEX R2,#6,#STRBU1	;		
	111222	012725	002346					MOV	#STRBU1,(R5)+
	111226	012725	000006					MOV	#6,(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	#5,-(SP)
	111272	010600						MOV	SP,R0
	111274	104414						TRAP	C\$PNTB
	111276	062706	000014					ADD	#14,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,EMSG40,ERR3			
	111314	104456						TRAP	C\$ERHRD
	111316	000034						.WORD	28
	111320	064561						.WORD	EMSG40
	111322	070432						.WORD	ERR3
6767	111324	000410			BR	101\$			
6768	111326			23\$:	PRINTB	#OKRE			
	111326	012746	061046					MOV	#OKRE,-(SP)
	111332	012746	000001					MOV	#1,-(SP)
	111336	010600						MOV	SP,R0
	111340	104414						TRAP	C\$PNTB
	111342	062706	000004					ADD	#4,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,#6,#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				MOV	#STRBU1,(R5)+
	111416	012725				MOV	#6,(R5)+
	111422	010225				MOV	R2,(R5)+
	111424	004437				JSR	R4,PREG14
	111430	005150				.WORD	BINHEX-ANCHOR
6775	111432		PRINTB	#TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1	;	ASSIST	NODE *
	111432	012746				MOV	#STRBU1,-(SP)
	111436	012746				MOV	#ARGTY6,-(SP)
	111442	012746				MOV	#STRBUF,-(SP)
	111446	012746				MOV	#ARGTY7,-(SP)
	111452	012746				MOV	#TSTMS4,-(SP)
	111456	012746				MOV	#5,-(SP)
	111462	010600				MOV	SP,R0
	111464	104414				TRAP	C#PNTB
	111466	062706				ADD	#14,SP
6776	111472		CALL	RUNCOM	;	DO	RECEIVE LOOP
	111472	004437				.JSR	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#ERHRD
	111506	000034				.WORD	28
	111510	064662				.WORD	MSG41
	111512	070432				.WORD	ERR3
6780	111514	000410	BR	102#			
6781	111516		PRINTB	#OKTR			
	111516	012746				MOV	#OKTR,-(SP)
	111522	012746				MOV	#1,-(SP)
	111526	010600				MOV	SP,R0
	111530	104414				TRAP	C#PNTB
	111532	062706				ADD	#4,SP
6782	111536		102#:	CALL	BLDFAS R1,R2	;	BUILD FULL ASSIST MESSAGE
	111536	010225				MOV	R2,(R5)+
	111540	010125				MOV	R1,(R5)+
	111542	004437				JSR	R4,PREG14
	111546	005502				.WORD	BLDFAS-ANCHOR
6783	111550		CALL	XMIT	;	TRANSMIT	MESSAGE
	111550	004437				.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	JMP	21#			
6787	111566		26#:	CALL	BINHEX R1,#6,#STRBUF	;	PRINT ERROR MESSAGE
	111566	012725				MOV	#STRBUF,(R5)+
	111572	012725				MOV	#6,(R5)+
	111576	010125				MOV	R1,(R5)+
	111600	004437				JSR	R4,PREG14
	111604	005150				.WORD	BINHEX-ANCHOR

6788	111606		CALL	BINHEX	R2,#6,#STRBU1	;			
	111606	012725						MOV	#STRBU1,(R5)+
	111612	012725						MOV	#6,(R5)+
	111616	010225						MOV	R2,(R5)+
	111620	004437						JSR	R4,PREG14
	111624	005150						.WORD	BINHEX-ANCHOR
6789	111626		PRINTB	#TSTMS4,#ARGTY7,#STRBUF,#ARGTY6,#STRBU1	;	ASSIST NODE =		MOV	#STRBU1,-(SP)
	111626	012746						MOV	#ARGTY6,-(SP)
	111632	012746						MOV	#STRBUF,-(SP)
	111636	012746						MOV	#ARGTY7,-(SP)
	111642	012746						MOV	#TSTMS4,-(SP)
	111646	012746						MOV	#5,-(SP)
	111652	012746						MOV	SP,R0
	111656	010600						TRAP	C#PNTB
	111660	104414						ADD	#14,SP
	111662	062706							
6790	111666		CALL	RUNCOM		;	DO RECEIVE LOOP		
	111666	004437						JSR	R4,PREG14
	111672	021104						.WORD	RUNCOM-ANCHOR
6791	111674		P#POP	R3		;	CHECK RESULTS		
	111674	014503						MOV	-(R5),R3
6792	111676	001405	BEQ	28\$		;	IF NO ERRORS, CONT		
6793	111700		ERRHRD	28,MSG42,ERR3					
	111700	104456						TRAP	C#ERHRD
	111702	000034						.WORD	28
	111704	064764						.WORD	MSG42
	111706	070432						.WORD	ERR3
6794	111710	000410	BR	103\$					
6795	111712		28\$: PRINTB	#OKFU					
	111712	012746						MOV	#OKFU,-(SP)
	111716	012746						MOV	#1,-(SP)
	111722	010600						MOV	SP,R0
	111724	104414						TRAP	C#PNTB
	111726	062706						ADD	#4,SP
6796	111732	005337	103\$: DEC	CPYCNT		;	DECREMENT 'COPIES' COUNTER		
6797	111736	001402	BEQ	29\$		;	IF MORE TO DO, LOOP		
6798	111740	000137	JMP	20\$					
6799	111744		29\$: CALL	WRITES #2,R1,R2		;	ELSE,UPDATE SUMMARY TABLE		
	111744	010225						MOV	R2,(R5)+
	111746	010125						MOV	R1,(R5)+
	111750	012725						MOV	#2,(R5)+
	111754	004437						JSR	R4,PREG14
	111760	007176						.WORD	WRITES-ANCHOR
6800	111762	062702		ADD	#10,R2				
6801	111766	010237		MOV	R2,SLOT				
6802	111772	000137		JMP	10\$				
6803	111776		30\$: RETURN						
	111776	000207						RTS	PC
6804	112000	005737	RUNCOM: TST	RETRY5		;	SEE IF FAILED DUE TO EXCESSIVE COLLISIONS		
6805	112004	001120	BNE	30\$		;	IF YES, BR, ELSE CONT.		
6806	112006	012704	MOV	#TIMERS,R4		;	SET UP FOR 10 SECOND TIMEOUT		
6807	112012	012714	MOV	#10.,(R4)					
6808	112016		10\$: BREAK						
	112016	104422						TRAP	C#BRK
6809	112020	005714		TST	(R4)	;	SEE IF TIME HAS EXPIRED		
6810	112022	001517		BEQ	40\$	;	IF YES, BRANCH		
6811	112024			CALL	RECEVE	;	CHECK FOR ANSWER		

```

112024 004437 070656
112030 003044
6812 112032 P$POP R1 ; R2 HOLDS NO. OF BUFFERS RECEIVED
112032 014501 ; JSR R4,PREG14
; .WORD RECEVE-ANCHOR
; MOV -(R5),R1
6813 112034 001770 BEQ 10$ ; 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 MOV STAT2(R3),R1 ; GET RCV BYTE COUNT BITS 0-7
6818 112060 116302 000011 MOV 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$: MOV LOAD(R3),R3 ; POINT R3 TO MESSAGE BUFFER
6832 112164 066303 000016 ADD 16(R3),R3 ; POINT R3 TO DATA AFTER SKIP COUNT
6833 112170 062703 000030 ADD #30,R3 ; POINT R3 TO FIRST DATA BYTE
6834 112174 063737 002374 050040 ADD P$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 #COMPAR,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 RUNPAT: P$PUSH P$TYPE ; SAVE TYPE PARAMETER
      112346 005037 002372 CLR P$TYPE ; SET TYPE TO FIRST TYPE
6868 112352
      112352 004437 070656 10$: CALL DIRCOM ; SEND MESSAGES
      112356 017372 JSR R4,PREG14
      112360 .WORD DIRCOM-ANCHOR
6870 112360
      112360 014501 P$POP R1 ; GET RESULTS TO KEEP STACK IN ORDER
      112362 001403 BEQ 20$ ; IF OK, CONT
6871 112364 022701 000001 CMP #1,R1 ; ELSE, WAS TABLE EMPTY
6872 112370 001406 BEQ 30$ ; IF YES, RETURN
6873 112372 005237 002372 20$: INC P$TYPE ; SET TO NEXT TYPE
6874 112376 022737 000005 002372 CMP #5,P$TYPE ; SEE IF DONE ALL OF THEM
6875 112404 002362 BGE 10$ ; IF NOT, DO MORE
6876 112406
      112406 014537 002372 30$: P$POP P$TYPE ; RESTORE MESSAGE TYPE
      112412 RETURN MOV -(R5),P$TYPE
      112412 000207 RTS PC
6877
6878
6879
6880
6881
6882
6883
6884 112414 013701 002372 ACTSMS: MOV P$TYPE,R1 ;GET MESSAGE TYPE INTO R1
6885 112420 006301 ASL R1 ;MULTIPLY BY 2
6886 112422 062701 003324 ADD #MSGTAB,R1 ;ADD MESSAGE TABLE OFFSET
6887 112426 PRINTF #MSGPRM ;PRINT MESSAGE PARAMETER MESSAGE
      112426 012746 054345 MOV #MSGPRM,-(SP)
      112432 012746 000001 MOV #1,-(SP)
      112436 010600 MOV SP,R0
      112440 104417 TRAP C$PNTF
      112442 062706 000004 ADD #4,SP
6888 112446 PRINTF #MSG4,(R1),P$SIZE,P$CPYS ;PRINT PARAMETERS
      112446 013746 002376 MOV P$CPYS,-(SP)
      112452 013746 002374 MOV P$SIZE,-(SP)
      112456 011146 MOV (R1),-(SP)
      112460 012746 055002 MOV #MSG4,-(SP)
  
```



```

112464 012746 000004
112470 010600
112472 104417
112474 062706 000012
6889 112500 105037 003220
6890 112504 000207
6891
6892
6893
6894
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$CPYS ;RESET COPIES
6900 112530 PRINTF #CLRMSG ;PRINT MESSAGE PARAMETERS RESET MESSAGE
112530 012746 053432 MOV #CLRMSG,-(SP)
112534 012746 000001 MOV #1,-(SP)
112540 010600 MOV SP,RO
112542 104417 TRAP C$PNTF
112544 062706 000004 ADD #4,SP
6901 112550 PRINTF #MSG4,MSGTAB,P$SIZE,P$CPYS ;PRINT PARAMETERS
112550 013746 002376 MOV P$CPYS,-(SP)
112554 013746 002374 MOV P$SIZE,-(SP)
112560 013746 003324 MOV MSGTAB,-(SP)
112564 012746 055002 MOV #MSG4,-(SP)
112570 012746 000004 MOV #4,-(SP)
112574 010600 MOV SP,RO
112576 104417 TRAP C$PNTF
112600 062706 000012 ADD #12,SP
6902 112604 105037 003220 CLR B P$NNUF ;CLEAR NOTNUF FLAG
6903 112610 000207 RTS PC
6904
6905
6906
6907
6908
6909
6910 112612 ACTCNT:
6911 112612 P$PUSH R1,R2,R3 ; SAVE REGISTERS
112612 010325 MOV R3,(R5)+
112614 010225 MOV R2,(R5)+
112616 010125 MOV R1,(R5)+
6912 112620 012701 050144 MOV #CNTRS,R1 ; GET ADDRESS OF COUNTERS
6913 112624 012702 050044 MOV #UCB12,R2 ; GET ADDRESS OF COUNTER BUFFER
6914 112630 016103 000000 MOV C.SIZ(R1),R3 ; GET SIZE OF BUFFER
6915 112634 10$:
6916 112634 112122 MOV B (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
112642 014501 MOV -(R5),R1
112644 014502 MOV -(R5),R2
112646 014503 MOV -(R5),R3
6920 112650 CALL BINHEX #PHYADR,#6,#STRBUF ;GET ADDRESS INTO ASCII
112650 012725 002324 MOV #STRBUF,(R5)+
112654 012725 000006 MOV #6,(R5)+

```

Address	OpCode	Operand 1	Operand 2	Operand 3	Instruction	Comment
112660	012725	004010				
112664	004437	070656				
112670	005150					
6921 112672					PRINTF #CNTR00,#STRBUF	
112672	012746	002324				
112676	012746	062047				
112702	012746	000002				
112706	010600					
112710	104417					
112712	062706	000006				
6922 112716					PRINTF #CNTR01,UCB12+2	
112716	013746	050046				
112722	012746	062127				
112726	012746	000002				
112732	010600					
112734	104417					
112736	062706	000006				
6923 112742					CALL BINDEC #UCB12+4	
112742	012725	050050				
112746	004437	070656				
112752	007446					
6924 112754					PRINTF #CNTR02,#DECSTR	
112754	012746	100526				
112760	012746	062176				
112764	012746	000002				
112770	010600					
112772	104417					
112774	062706	000006				
6925 113000					CALL BINDEC #UCB12+10	
113000	012725	050054				
113004	004437	070656				
113010	007446					
6926 113012					PRINTF #CNTR03,#DECSTR	
113012	012746	100526				
113016	012746	062232				
113022	012746	000002				
113026	010600					
113030	104417					
113032	062706	000006				
6927 113036					PRINTF #CNTR04,UCB12+14	
113036	013746	050060				
113042	012746	062277				
113046	012746	000002				
113052	010600					
113054	104417					
113056	062706	000006				
6928 113062					PRINTF #CNTR05,UCB12+16	
113062	013746	050062				
113066	012746	062354				
113072	012746	000002				
113076	010600					
113100	104417					
113102	062706	000006				
6929 113106					CALL BINDEC #UCB12+20	
113106	012725	050064				
113112	004437	070656				
113116	007446					

```

MOV #PHYADR,(R5)+
JSR R4,PREG14
.WORD BINHEX-ANCHOR

MOV #STRBUF,-(SP)
MOV #CNTR00,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV UCB12+2,-(SP)
MOV #CNTR01,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV #UCB12+4,(R5)+
JSR R4,PREG14
.WORD BINDEC-ANCHOR

MOV #DECSTR,-(SP)
MOV #CNTR02,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV #UCB12+10,(R5)+
JSR R4,PREG14
.WORD BINDEC-ANCHOR

MOV #DECSTR,-(SP)
MOV #CNTR03,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV UCB12+14,-(SP)
MOV #CNTR04,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV UCB12+16,-(SP)
MOV #CNTR05,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

MOV #UCB12+20,(R5)+
JSR R4,PREG14
.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	C:PNTF
	113140	062706	000006			ADD	@6,SP
6931	113144			CALL	BINDEC @UCB12.24	MOV	@UCB12.24,(R5).
	113144	012725	050070			JSR	R4,PREG14
	113150	004437	070656			.WORD	BINDEC-ANCHOR
	113154	007446					
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	C:PNTF
	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	C:PNTF
	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	C:PNTF
	113246	062706	000006			ADD	@6,SP
6935	113252			CALL	BINDEC @UCB12.34	MOV	@UCB12.34,(R5).
	113252	012725	050100			JSR	R4,PREG14
	113256	004437	070656			.WORD	BINDEC-ANCHOR
	113262	007446					
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	C:PNTF
	113304	062706	000006			ADD	@6,SP
6937	113310			CALL	BINDEC @UCB12.40	MOV	@UCB12.40,(R5).
	113310	012725	050104			JSR	R4,PREG14
	113314	004437	070656			.WORD	BINDEC-ANCHOR
	113320	007446					
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	C:PNTF
	113342	062706	000006			ADD	@6,SP
6939	113346			CALL	BINDEC @UCB12.44	MOV	@UCB12.44,(R5).
	113346	012725	050110			JSR	R4,PREG14
	113352	004437	070656				

6940	113356	007446		PRINTF	@CNTR12,@DECSTR	.WORD	BINDEC-ANCHOR
	113360						MOV @DECSTR,-(SP)
	113360	012746	100526				MOV @CNTR12,-(SP)
	113364	012746	062766				MOV @2,-(SP)
	113370	012746	000002				MOV SP,R0
	113374	010600					TRAP C\$PNTF
	113376	104417					ADD @6,SP
	113400	062706	000006				
6941	113404			CALL	BINDEC @UCB12+50	MOV	@UCB12+50,(R5)+
	113404	012725	050114			JSR	R4,PREG14
	113410	004437	070656			.WORD	BINDEC-ANCHOR
	113414	007446					
6942	113416			PRINTF	@CNTR13,@DECSTR		
	113416	012746	100526				MOV @DECSTR,-(SP)
	113422	012746	063034				MOV @CNTR13,-(SP)
	113426	012746	000002				MOV @2,-(SP)
	113432	010600					MOV SP,R0
	113434	104417					TRAP C\$PNTF
	113436	062706	000006				ADD @6,SP
6943	113442			CALL	BINDEC @UCB12+54	MOV	@UCB12+54,(R5)+
	113442	012725	050120			JSR	R4,PREG14
	113446	004437	070656			.WORD	BINDEC-ANCHOR
	113452	007446					
6944	113454			PRINTF	@CNTR14,@DECSTR		
	113454	012746	100526				MOV @DECSTR,-(SP)
	113460	012746	063101				MOV @CNTR14,-(SP)
	113464	012746	000002				MOV @2,-(SP)
	113470	010600					MOV SP,R0
	113472	104417					TRAP C\$PNTF
	113474	062706	000006				ADD @6,SP
6945	113500			CALL	BINDEC @UCB12+60	MOV	@UCB12+60,(R5)+
	113500	012725	050124			JSR	R4,PREG14
	113504	004437	070656			.WORD	BINDEC-ANCHOR
	113510	007446					
6946	113512			PRINTF	@CNTR15,@DECSTR		
	113512	012746	100526				MOV @DECSTR,-(SP)
	113516	012746	063135				MOV @CNTR15,-(SP)
	113522	012746	000002				MOV @2,-(SP)
	113526	010600					MOV SP,R0
	113530	104417					TRAP C\$PNTF
	113532	062706	000006				ADD @6,SP
6947	113536			CALL	BINDEC @UCB12+64	MOV	@UCB12+64,(R5)+
	113536	012725	050130			JSR	R4,PREG14
	113542	004437	070656			.WORD	BINDEC-ANCHOR
	113546	007446					
6948	113550			PRINTF	@CNTR16,@DECSTR		
	113550	012746	100526				MOV @DECSTR,-(SP)
	113554	012746	063177				MOV @CNTR16,-(SP)
	113560	012746	000002				MOV @2,-(SP)
	113564	010600					MOV SP,R0
	113566	104417					TRAP C\$PNTF
	113570	062706	000006				ADD @6,SP
6949	113574			PRINTF	@CNTR17,UCB12+70		
	113574	013746	050134				MOV UCB12+70,-(SP)
	113600	012746	063245				MOV @CNTR17,-(SP)
	113604	012746	000002				MOV @2,-(SP)
	113610	010600					MOV SP,R0

	113612	104417					TRAP	C\$PNTF
	113614	062706	000006				ADD	#6,SP
6950	113620				PRINTF	#CNTR18,UCB12.72		
	113620	013746	050136				MOV	UCB12.72,-(SP)
	113624	012746	063320				MOV	#CNTR18,-(SP)
	113630	012746	000002				MOV	#2,-(SP)
	113634	010600					MOV	SP,RO
	113636	104417					TRAP	C\$PNTF
	113640	062706	000006				ADD	#6,SP
6951	113644				PRINTF	#CNTR19,UCB12.74		
	113644	013746	050140				MOV	UCB12.74,-(SP)
	113650	012746	063365				MOV	#CNTR19,-(SP)
	113654	012746	000002				MOV	#2,-(SP)
	113660	010600					MOV	SP,RO
	113662	104417					TRAP	C\$PNTF
	113664	062706	000006				ADD	#6,SP
6952	113670	105037	003220		CLRB	P\$NNUF		
6953	113674	000207			RTS	PC		
6954								
6955								
6956								
6957								
6958								
6959								
6960	113676	105037	003220		ACTSND:	CLRB	P\$NNUF	
6961	113702	012737	002412	002410		MOV	#NODTBL,SLOT	
6962	113710					CALL	FULSLT	
	113710	004437	070656					
	113714	025556						
6963	113716	022737	177777	002410		CMP	#-1,SLOT	
6964	113724	001437				BEQ	20\$	
6965	113726					PRINTF	#NTBHDR	
	113726	012746	053176					
	113732	012746	000001					
	113736	010600						
	113740	104417						
	113742	062706	000004					
6966	113746				10\$:	CALL	FULSLT	
	113746	004437	070656					
	113752	025556						
6967	113754	022737	177777	002410		CMP	#-1,SLOT	
6968	113762	001432				BEQ	30\$	
6969	113764					CALL	BINHEX SLOT,#6,#STRBUF	
	113764	012725	002324					
	113770	012725	000006					
	113774	013725	002410					
	114000	004437	070656					
	114004	005150						
6970	114006					CALL	PRTNOD	
	114006	004437	070656					
	114012	026040						
6971	114014	062737	000010	002410		ADD	#8,SLOT	
6972	114022	000751				BR	10\$	
6973	114024				20\$:	PRINTF	#TABEMT,#NOD	
	114024	012746	053415					
	114030	012746	053346					
	114034	012746	000002					

; ACTION ROUTINE TO PRINT OUT THE NODE TABLE

```

;MOVE NODE TABLE ADDRESS INTO SLOT
;SEE IF TABLE EMPTY
      JSR      R4,PREG14
      .WORD   FULSLT-ANCHOR
;IF YES, DON'T PRINT HEADER
;PRINT NODE TABLE HEADER
      MOV     #NTBHDR,-(SP)
      MOV     #1,-(SP)
      MOV     SP,RO
      TRAP   C$PNTF
      ADD     #4,SP
;FIND LOCATION IN TABLE WITH AN ADDRESS
      JSR     R4,PREG14
      .WORD   FULSLT-ANCHOR
;CHECK IF AT END OF TABLE
;IF YES, RETURN
;ELSE, PUT ASCII ADDRESS INTO BUFFER
      MOV     #STRBUF,(R5)
      MOV     #6,(R5)
      MOV     SLOT,(R5)
      JSR     R4,PREG14
      .WORD   BINHEX-ANCHOR
;PRINT NODE TABLE ENTRY
      JSR     R4,PREG14
      .WORD   PRTNOD-ANCHOR
;INCR. SLOT TO POINT TO NEXT TABLE ENTRY
;CONTINUE UNTIL ALL ENTRIES PRINTED
      MOV     #NOD,-(SP)
      MOV     #TABEMT,-(SP)
      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$NUM,R2 ;PUT NODE LOGICAL NUMBER INTO R2
6983 114060 006302 ASL R2 ;MULTIPLY BY 8
6984 114062 006302 ASL R2 ;NODE TABLE ADDRESS =
6985 114064 006302 ASL R2 ; (LOG. NO. X 8) + #NODTBL
6986 114066 062702 002412 ADD #NODTBL,R2 ;ADD OFFSET
6987 114072 005022 CLR (R2); ;CLEAR ENTRY (8 BYTES)
6988 114074 005022 CLR (R2);
6989 114076 005022 CLR (R2);
6990 114100 005012 CLR (R2);
6991 114102 P$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 114136 ACTSAV: P$PUSH R2,R3 ;SAVE R2 AND R3
114136 010325 MOV R3,(R5);
114140 010225 MOV R2,(R5);
7000 114142 012702 002412 MOV #NODTBL,R2 ;SET REGISTERS FOR COPYING
7001 114146 012703 002536 MOV #SAVTBL,R3 ;R2 = FROM, R3 = TO
7002 114152 PRINTF @UNSMMSG,@SAVED ;PRINT 'TABLE SAVED' MESSAGE
114152 012746 054324 MOV @SAVED,-(SP)
114156 012746 054265 MOV @UNSMMSG,-(SP)
114162 012746 000002 MOV @2,-(SP)
114166 010600 MOV SP,R0
114170 104417 TRAP C$PNTF
114172 062706 000006 ADD @6,SP
7003 114176 000137 114260 JMP SAVCOM
7004
7005 ;ACTION ROUTINE TO UNSAVE NODE TABLE
7006 ;
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
7012 114216 001002                  BNE    10$
7013 114220 000137 114272          JMP    QUIT
7014 114224 012703 002412          10$:  MOV    @NODTBL,R3
7015 114230 012702 002536          MOV    @SAVTBL,R2
7016 114234                  PRINTF @UNSMMSG,@RESTOR
114234 012746 054333
114240 012746 054265
114244 012746 000002
114250 010600
114252 104417
114254 062706 000006
7017 114260 012701 000050          SAVCOM: MOV   @TBLEN,R1
7018 114264 012223          10$:  MOV   (R2)+,(R3)+
7019 114266 005301          DEC   R1
7020 114270 001375          BNE   10$
7021 114272          QUIT:  P$POP R2,R3
114272 014502
114274 014503
7022 114276 105037 003220          CLRB  P$NNUF
7023 114302 000207          RTS   PC
7024
7025
7026          ; ACTION ROUTINE TO CLEAR SUMMARY TABLE
7027          ;
7028
7029 114304 105037 003220          ACTCSU: CLRB  P$NNUF
7030 114310          P$PUSH R2
114310 010225
7031 114312 012701 000132          MOV   @STBLN,R1
7032 114316 012702 002714          MOV   @STATBL,R2
7033 114322 005022          10$:  CLR   (R2)+
7034 114324 005301          DEC   R1
7035 114326 001375          BNE   10$
7036 114330          PRINTF @TABCLR,@SUMM
114330 012746 053422
114334 012746 054220
114340 012746 000002
114344 010600
114346 104417
114350 062706 000006
7037 114354          P$POP R2
114354 014502          ; AND RESTORE R2
7038 114356 000207          RTS   PC
7039
7040          ; ACTION ROUTINE TO CHECK FOR PASS DEFAULT VALUE
7041          ;
7042          ;
7043
7044 114360          ACTDFT:
7045 114360 121427 000040          10$:  CMPB  (R4),#40
7046 114364 001002          BNE   20$
7047 114366 005204          INC   R4
7048 114370 000773          BR    10$
7049 114372 121427 000000          20$:  CMPB  (R4),#0
7050 114376 001007          BNE   30$

```

MOV R2,(R5)+

```

;SET REGISTERS FOR COPYING
;R2 = FROM, R3 = TO
;PRINT 'TABLE RESTORED' MESSAGE
MOV @RESTOR,-(SP)
MOV @UNSMMSG,-(SP)
MOV @2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD @6,SP

```

```

;MOVE TABLE LENGTH TO R1
;MOVE WORD
;DECREMENT COUNTER
;IF MORE, LOOP
; ELSE, RESTORE COUNTERS
MOV -(R5),R2
MOV -(R5),R3
; CLEAR 'NOT ENOUGH' FLAG

```

```

;CLEAR 'NOT ENOUGH' COUNTER
;SAVE R2
MOV R2,(R5)+
;MOVE TABLE LENGTH TO R1
;MOVE SUMMARY TABLE ADDRESS TO R2
;CLEAR FIRST WORD
;SEE IF FINISHED
; IF NO, DO MORE
; ELSE, PRINT 'TABLE CLEARED' MESSAGE
MOV @SUMM,-(SP)
MOV @TABCLR,-(SP)
MOV @2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD @6,SP
; AND RESTORE R2
MOV -(R5),R2

```

```

;SEE IF SPACES
; IF NO, CONT.
; ELSE, POINT TO NEXT CHAR
; AND CHECK AGAIN
;SEE IF DEFAULT VALUE
; IF NO, BR

```

```

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      ACTUSF:
7062      P$PUSH R2      ; SAVE R2
114426 010225      MOV    R2,(R5)+      ; INITIALIZE R2 TO NODE TYPE 'TARGET'
7063 114430 005002      CLR    R2           ; OPEN FILE, NAME=ASCIZ STRING
7064 114432          OPEN   CBOADR
114432 013700 002370      MOV    CBOADR,R0
114436 104434          TRAP  C$OPEN
7065 114440          BCOMPLETE 10$      ; RETURN IF SUCCESSFUL
114440 103413          PRINTF #OPNERR,CBOADR ; ELSE PRINT "OPEN ERROR"
7066 114442          MOV    CBOADR,-(SP)
114442 013746 002370      MOV    #OPNERR,-(SP)
114446 012746 116072      MOV    #2,-(SP)
114452 012746 000002      MOV    SP,R0
114456 010600          TRAP  C$PNTF
114460 104417          ADD   #6,SP
114462 062706 000006      ; CLOSE FILE
7067 114466          TRAP  C$CLOS
114466 104435          ; READ A LINE AT A TIME
7068 114470          JSR   R4,PREG14
10$:  114470 004437 070656      .WORD  RDLIN-ANCHOR
114474 024520          ; SEE IF AN ERROR DURING READ
7069 114476 005737 116066      TST   BAD           ; BR ON ERROR TO LEAVE
7070 114502 001064          TST   EOFF          ; SEE IF EOF BEFORE PROCESS
7071 114504 005737 116070      BEQ   20$           ; IF VALID, PROCESS
7072 114510 001411          PRINTF #EOFFND      ; ELSE SAY 'END OF FILE' AND LEAVE
7073 114512          MOV    #EOFFND,-(SP)
114512 012746 116205      MOV    #1,-(SP)
114516 012746 000001      MOV    SP,R0
114522 010600          TRAP  C$PNTF
114524 104417          ADD   #4,SP
114526 062706 000004      ; PRINT LINE READ FROM FILE
7074 114532 000450          BR     40$
7075 114534          PRINTF #PLINE,#FILLIN
20$:  114534 012746 115542      MOV    #FILLIN,-(SP)
114540 012746 116154      MOV    #PLINE,-(SP)
114544 012746 000002      MOV    #2,-(SP)
114550 010600          MOV    SP,R0
114552 104417          TRAP  C$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

```





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  
7136  
7137  
7138  
7139  
7140  
7141  
7142  
7143  
7144  
7145  
7146  
7147

114710  
114710 005237 003202  
114714  
114714 013701 003750  
114720 062701 000010  
114724  
114724 104422  
114726 020137 003750  
114732 002016  
114734 005001  
114736 012703 004032  
114742 012704 047674  
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:  
INC RSPFLG ; INDICATE WE ARE IN LISTEN MODE  
10$: MOV TIMMIN,R1 ; GET CURRENT NUMBER OF MINUTES  
ADD #8.,R1 ; AND ADD 9 MINUTES TO IT  
20$: BREAK ; ALLOW OPERATOR TO RETAKE CONTROL  
; TRAP C$BRK  
CMP R1,TIMMIN ; HAS NINE MINUTES GONE BY?  
BGE 30$ ; 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						
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,MSG58	:	ELSE, REPORT ERROR
	115026	104456					TRAP C\$ERHRD
	115030	000072					.WORD 58
	115032	066211					.WORD MSG58
	115034	000000					.WORD 0
7162	115036	000555		BR	110\$	:	DO IT ALL AGAIN
7163	115040						
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	PROT00,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,MSG59	:	ELSE, REPORT UNINVITED REPLY RCVD
	115074	104456					TRAP C\$ERHRD
	115076	000073					.WORD 59
	115100	066267					.WORD MSG59
	115102	000000					.WORD 0
7174	115104	000532		BR	110\$	:	DO IT ALL AGAIN
7175	115106						
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	MOVB	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$:
7198 115206 012224      MOV      (R2)+,(R4)+      ; START FILLING IN THE DATA BUFFER
7199 115210 005203      INC      R3                ; BUMP WORDCOUNT
7200 115212 001375      BNE      60$              ; DO IT UNTIL THE PACKET HAS BEEN COPIED
7201 115214 000434      BR       90$              ; GO XMIT THE PACKET
7202 115216                                70$:
7203 115216 022763 000005 000020  CMP      #IDFUNC,RIFUNC(R3) ; IS THIS A REQUEST ID CONSOLE TYPE?
7204 115224 001062      BNE      110$            ; IF NOT, THROW IT OUT
7205 115226 012704 047674      MOV      #IDTDAT,R4      ; POINT TO ID TRANSMIT DATA
7206 115232 016364 000006 000000  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$:
7222 115306      CALL    XMIT      #XMTDAT ; TRANSMIT THE PACKET
115306 012725 000001      MOV      #XMTDAT,(R5)+  ;
115312 004437 070656      JSR      R4,PREG14      ;
115316 002326      .WORD   XMIT-ANCHOR
7223 115320      P$POP   R2                ; GET STATUS
115320 014502      MOV      -(R5),R2      ;
7224 115322 001404      BEQ      100$           ; BRANCH IF OKAY
7225 115324      ERRHRD  62,EMSG62
115324 104456      TRAP    C$ERHRD
115326 000076      .WORD   62
115330 066451      .WORD   EMSG62
115332 000000      .WORD   0
7226 115334                                100$:
7227 115334      CALL    GETXNX #1,#XRGNXT ; UPDATE XMIT RING POINTERS
115334 012725 024224      MOV      #XRGNXT,(R5)+  ;
115340 012725 000001      MOV      #1,(R5)+      ;
115344 004437 070656      JSR      R4,PREG14      ;
115350 006534      .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 PNT
R
7233 115372                                110$:
7234 115372      CALL    GETRNX #1,#RRGNXT ; UPDATE RECEIVE RING POINTERS
115372 012725 024226      MOV      #RRGNXT,(R5)+  ;
115376 012725 000001      MOV      #1,(R5)+      ;
115402 004437 070656      JSR      R4,PREG14      ;
115406 006450      .WORD   GETRNX-ANCHOR

```

7235 115410 000137 114724  
7236

JMP 20\$

; 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>..<0><0><0>.....???"
7250 ; SO I MADE ASSUMPTION THAT SINCE SEE "0-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, 0-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 10%: GETBYT (R2) ;GO GET A BYTE FROM INPUT FILE
7268 115430 104426 TRAP C$GETB
7269 115432 110012 MOV RO,(R2)
7270 115434 30%: BCOMPLETE 30% ;BR IF READ-BYTE SUCESSFUL
7271 115434 103414 BCS 30%
7272 115436 20%: PRINTF #RDERR ; ELSE PRINT "READ-ERROR"
7273 115436 012746 116126 MOV #RDERR,-(SP)
7274 115442 012746 000001 MOV #1,-(SP)
7275 115446 010600 MOV SP,RO
7276 115450 104417 TRAP C$PNTF
7277 115452 062706 000004 ADD #4,SP
7278 115456 012737 177777 116066 MOV #-1,BAD ; SET BAD-TRY FLAG AND LEAVE
7279 115464 000416 BR 50%
7280 115466 122712 000015 30%: CMPB #15,(R2) ;IS THE CHARACTER A <CR>
7281 115472 001756 BEQ 10% ; BR IF YES (GO BACK TO GET <LF>)
7282 115474 122712 000012 CMPB #12,(R2) ;IS THE CHARACTER A <LF>
7283 115500 001410 BEQ 50% ; BR IF YES (TERMINATE AND LEAVE)
7284 115502 122712 000040 CMPB #40,(R2) ;IS IT A "EOF" (END-OF-FILE(TEXT))
7285 115506 101002 BHI 40% ; (EOF=ANY NON-CHAR>37 EXCEPT CR,LF)
7286 115510 005202 INC R2 ; BR IF YES
7287 115512 000746 BR 10% ; IF NO, LEAVE CHAR IN BUFFER
7288 115514 012737 177777 116070 40%: MOV #-1,EOFF ; AND GO GET MORE CHARS
7289 115522 105012 50%: CLRB (R2) ;IF YES, TERMINATE INPUT BUFF
7290 115524 RETURN ; AND SET EOF-FLAG
    
```

RTS PC

115524 000207

7287							
7288	115526				FILENM:	.BLKB	12.
7289	115542				FILLIN:	.BLKB	132.
7290	115746				MATCH:	.BLKB	80.
7291	116066	000000			BAD:	.WORD	0
7292	116070	000000			EJFF:	.WORD	0
7293							

7294	116072	045	116	045	OPNERR:	.ASCIZ	/%N%A?UNABLE TO OPEN "%T%A"?/
------	--------	-----	-----	-----	---------	--------	-------------------------------

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	RDERR:	.ASCIZ	/%N%A?FILE READ ERROR?/
------	--------	-----	-----	-----	--------	--------	-------------------------

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

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	EJFFND:	.ASCIZ	/%N%A?END-OF-FILE FOUND, FILE READ/
------	--------	-----	-----	-----	---------	--------	-------------------------------------

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

```

:---+
:
: 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 - NONE
7312      ; CALLING PROCEEDURE - CALL SELMSG P1 ;INPUT ADDRESS OF ASCII STRING
7313      ; REGISTER USAGE - R1 CONTAINS ADDRESS OF INPUT STRING
7314      ; R2 CONTAINS ADDRESS OF OUTPUT STRING
7315      ;
7316      ;---
7317
7318 116246 SELMSG: P$POP R1 ;PUT ADDRESS OF OPR. SEL ASCII STRING INTO R1
      116246 014501 ; MOV -(R5),R1
7319 116250 012702 003632 MOV #OPSLBF,R2 ;PUT ADDRESS OF OUTPUT BUFFER INTO R2
7320 116254 005003 CLR R3 ;CLEAR CHARACTER COUNTER
7321 116256 105711 10$: TSTB (R1) ;CHECK FOR END OF STRING
7322 116260 001404 BEQ 20$ ;GO TO 20$ IF END
7323 116262 112122 MOV#B (R1)+,(R2)+ ;ELSE, MOVE BYTE TO OUTPUT BUFFER
7324 116264 005203 INC R3 ;COUNT NUMBER OF CHARACTERS IN INPUT BUFFER
7325 116266 000137 116256 JMP 10$ ;GO DO MORE CHARACTERS
7326 116272 112712 000000 20$: MOV#B #0,(R2) ;PUT ZERO AT END OF OUTPUT BUFFER
7327 116276 010337 003356 MOV R3,MSG6C ;STORE NUMBER OF CHARACTERS FOR USE IN BUF. BUILDING
7328 116302 RETURN
      116302 000207 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
      116312 014501 MOV -(R5),R1
7349 116314 001403 BEQ 10$ ;IF NOT FULL BR TO 10$
7350 116316 P$PUSH # -1 ;ELSE PUT FULL INDICATION ON STACK
      116316 012725 177777 MOV # -1,(R5)+
7351 116322 000416 BR 30$ ;RETURN
7352 116324 012703 000006 10$: MOV #6,R3 ;SET INCR. COUNTER TO 6 (BYTES)
7353 116330 013701 002410 MOV SLOT,R1 ;MOV ADDRESS OF AVAILABLE SLOT TO R1
7354 116334 012702 002316 MOV #ADRBUF,R2 ;MOV ADDRESS OF NODE ADDRESS TO R2
7355 116340 112221 20$: MOV#B (R2)+,(R1)+ ;MOV BYTE OF ADDRESS
7356 116342 005303 DEC R3 ;DECR. COUNTER
7357 116344 001375 BNE 20$ ;CONTINUE UNTIL 6 BYTES TRANSFERED
7358 116346 005201 INC R1 ;SET POINTER TO NODE TYPE LOCATION
7359 116350 113711 002406 MOV#B NODTY,(R1) ;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      : FINDSL      FIND EMPTY SLOT IN NODE TABLE
7365      :
7366      : INPUTS      NONE
7367      : EXPLICIT OUTPUTS  NONE
7368      : IMPLICIT OUTPUTS  THE ADDRESS OF THE FIRST AVAILABLE SLOT IN THE
7369      :      NODE TABLE WILL BE LOCATED IN SLOT. THE
7370      :      PARAMETER STACK WILL CONTAIN -1 IF THE NODE
7371      :      TABLE IS FULL AND 0 IF AN EMPTY SLOT WAS FOUND
7372      :
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      : FULSLT      FULL SLOT ROUTINE
7393      :
7394      : THIS ROUTINE FINDS A LOCATION IN THE TABLE WHERE A NODE PHYSICAL
7395      : ADDRESS EXISTS. IT IS USED WHEN PRINTING OUT THE NODE TABLE.
7396      :
7397      : INPUTS      NONE
7398      : EXPLICIT OUTPUTS  NONE
7399      : IMPLICIT OUTPUTS  THE LOCATION SLOT WILL CONTAIN THE PHYSICAL
7400      :      ADDRESS OF A NODE TABLE ENTRY. SLOT WILL
7401      :      CONTAIN -1 WHEN POINTING TO THE END OF THE
7402      :      NODE TABLE
7403      :
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      40$:  RETURN          ;RETURN
      116516 000207
8420      ; ~~~~~ RTS      PC
8421      ; ~~~~~
8422 116520      CHECK:
8423 116520 010146      MOV     R1,-(SP)
8424 116522 010346      MOV     R3,-(SP)
8425 116524 010446      MOV     R4,-(SP)
8426 116526 010546      MOV     R5,-(SP)
8427
8428 116530 005005      CLR     R5
8429 116532 012701 177776      MOV     #-2.,R1
8430 116536
8431 116536 062701 000002 3$:  ADD     #2.,R1
8432 116542 022761 000000 002412  CMP     #0.,NODTBL(R1)
8433 116550 001772      BEQ     3$
8434
8435 116552 062701 000006      ADD     #6.,R1
8436 116556 022761 000000 002412  CMP     #0.,NODTBL(R1)
8437 116564 001401      BEQ     5$
8438 116566 000413      BR      7$
8439 116570
8440 116570 016104 002412 5$:  MOV     NODTBL(R1),R4
8441 116574 022761 000400 002412  CMP     #400,NODTBL(R1)
8442 116602 001013      BNE     10$
8443
8444 116604 062705 000001      ADD     #1,R5
8445 116610 022705 000002      CMP     #2,R5
8446 116614 001007      BNE     12$
8447 116616
8448 116616 005061 002412 7$:  CLR     NODTBL(R1)
8449 116622      ERRSOFT 69,MSG69      ; REPORT IT
      116622 104457      TRAP   C$ERSOFT
      116624 000105      .WORD 69
      116626 067204      .WORD MSG69
      116630 000000      .WORD 0
8450 116632 10$:  CLR     R5
8451 116632 005005
8452 116634 12$:
8453 116634 062701 000002      ADD     #2.,R1
8454 116640 022761 177777 002412  CMP     #-1,NODTBL(R1)
8455 116646 001407      BEQ     EXIT1
8456 116650 022761 000000 002412  CMP     #0.,NODTBL(R1)
8457 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  
116676 000207

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

RTS PC

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

; ~~~~~  
; ~~~~~

```
---+
: 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

```
---+
: PRTNOD 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 PRTNOD
```

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

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

```

116754 062706 000006
7505 116760 013702 002410      MOV     SLOT,R2      ;MOVE SLOT ADDRESS INTO R2
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              PRINTF @LOGNAM,R2   ;PRINT LOGICAL NAME
              116776 010246
              117000 012746 053154
              117004 012746 000002
              117010 010600
              117012 104417
              117014 062706 000006
7511 117020 013702 002410      MOV     SLOT,R2      ;SEE IF TARGET OR ASSIST NODE
7512 117024 062702 000007      ADD     @7,R2        ; INFO CONTAINED IN 7TH BYTE OF ENTRY
7513 117030 111203              MOV     (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$:  MOV     @ARGTY6,KEYWD2 ; MOVE 'ASSIST' INTO KEYWD2
7519 117056              20$:  PRINTF @NODTYP,KEYWD2 ;PRINT NODE TYPE
              117056 013746 002314
              117062 012746 053167
              117066 012746 000002
              117072 010600
              117074 104417
              117076 062706 000006
7520 117102              RETURN      ;RETURN
              117102 000207
7521 117104              EXIT     TST
              117104 104432
              117106 000002
              117110
              117110
              117110 104401
7537
7538
7540
7541
7542

```

```

;*****
;   INSERT LOCAL STORAGE THAT IS USED ONLY
;   DURING THIS TEST.
;*****
;*****
;   INSERT MESSAGES THAT ARE USED ONLY
;   DURING THIS TEST.
;*****
.EVEN
ENDTST
L10015: TRAP C$ETST
;*****
;   BEGIN THE REMAINING TESTS ON NEW PAGES.
;*****

```

```

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      .WORD L10016-L$HARD/2
7560      117112
7561      117114      L$HARD:
7562
7563      : *****
7564      : INSERT HARDWARE PARAMETER INTERPRETIVE CODE HERE.  THIS CODE
7565      : IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR
7566      : DEVICE INFORMATION TO PUT IN THE P-TABLE.  THIS CODE IS USED
7567      : IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
7568      : USED IN THIS SECTION ARE "GPRMD", "GPRMA" AND "GPRML".
7569      : *****
7570      GPRMA  ASKCSR,0,0,160000,177776,YES      ; GET CSR ADDRESS
7571      .WORD  T$CODE
7572      .WORD  ASKCSR
7573      .WORD  T$LOLIM
7574      .WORD  T$HILIM
7575      GPRMA  ASKVEC,2,0,0,776,YES      ; GET VECTOR ADDRESS
7576      .WORD  T$CODE
7577      .WORD  ASKVEC
7578      .WORD  T$LOLIM
7579      .WORD  T$HILIM
7580      GPRMD  ASKPRI,4,0,340,0,7,YES      ; GET PRIORITY LEVEL
7581      .WORD  T$CODE
7582      .WORD  ASKPRI
7583      .WORD  340
7584      .WORD  T$LOLIM
7585      .WORD  T$HILIM
7586
7587      ENDHRD
7588
7589      .EVEN
7590      L10016:
7591
7592      : *****
7593      : INSERT MESSAGES THAT ARE USED ONLY
7594      : DURING THE HARDWARE PARAMETER CODING SECTION.
7595      : *****
7596
7597      117146      127      110      101  ASKCSR: .ASCIZ  /WHAT IS THE PCSRO ADDRESS?/
7598      117151      124      040      111
7599      117154      123      040      124
7600      117157      110      105      040
7601      117162      120      103      123
7602      117165      122      117      040
7603      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      ;**
7592      ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7593      ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
7594      ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7595      ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
7596      ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7597      ; WITH THE OPERATOR.
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
7609      .EVEN
7610
7611      ENDSFT
7612      117274
117274      .EVEN
L10017:

7613
7614
7615      ;*****
7616      ;
7617      ; INSERT MESSAGES THAT ARE USED ONLY
7618      ; DURING THE SOFTWARE PARAMETER CODING SECTION.
7619      ;*****
7620
7621      $PATCH::
7622      117274      .BLKW  5      ;; N.M.  (CHANGED FROM 100 TO 5)
7623      117274
7624
7625      ;*****
7626      ;
7627      ; THIS IS A PATCH AREA THAT SHOULD BE INCLUDED IN ALL DIAGNOSTICS.
7628      ; ADJUST THE SIZE TO FIT YOUR OWN PREFERENCES.
7629      ;*****
7630
7631      LASTAD
7632      117306
117306      000000      .EVEN
117310      000000      .WORD  0
117312      .WORD  0
7633      117312      L$LAST::
7634      .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

```



## SYMBOL TABLE

ACTALP	106676	B	= 000010	CCLNAD	= 000004 G	CMPER2	067677	C#CEFG	= 000045
ACTBLD	104440	BAD	116066	CCLNAL	= 000010 G	CMPSTR	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	CNDAL	= 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	CITTST	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#GPLO	= 000030
ACTSAV	114136	BIT4	= 000020 G	CLIEXI	= 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	CLI#PM	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
ACT1AL	106726	BLDREQ	077224 G	CLKHZ	003744	CSAVE	= 000006	C#RDBU	= 000007
ADDMUL	= 000001 G	BMPCNT	074632	CLKINT	070530 G	CSAVR4	= 000014	C#REFG	= 000047
ADR	= 000020 G	BMPEXI	075356	CLKSET	070504	CSHCTR	= 000002 G	C#RESE	= 000033
ADRBUF	002316	BMPMUL	075164	CLKVEC	003742	CSHMSG	= 000034	C#REVI	= 000003
ADRDEL	054044	BMPRCV	= 000000 G	CLMSG	053432	C#SIZE	= 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	CTARGET	= 000000 G	C#TPRI	= 000013
ARGTY3	061721	BUILD	= 000003	CMDTY4	061633	CUNSAV	= 000041	C#COLL	= 000074 G
ARGTY4	061732	CADERR	053022	CMDTY5	061640	CUNSVF	= 000045	C#CRC	= 000001 G
ARGTY5	061743	CADRER	052746	CMDTY6	061644	CZEROS	= 000020	C#FRAM	= 000002 G
ARGTY6	061747	CALPHA	= 000016	CMDTY7	061654	C#AU	= 000052	C#LCAR	= 000002 G
ARGTY7	061757	CASIST	= 000001 G	CMDTY8	061661	C#AUTO	= 000061	C#MLN	= 000004 G
ASKCSR	117146	CBOADR	002370	CMDTY9	061667	C#BRK	= 000022	C#MREC	= 000010 G
ASKPRI	117235	CCCITT	= 000023	CMPADR	116700	C#BSEG	= 000004	C#MXMT	= 000040 G
ASKVEC	117201	CCITT	= 000005 G	CMPBUF	050334 G	C#BSUB	= 000002	C#OPEN	= 000010 G
ASSEMB	= 000010	CCLMSG	= 000035	CMRPER1	067602			C#PREC	= 000004 G

C.PXMD= 000054 G	EMSG37 064451	FAFCT4= 000050 G	G\$RADO= 000020	ILLADR 002656
C.PXMT= 000034 G	EMSG38 064515	FASIST 050454 G	G\$XFER= 000004	INIADR= 000000 G
C.PXM2= 000050 G	EMSG4 003532 G	FASKIP= 000016 G	G\$YES = 000010	INICLN 103344
C.PXM3= 000044 G	EMSG40 064561	FATFLG 050252 G	HDMSG1 055046	INIXI 103354
C.RDAT= 000020 G	EMSG41 064662	FILENM 115526	HEADER= 000016 G	INIT 102126
C.RERB= 000014 G	EMSG42 064764	FILLIN 115542	HELP = 000001	INIT1 102154
C.RERR= 000016 G	EMSG43 065062	FINDSL 116362	HELP1 055120	ISR = 000100 G
C.RLEX= 000032 G	EMSG44 065141	FLAG = 000000 G	HELP10 056141	IXE = 004000 G
C.RLIN= 000030 G	EMSG45 065237	FLAGWO= 000000 G	HELP11 056232	I\$AU = 000041
C.RMDB= 000024 G	EMSG46 065303	FLAG1 050250 G	HELP12 056435	I\$AUTO= 000041
C.RTRY= 000001 G	EMSG47 065366	FORWRD= 000002 G	HELP13 056542	I\$CLN = 000041
C.RUNT= 000010 G	EMSG48 065446	FRDADR= 000004 G	HELP14 056641	I\$DU = 000041
C.SECs= 000002 G	EMSG5 003632 G	FREMEM 050024 G	HELP15 056740	I\$HRD = 000041
C.SMRT= 000004 G	EMSG50 065544	FRESIZ 050022 G	HELP16 057043	I\$INIT= 000041
C.SIZ = 000000 G	EMSG51 065600	FULAST 061353	HELP17 057132	I\$MOD = 000041
C.XABB= 000070 G	EMSG52 065633	FULSLT 116452	HELP18 057235	I\$MSG = 000041
C.XABT= 000072 G	EMSG53 065721	F\$AU = 000015	HELP19 057305	I\$PROT= 000040
C.XDAT= 000060 G	EMSG54 065773	F\$AUTO= 000020	HELP2 055221	I\$PTAB= 000041
C.XMDB= 000064 G	EMSG55 066063	F\$BGN = 000040	HELP20 057410	I\$PWR = 000041
COALT = 000022	EMSG56 066116	F\$CLEA= 000007	HELP21 057466	I\$RPT = 000041
C1ALT = 000021	EMSG57 066152	F\$DU = 000016	HELP22 057551	I\$SEG = 000041
DATCMP 077734 G	EMSG58 066211	F\$END = 000041	HELP23 057652	I\$SETU= 000041
DECSTR 100526 G	EMSG59 066267	F\$HARD= 000004	HELP24 057752	I\$SFT = 000041
DEPADR 003774 G	EMSG60 066344	F\$HW = 000013	HELP25 060102	I\$SRV = 000041
DESC = 000002 G	EMSG61 066412	F\$INIT= 000006	HELP26 060166	I\$SUB = 000041
DESTIN= 000000 G	EMSG62 066451	F\$JMP = 000050	HELP27 060272	I\$TST = 000041
DFPTBL 002170 G	EMSG63 066526	F\$MOD = 000000	HELP28 060374	J\$JMP = 000167
DIAGMC= 000000	EMSG64 066615	F\$MSG = 000011	HELP29 060513	KEYWD1 002312
DIRCOM 110266	EMSG65 066705	F\$PROT= 000021	HELP3 055314	KEYWD2 002314
DIRECT 061337	EMSG66 066777	F\$PWR = 000017	HELP30 060563	KILMUL = 000002 G
DNIFLG 050262 G	EMSG67 067071	F\$RPT = 000012	HELP4 055365	LAPCNT 050336
EDPACK 075474	EMSG68 067121	F\$SEG = 000003	HELP5 055436	LCLKEN= 000100 G
EF.CON= 000036 G	EMSG69 067204	F\$SOFT= 000005	HELP6 055536	LDADR1= 000022 G
EF.NEW= 000035 G	ENTRND 116304	F\$SRV = 000010	HELP7 055651	LDADR2= 000032 G
EF.PWR= 000034 G	EOFF 116070	F\$SUB = 000002	HELP8 055762	LDFCT1= 000020 G
EF.RES= 000037 G	EOFFND 116205	F\$SW = 000014	HELP9 56052	LDFCT2= 000030 G
EF.STA= 000040 G	ERRBLK 052102 G	F\$TEST= 000001	HEXBIN 075724	LDRESP 052310
EL = 001000 G	ERRFLG 050270 G	GETCL 103544	HEXC 076022	LDSKIP= 000016 G
EMSG0 003526 G	ERRMSG 052100 G	GETCOM 077442	HIADD = 000002 G	LENGTH 061506
EMSG01 063436	ERRNBR 052076 G	GETRNX 077344 G	HIRCV = 000006 G	LF = 000012 G
EMSG02 063467	ERROR 071002 G	GETXNX 077430 G	HIXMT = 000012 G	LGERMS 067745
EMSG03 063527	ERRTYP 052074 G	G\$CNT0= 000200	HLPEND 003322	LOADD = 000004 G
EMSG04 063563	ERR1 070306 G	G\$DELM= 000372	HLPTAB 003224	LOCDST 075360
EMSG08 063605	ERR2 070344 G	G\$DISP= 000003	HLP115 056330	LOE = 040000 G
EMSG1 003527 G	ERR3 070432 G	G\$EXCP= 000400	HN 075700	LOGDEL 054132
EMSG10 063670	EVL = 000004 G	G\$HILI= 000002	HOE = 100000 G	LOGNAM 053154
EMSG11 063735	EXIT = 000011	G\$LOLI= 000001	HXFORM 075566	LOPDIR 050346 G
EMSG14 063774	EXIT1 116666	G\$NO = 000000	HXN 075564	LORCV = 000004 G
EMSG2 003530 G	E\$END = 002100	G\$OFFS= 000400	IBE = 010000 G	LORTGT 004040
EMSG22 064050	E\$LOAD= 000035	G\$OFSI= 000376	IDENT = 000010	LOT = 000010 G
EMSG24 064103	FAADR1= 000022 G	G\$PRMA= 000001	IDFUNC= 000005 G	LOXMT = 000010 G
EMSG25 064157	FAADR2= 000032 G	G\$PRMD= 000002	IDTDT 047674	LOXTGT 004060
EMSG3 003531 G	FAADR3= 000042 G	G\$PRML= 000000	IDTSIZ 047770	LPADR 004024 G
EMSG32 064245	FAADR4= 000052 G	G\$RADA= 000140	IDU = 000040 G	LSRTRY 002402
EMSG34 064322	FAFCT1= 000020 G	G\$RADB= 000000	IER = 020000 G	LST 076042
EMSG35 064367	FAFCT2= 000030 G	G\$RADD= 000040	ILADMS 052565	LUPAIR 061326
EMSG36 064424	FAFCT3= 000040 G	G\$RADL= 000120	ILADM1 052651	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\$DU	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	MEOM	= 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	NODO	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	SLFTST	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\$NNUF	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	036674 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	SUMMRY =	000007
N141\$	051376	O\$AU =	000000	QNAPR1	050012 G	RRNGTB	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	RBUF8	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	SELMSG	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	TBLEN =	000050 G
N186\$	052020	PRI02 =	000100 G	RETRY	061462	SIADDR =	000042 G	TEMP	050314 G
N190\$	052044	PRI03 =	000140 G	RETRY5	050274 G	SIDEV =	000053 G	TEMP1	050316 G
N20\$	050622	PRI04 =	000200 G	RIFUNC =	000020 G	SIECO =	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\$LTND=	000001	T1	103544 G	VLOBYT=	000007 G	WRTEXI	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	VOVF =	000000 G	XMIT	073222 G
TRVDEC	101072	T\$SEGL=	177777	UNSMG	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\$\$AU =	010014	VDISC =	000014 G	VRUNT =	000013 G	XRG001	024574 G
TSTMS1	060715	T\$\$AUT=	010011	VECTOR=	000014 G	VRUSED=	000016 G	XRING	024236 G
TSTMS2	060735	T\$\$CLE=	010012	VELOOP=	000011 G	VSANIT=	000012 G	XRINGH	024252
TSTMS3	060755	T\$\$DU =	010013	VEOM =	000015 G	VSHORT=	000003 G	XRNGTB	024212 G
TSTMS4	060770	T\$\$HAR=	010016	VFAIL =	000010 G	VTDR =	000000 G	X\$	000171
T\$ARGC=	000002	T\$\$HW =	010000	VFERRS=	000016 G	VVALID=	000017 G	X\$ALWA=	000000
T\$CODE=	002032	T\$\$INI=	010010	VFLAST=	000017 G	VXERRS=	000016 G	X\$FALS=	000040
T\$ERRN=	000105	T\$\$MSG=	010004	VFRAM =	000002 G	VXLAST=	000015 G	X\$OFFS=	000400
T\$EXCP=	000000	T\$\$PRO=	010007	VFUSED=	000016 G	VXMTIN=	000007 G	X\$TRUE=	000020
T\$FLAG=	000040	T\$\$RPT=	010006	VHIBYT=	000006 G	VXSETP=	000014 G	ZEROS =	000002 G
T\$GMAN=	000000	T\$\$SOF=	010017	VILoop=	000010 G	VXUSED=	000016 G	ZROALT=	000004 G
T\$HILI=	000007	T\$\$SRV=	010005	VINTEN=	000006 G	WAIT	070720 G	\$PATCH	117274 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=SV34R.MLB,CVNIAA.P11