

DMR11
DMC11

DMR/C11 DCLT
CZCLKB0

AH-F593B-MC
FICHE 1 OF 1

OCT 1981
COPYRIGHT © 80-81
MADE IN USA



A large grid of approximately 15 columns and 15 rows of small, dense data tables. Each cell in the grid contains a small table with multiple columns and rows of text and numbers. The text is too small to read clearly, but the layout is consistent across the entire page. The data appears to be organized in a structured, repetitive format, possibly representing a large dataset or a series of related measurements.

2207
2208

.TITLE CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST

.REM 8

IDENTIFICATION

PRODUCT CODE: AC-F591B-MC
PRODUCT NAME: CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
PRODUCT DATE: 26-OCT-81
MAINTAINER: MERRIMACK DIAGNOSTIC ENGINEERING
AUTHOR: BRUCE LUHRS - BRUCE RIBOLINI

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) 1980,1981 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

REVISION HISTORY:

<u>REV</u>	<u>DATE</u>	<u>AUTHOR</u>	<u>REASON</u>
A	23-APR-80	BRUCE LUHRS BRUCE RIBOLINI	ORIGINAL ISSUE, DCLT FOR THE DMC OR DMR-11
B	15-JUL-81	ERNIE COOPER	ADD 'MODEM/NO MODEM' COMMAND ADD 'SET EXPECT=TRANSMIT' COMMAND ADD 'EXIT' COMMAND ADD 'RPT >' COMMAND ADD PASSWORD AND ID ON DOWNLINE LOAD ADD TX / EXPECT MESSAGE TOTAL CHECK UPDATE DOCUMENTATION

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 - RESTRICTIONS
- 2.0 OPERATING INSTRUCTIONS
 - 2.1 COMMANDS
 - 2.2 SWITCHES
 - 2.3 FLAGS
 - 2.4 HARDWARE QUESTIONS
 - 2.5 DATA COMM. LINK TEST COMMANDS
 - 2.5.1 MESSAGE COMMANDS
 - 2.5.2 STATISTICAL COMMANDS
 - 2.5.3 RUN COMMANDS
 - 2.5.4 DEFAULTS
 - 2.5.5 PRINT COMMANDS
 - 2.5.6 MISC COMMANDS
 - 2.6 QUICK STARTUP PROCEDURE
- 3.0 ERROR INFORMATION
 - 3.1 TYPES OF ERROR MESSAGES
 - 3.2 SPECIFIC ERROR MESSAGES
 - 3.2.1 COMMAND LINE INTERPRETER ERRORS
 - 3.2.2 DCLT ERRORS
 - 3.2.3 DEVICE ERRORS
- 4.0 PERFORMANCE AND PROGRESS REPORTS
 - 4.1 PRINTING EVENT LOG
 - 4.2 OPERATOR STATUS MESSAGES
 - 4.3 PRINTING DMR,DMC-11 BASE TABLE
 - 4.3.1 PRINTING ERROR COUNTER LOCATIONS
 - 4.3.2 PRINTING ENTIRE BASE TABLE
 - 4.3.3 PRINTING SINGLE LOCATION
- 5.0 DEVICE INFORMATION TABLES

6.0 MODE AND MESSAGE DESCRIPTIONS

6.1 MODE DESCRIPTIONS

- 6.1.1 TRANSMIT MODE
- 6.1.2 RECEIVE MODE
- 6.1.3 PASSIVE MODE
- 6.1.4 ACTIVE MODE
- 6.1.5 DOWN-LINE LOAD MODE
- 6.1.6 TALK MODE
- 6.1.7 LISTEN MODE
- 6.1.8 MAINTENANCE MODE

6.2 MESSAGE DESCRIPTIONS

7.0 OTHER INFORMATION

- 7.1 INTERFACING TO AN "ITEP" NODE
- 7.2 TROUBLESHOOTING HINTS

- 7.2.1 INTERNAL LOOP AT EACH NODE
- 7.2.2 TRANSMIT ON ONE NODE-RECEIVE ON THE OTHER
- 7.2.3 ONE NODE ACTIVE-THE OTHER NODE PASSIVE
- 7.2.4 BOTH NODES ACTIVE
- 7.2.5 TALK AND LISTEN MODES FOR COMMUNICATIONS

7.3 EXAMPLES OF COMMANDS

- 7.3.1 MESSAGES COMMANDS
- 7.3.2 STATISTICAL COMMANDS
- 7.3.3 RUN COMMANDS
- 7.3.4 PRINT COMMANDS
- 7.3.5 EXIT COMMAND

7.4 THINGS TO WATCH OUT FOR

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

THIS DCLT (DATA COMMUNICATION LINK TEST) PROGRAM IS MEANT TO PROVIDE FIELD SERVICE WITH A TOOL TO MAINTAIN DMR/DMC-11 TO DMR/DMC-11 AND OTHER (POINT TO POINT) DDCMP SUPPORTED COMMUNICATION LINKS. THIS DCLT PROGRAM WILL PROVIDE THE COVERAGE NECESSARY TO DETECT FAILURES TO THE COMPUTER EQUIPMENT, THE COMMUNICATION LINK, OR THE MODEM.

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 (CHOUS?.SEQ WHERE ? IS REV. LEVEL OF THE 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 DMR/DMC-11 DCLT PROGRAM, THE FOLLOWING MINIMUM HARDWARE IS REQUIRED:

- A PDP-11 CPU
- MINIMUM OF 24K WORDS OF MEMORY
- A WORKING, LINE OR REAL-TIME CLOCK
- A CONSOLE TERMINAL
- ANY XXDP+ SUPPORTED LOAD MEDIA
- ONE OF THESE DMR-11 OR DMC-11 CONFIGURATIONS:
 - DMC11-AL - LOCAL MICROPROCESSOR
 - DMC11-AR - REMOTE MICROPROCESSOR
 - DMC11-DA - E.I.A. LINE UNIT
 - DMC11-FA - CCITT V.35 LINE UNIT
 - DMC11-MA - 1M BPS LINE UNIT
 - DMC11-MD - 56K BPS LINE UNIT

 - DMR11-AA - E.I.A. (RS 232/423)
 - DMR11-AB - CCITT V.35
 - DMR11-AC - LOCAL
 - DMR11-AE - E.I.A. (RS 422)

IF DOWN-LINE-LOADING A DMC-11 SATELLITE, THE SATELLITE END REQUIRES:
M9301-YJ/M9312 - BOOTSTRAP MODULE

1.3 RELATED DOCUMENTS AND STANDARDS

- XXDP+ USER'S MANUAL (CHOUS?.SEQ WHERE ? IS THE REV. LEVEL OF

THE MANUAL - "C" IS THE CURRENT REV.).

1.4 DIAGNOSTIC HIERARCY PREREQUISITES

THE GOAL OF THE DATA COMM. LINK TEST PROGRAM IS TO TEST THE COMMUNICATION LINK AND THEREFORE ASSUMES THAT THE CPU'S, CLOCKS, AND DMR OR DMC-11'S AT EACH END OF THE LINK HAVE ALREADY BEEN TESTED.

IF NO LINE OR REAL-TIME CLOCK IS FOUND, THE PROGRAM WILL CONTINUE BUT ANY OF THE PROGRAM THAT TIMES THE DEVICE WILL HANG IF THE DEVICE TIMES OUT. ALSO, THE EVENT LOG WILL CONTAIN A ZERO EVENT TIME FOR ALL EVENTS LOGGED.

IT IS NOT THE INTENTION OF A DATA COMM. LINK TEST PROGRAM TO TEST THE DMR OR DMC-11, BUT TO TEST THE COMMUNICATION LINK TO WHICH THEY ARE CONNECTED.

SOME OF THE DIAGNOSTICS THAT COULD BE RUN IF THE DMC-11 OR DMR-11 LOOKS BAD:

DMR: CZDMIXX DMR-11 FCTNL DIAG
CZDMPXX M8207 STATIC DIAG #1
CZDMQXX M8207 STATIC DIAG #2
CZDMRXX M8203 STATIC DIAG #1
CZDMSXX M8203 STATIC DIAG #2

DMC: CZDMCXX BSC W/R MICRO-PROC TST
CZDMEXX DDCMP MDLN UNIT TST
CZDMGXX DMC-11 CROM + JMUP TEST
MD-11-DZDMHXX DMC-11 FREE RUNNING TEST

XX = LATEST REVISION

1.5 ASSUMPTIONS - RESTRICTIONS

IT IS ASSUMED THAT THE COMMUNICATIONS DEVICE (DMC OR DMR-11) HAS BEEN TESTED USING THE PREREQUISTE 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 DCLT.

BECAUSE THE DMC-11 AND DMR-11 SUPPORT DDCMP OPERATION IN THE FIRMWARE, THE PDP-11 D.C.L.T. PROGRAM IS UNABLE TO CONTROL OR KNOW EXACTLY WHAT IS BEING TRANSMITTED AT ANY GIVEN TIME. ALL DATA MESSAGES ARE ENCLOSED IN A DDCMP ENVELOPE AND THERE MAY ALSO BE CONTROL MESSAGES (AKS, NAKS,.....) BEING TRANSMITTED. BECAUSE OF THIS PLEASE BEWARE IF IF YOU ARE SCOPING DATA. -----

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. LIST EXAMPLE - /UNITS:0:5:10-12 USE UNITS 0,5,10,11,12 (UNIT NUMBERS = 0-63)

EXAMPLE OF SWITCH USAGE:

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

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

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

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

2.3 FLAGS

FLAGS ARE USED TO SET UP CERTAIN OPERATIONAL PARAMETERS SUCH AS LOOPING ON ERROR. ALL FLAGS ARE CLEARED AT STARTUP AND REMAIN CLEARED UNTIL EXPLICITLY SET USING THE FLAGS SWITCH. FLAGS ARE ALSO CLEARED AFTER A START COMMAND UNLESS SET USING THE FLAG SWITCH. THE ZFLAGS COMMAND MAY ALSO BE USED TO CLEAR ALL FLAGS. WITH THE EXCEPTION OF THE START AND ZFLAGS COMMANDS, NO COMMANDS AFFECT THE STATE OF THE FLAGS; THEY REMAIN SET OR CLEARED AS SPECIFIED BY THE LAST FLAG SWITCH.

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

IDR
ADR
LOT
EVL

STATISTICAL REPORTING)
INHIBIT PROGRAM DROPPING OF UNITS
EXECUTE AUTODROP CODE
LOOP ON TEST
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).

THE DMR/DMC-11 DATA COMM. LINK TEST PROGRAM WILL NOT USE MORE THAN
ONE UNIT. FOR THE DMC/DMR-11, THE HARDWARE INFORMATION REQUESTED
WILL BE:

UNITS (D) ? 1<CR>

UNIT 0

FULL DUPLEX OPERATION : (L) Y ?

DMR,DMC-11 CSR ADDRESS : (0) 160170 ?

INTERRUPT VECTOR ADDRESS: (0) 300 ?

INTERRUPT PRIORITY : (0) 5 ?

DEVICE OPTION TYPE : (0=DMC, 5=DMR-DMC MODE ,7=DMR) (0) 0 ?

2.5 DATA COMM. LINK TEST COMMANDS

THE 'DCLT>' COMMAND LEVEL FOLLOWS THE ANSWERING OF THE HARDWARE P-TABLE QUESTIONS. THESE COMMANDS CAN BE TYPED WHEN THE 'DCLT> (A) ?' PROMPT IS PRINTED.

MESSAGE COMMANDS AVAILABLE:

YOU ONLY HAVE TO TYPE ENOUGH CHARACTERS TO UNIQUELY SPECIFY A COMMAND.

THE COMMAND LINE IS INTERPRETED FROM LEFT TO RIGHT. THEREFORE, IF A QUALIFIER ON THE COMMAND LINE IS RELATED OR EFFECTS A QUALIFIER TO THE LEFT ON THE COMMAND LINE, THE QUALIFIER FARTHEREST TO THE RIGHT TAKES PRECEDENCE SINCE IT IS INTERPRETED LAST. (I.E. IF /CHECK..... .../NOCHECK APPEAR ON THE SAME LINE, NOCHECK WILL BE INDICATED IN THE PARAMETERS WORD.)

REFER TO SECTION 6.0 FOR A DESCRIPTION OF THE DIFFERENT MODES OF OPERATION AND THE TYPES OF MESSAGES AVAILABLE.

2.5.1 MESSAGE COMMANDS

COMMAND

CLEAR EXPECTLIST

CLEAR TRANSMITLIST

SET EXPECTMSG=TYPE/QUAL

WHERE: 'TYPE' IS:
=ONES
=ZEROES
=1ALT
=OALT
=ITEP
=CCITT
=ALPHA
='A-Z,0-9,SPACES OR TABS IN QUOTES'

WHERE THE OPTIONAL 'QUAL' IS:

DESCRIPTION

ZEROES THE EXPECTLIST (000'S)
AND THEN PUTS DEFAULT ITEP MSG
IN SO NOT REALLY EMPTY

ZEROES TRANSMITLIST (000'S)
AND THEN PUTS DEFAULT ITEP MSG
IN SO NOT REALLY EMPTY

DEFINE A MESSAGE TO BE PUT ON
THE EXPECTED LIST

/SIZE=NNN MAKE THE MESSAGE 'NNN' BYTES
LONG. (DEFAULT VALUE IS
SIZE OF MESSAGE SPEC'D BY
OPERATOR OR DEFAULTS.)
/COPY=NN COPY THIS MESSAGE INTO THE
BUFFER 'NN' TIMES (DEFAULT
IS 0 = PUT THE MESSAGE IN
ONLY ONCE)

NOTE: SET'S ADD MESSAGES TO THE LIST IN THE ORDER THEY'RE
DEFINED. 'NNN' IS A DECIMAL NUMBER. THE FIRST SET
OVERWRITES THE DEFAULT ITEP MESSAGE PLACED THERE BY
INITIALIZATION OR A 'CLEAR' COMMAND.

SEE SECTION 6.2 FOR A DESCRIPTION OF THE PRE-DEFINED
MESSAGES THAT ARE AVAILABLE. (ZEROS,ONES ...)

SET EXPECTLIST=TRANSMITLIST MAKES A COPY OF THE TRANSMIT
LIST IN THE EXPECT LIST.
SET TRANSMITMSG=TYPE/QUAL DEFINE A MESSAGE TO BE PUT ON
THE TRANSMIT LIST
(SEE DESCRIPT FOR SET EXP)
SHOW EXPECTLIST LISTS THE MESSAGE SIZE AND TYPE
FOR THE MESSAGES IN THE
EXPECT LIST
SHOW TRANSMITLIST LISTS THE MESSAGE SIZE AND TYPE
FOR THE MESSAGES IN THE
TRANSMIT LIST

2.5.2 STATISTICAL COMMANDS

(COMMAND)

DESCRIPTION

PRINT

TAKES THE OPERATOR TO THE
REPORT LEVEL. FROM HERE
YOU CAN EXAMINE THE EVENT
LOG OR BASE TABLE.

DUMP SSSSSS-EEEEEE/B

PRINTS THE CONTENTS OF THE
MEMORY LOCATIONS BETWEEN
OCTAL ADDRESSES 'SSSSSS' AND
'EEEEEE' WHERE 'SSSSSS' IS
THE START ADDRESS AND
'-EEEE' IS THE END ADDRESS.

WHERE '/B' IS OPTIONAL:
DEFAULT IS PRINT WORDS
'/B' CAUSES PRINT BYTES

IF '-EEEEEE' IS NOT SPECIFIED
THEN THE CONTENTS OF 'SSSSSS'
IS PRINTED IN WORD FORMAT.

NOTE: THE DUMP COMMAND IS USEFUL FOR EXAMINING
MESSAGE DATA. STARTING ADDRESSES CAN
BE FOUND BY LOOKING IN THE EVENT LOG.

2.5.3 RUN COMMAND

COMMAND

DESCRIPTION

RUN MODE=MTYPE/QUAL

STARTS DCLT EXECUTING IN THE
MODE SPECIFIED

NOTE: MODE=ACTIVE IS NOT DEFAULT, A MODE=MTYPE MUST BE TYPED
----- EACH TIME A RUN IS TYPED

WHERE THE 'MTYPE' IS ANY ONE OF THE FOLLOWING:

=ACTIVE	(FORCES /NOECHO ,NO LOOPING)
=PASSIVE	(FORCES NO LOOPING)
=RECEIVE	(FORCES /NOECHO ,NO LOOPING)
=LISTEN	(FORCES /NOECHO ,NO LOOPING, /NOCHECK)
=TRANSMIT	(FORCES /NOECHO ,NO LOOPING, /NOCHECK)
=TALK	(FORCES /NOECHO ,NO LOOPING, /NOCHECK)
=DOWNLINELOAD	(FORCES /NOECHO ,NO LOOPING, /NOCHECK,

(FORCING NO LOOPING MEANS IT MUST BE
SPECIFIED AS A QUALIFIER ANY TIME ITS
DESIRED, THERE IS NO DEFAULT)

AND OPTIONAL 'QUAL' IS ANY COMBINATION OF THE FOLLOWING:

/CHECK/NOCHECK ENABLES/DISABLES CHECKING OF RECEIVED
DATA AGAINST THE EXPECTED DATA

NOTE: IF BOTH NODES IN ACTIVE AND '/NOCHECK' IS USED,
----- END-OF-PASS IS DEFINED AS RECEIVING 1 MESSAGE
AND COMPLETING THE TRANSMIT LIST. WITH NO DATA
CHECKING, THERE IS NO WAY FOR DCLT TO KNOW HOW
MANY MESSAGES IT SHOULD EXPECT TO RECEIVE.

/STATUS/NOSTATUS ENABLES/DISABLES PRINTING OF PROGRAM
STATUS MESSAGES TO THE OPERATOR

/ECHO/NOECHO ENABLES/DISABLES THE RETRANSMISSION OF
THE DATA RECEIVED IN PASSIVE MODE.
(IGNORED IN MODES OTHER THAN PASSIVE)

/MODEM/NOMODEM ENABLES/DISABLES THE REPORTING OF MODEM STATUS
INTERRUPT CHANGES.
NOTE: THIS SWITCH CAUSES NO ACTION IN THIS DCLT
PROGRAM BUT IT IS INCLUDED BECAUSE IT IS USED
IN OTHER DCLT PROGRAMS.

/LOOP=LTYPE SPECIFIES WHICH, IF ANY, TYPE OF
MAINTENANCE LOOPBACK IS BEING USED.
(IGNORED IN MODES OTHER THAN ACTIVE)
MUST BE SPECIFIED EACH TIME ELSE NO
LOOP IS USED.

'LTYPE' IS:
=INTERNALTL
=CABLE
=LOCALMODEM (DMR IN DMR MODE AND RS449 MODEMS ONLY.
CAUSES A 'WRITE MODEM' TO BE DONE TO SET UP
LOCAL-LOOPBACK (MAINT1) . ALSO CALLED
ANALOG-LOOPBACK.

=REMOTEMODEM (DMR IN DMR MODE AND RS449 MODEMS ONLY.
CAUSES A 'WRITE MODEM' TO BE DONE TO SET UP
REMOTE-LOOPBACK (MAINT2) . ALSO CALLED
DIGITAL-LOOPBACK.

/PASS=NN SPECIFIES NUMBER OF ITERATIONS TO MAKE BEFORE
END-OF-PASS. DEFAULT VALUE OF 1
WILL BE USED ON ANY RUN THAT A /PASS=N
IS NOT ADDED TO THE 'RUN ...' COMMAND.
IF A '-1' IS TYPED, THEN THE PROGRAM
RUN UNTIL A ^C IS TYPED.

NOTE: SEE SECTION 6.1 FOR A DESCRIPTION
----- OF THE 'RUN MODES' AND 'LOOP MODES'

2.5.4 DEFAULTS -----

IF NO 'SET'S' THEN THE DEFAULT IS SAME AS IF TYPED:
SET TRANSMITMSG=ITEP/SIZE=58/COPY=0
SET EXPECTMSG=ITEP/SIZE=58/COPY=0

THE DEFAULT COPY AND SIZE FOR EACH OF THE MESSAGE TYPES:
ONES - /SIZE=64/COPY=0
ZEROES - /SIZE=64/COPY=0
OALT - /SIZE=64/COPY=0
1ALT - /SIZE=64/COPY=0
CCITT - /SIZE=64/COPY=0
ALPHA - /SIZE=65/COPY=0
ITEP - /SIZE=58/COPY=0
OPER. SPEC'D - /SIZE=LENGTH-OF-TEXT-TYPED-BETWEEN-QUOTES/COPY=0

FOR THE RUN COMMAND THE DEFAULTS ARE:

RUN MODE=ACTIVE/NOSTATUS/CHECK/NOECHO/PASS=1

NOTE: MODE=ACTIVE IS NOT DEFAULT, A MODE=MTYPE MUST BE TYPED
----- EACH TIME A RUN IS TYPED

IF THE DCLT PROGRAM IS RUN IN UNATTENDED MODE (UAM FLAG=1 OR CHAINED),
THE DEFAULTS ARE AS IF THESE SETUP AND RUN COMMANDS WERE TYPED:

SET TRANS=ITEP
SET EXPECT=ITEP
RUN MODE=ACTIVE/LOOP=INTERNAL/NOSTAT/CHECK/PASS=1

OTHER NOTES:

^C
<CR>
'RUBOUT'

ALWAYS RETURNS YOU TO 'DR>' (THE SUPERVISOR)
IS SEEN AS A COMMAND TERMINATOR
DELETE LAST CHAR. TYPED IN COMMAND STRING

2.5.5 PRINT

THE PRINT COMMAND TAKES YOU A LEVEL BELOW DCLT> CALLED REPORT.
THE COMMANDS AVAILABLE IN RPT> ARE ...

COMMAND

DESCRIPTION

HELP OR ?

PRINT HELP INFORMATION FOR RPT>

LOG

PRINTS THE DCLT EVENT LOG.

BASE/FULL

PRINTS ENTIRE BASE TABLE.

BASE/ERROR

PRINTS ONLY ERROR COUNTERS IN
BASE TABLE.

BASE/OFFSET=NNN

PRINTS SINGLE LOCATION IN BASE
TABLE AS SPECIFIED BY OFFSET.

EXIT

RETURNS YOU TO THE LEVEL THAT
YOU ENTERED FROM. (DCLT> OR DR>)

2.5.6 MISC COMMANDS

COMMAND

DESCRIPTION

EXIT

FROM THE DCLT> LEVEL RETURNS YOU
TO DR>.

HELP OR ?

PRINTS HELP INFORMATION.

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. THE NUMBER OF UNITS THAT CAN DCLT CAN USE IS ALWAYS '1'.

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS. THESE DEFAULTS ARE DESCRIBED IN SECTION 2.3.

7. AFTER THE 'DCLT> (A) ?' PROMPT, TYPE
'RUN MOD=ACTIVE<CR>'

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING THE DEFAULT TRANSMIT AND EXPECTED MESSAGES. THE DEFAULT PASS COUNT AND 'RUN' QUALIFIERS ARE ALSO BEING USED. THESE DEFAULTS ARE DESCRIBED IN SECTION 2.5.3.

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 "IBE" 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", "IBE" OR "IXE" 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

3.2.1 COMMAND LINE INTERPRETER ERRORS:

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.
?INCMPLTE CMD?	A REQUIRD PART OF A COMMAND WAS LEFT OUT.
?NUM 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.

? 'LOOP' VALID ONLY IN ACTIVE? THE '/LOOP=..' SWITCH WAS TYPED IN A RUN COMMAND BUT THE MODE WAS NOT SET TO ACTIVE. MAINTENANCE LOOP IS ONLY POSSIBLE IF THE MODE OF OPERATION IS ACTIVE.

? 'ECHO' VALID ONLY IN PASSIVE? THE '/ECHO' SWITCH WAS TYPED IN A RUN COMMAND BUT THE MODE WAS NOT SET TO PASSIVE. ECHOING OF RECEIVED DATA IS ONLY POSSIBLE IF THE MODE OF OPERATION IS PASSIVE.

? ILL CHR- 'A-Z,0-9,SP,TAB' ONLY? A CHARACTER TYPED WITHIN QUOTES WHEN TRYING TO DEFINE THE CONTENTS OF A TRANSMIT OR EXPECT MESSAGE WAS NOT A 'A-Z,0-9,SPACE OR TAB'. RETYPE THE COMMAND WITH ONLY THESE CHARACTERS BETWEEN QUOTES.

? 'SIZE=0' NOT VALID? A MESSAGE ZERO BYTES LONG CAN NOT BE BUILT. RETYPE THE COMMAND WITH A '/SIZE=NNN'. IF NO '/SIZE=' IS TYPED A DEFAULT SIZE WILL BE USED.

? TRANSMIT AND EXPECT LIST MUST BE IDENTICAL FOR LOOP?

IF RUN COMMAND WITH '/LOOP/CH' IS TYPED TRANSMIT AND EXPECT LISTS MUST BE EQUAL. IF THEY ARE NOT THIS ERROR WILL BE DISPLAYED. USE 'SE E=T' COMMAND.

3.2.2 JCLT ERROR MESSAGES:

BAD CLOCK - PROGRAM WILL HANG ON 'TIMEOUT'!!
THIS MEANS THAT EITHER NO CLOCK WAS ON THE SYSTEM OR THE ONE THAT WAS FOUND DID NOT INTERRUPT WHEN ASKED TO DO A 'TICK'.
THE PROGRAM WILL STILL RUN, BUT ANY OF THE PROGRAM THAT TIMES THE DEVICE WILL HANG IF THE DEVICE TIMES OUT. ALSO, THE EVENT LOG WILL CONTAIN A ZERO EVENT TIME FOR ALL EVENTS LOGGED.

MAX. CHAR. MSG COUNT EXCEEDED - MSG. NOT BUILT !!

THIS MEANS THAT THE TRANSMIT OR EXPECT BUFFER IS FULL. NO MORE MESSAGES CAN BE ADDED TO THAT BUFFER.

BUFFER FULL - MSG. NOT BUILT !!

THIS MEANS THAT THE LAST MESSAGE YOU

TRIED TO ADD TO EITHER THE TRANSMIT OR EXPECT BUFFER CAUSED THE TOTAL NUMBER OF MESSAGES TO BE EXCEEDED. NO MORE MESSAGES CAN BE ADDED TO THAT BUFFER. THE LIMIT IS DETERMINED BY THE SIZE OF THE MESSAGE POINTER TABLE.

CHAR. COUNT EXCEEDS BUFF LIMIT - MSG TRUNCATED

THIS MEANS THAT THE LAST MESSAGE YOU TRIED TO ADD TO THE TRANSMIT OR EXPECT BUFFER CAUSED THE TOTAL CHAR. COUNT FOR THAT BUFFER TO EXCEED THE LIMIT. THE MESSAGE WAS TRUNCATED TO COMPLETELY FILL THE BUFFER. NO MORE MESSAGES CAN BE ADDED TO THAT BUFFER.

3.2.3 DEVICE ERROR MESSAGES

DATA COMPARISON DATA ERROR
BYTE # IN MSG=XXX EXPTD=YYY

RECVD=ZZZ

XXX= OFFSET OF THAT BYTE FROM THE START OF THE COMPARE OR EXPECT MESSAGE.
YYY= THE CONTENTS OF THAT BYTE IN THE EXPECTED MESSAGE
ZZZ= THE CONTENTS OF THAT BYTE IN THE RECEIVED MESSAGE

UP TO FIVE OF THESE ERRORS WILL BE PRINTED PER MESSAGE COMPARED. ONLY THE FIRST FIVE MISMATCHES WILL BE INDIVIDUALLY REPORTED, BUT TOTAL NUMBER OF MISTMATCHES IS REPORTED BY ANOTHER ERROR.

PRINTING THE EVENT LOG AND USING THE DCLT 'DUMP' COMMAND WILL ALLOW YOU TO FIND THE ADDRESS OF THE MESSAGE AND EXAMINE IT.

DATA COMPARISON DATA ERROR
TOTAL MISMATCHES IN MSG = NNN

THIS MEANS THAT WHEN THE MESSAGE RECEIVED WAS COMPARED AGAINST THE MESSAGE THAT WAS EXPECTED, SOME OF THE CHARS. WERE NOT THE SAME.

DATA COMPARISON LENGTH ERROR
COMPARE COUNT= XXX RECEIVE COUNT= ZZZ

XXX= NUMBER OF BYTES IN THE COMPARE MESSAGE
ZZZ= NUMBER OF BYTES IN THE RECEIVED MESSAGE

THIS MEANS THAT THE MESSAGE RECEIVED
WAS A DIFFENT LENGTH THEN THE MESSAGE
THAT WAS EXPECTED.

* NOTE * - IN THE FOLLOWING ERROR DESCRIPTIONS XXXXX
***** REFERS TO THE OCTAL CONTENTS OF THE DEVICE REGISTERS
SPECIFIED.

TIME OUT WAITING FOR RDI TO CLEAR

SELO SEL2
XXXXXX XXXXXX

THIS MEANS THAT A SOFTWARE TIMER EXPIRED BEFORE
THE DEVICE CLEARED RDI IN RESPONSE TO THE DROPPING
OF RQI.

NOTE: PROGRAM RESETS TIMER AND WAITS AGAIN
SO AN EFFECTIVE LOOP ON ERROR IS SETUP.

TIME OUT WAITING FOR RDI TO SET

SELO SEL2
XXXXXX XXXXXX

THIS MEANS THAT A SOFTWARE TIMER EXPIRED BEFORE
THE DEVICE CAUSED AN INTERRUPT IN RESPONSE TO THE
PROGRAM SETING RQI.

NOTE: PROGRAM RESETS TIMER AND WAITS AGAIN
SO AN EFFECTIVE LOOP ON ERROR IS SETUP.

TIME OUT WAITING FOR RUN TO SET

SELO SEL2
XXXXXX XXXXXX

THIS MEANS THAT A SOFTWARE TIMER EXPIRED BEFORE
THE DEVICE SET THE RUN BIT IN RESPONSE TO THE
PROGRAM SETING MASTER CLEAR.

NOTE: PROGRAM RESETS TIMER AND ISSUES ANOTHER
MASTER CLEAR AND WAITS AGAIN SO AN EFFECITVE
LOOP ON ERROR IS SETUP.

THIS ERROR COULD INDICATE WRONG ADDRESS FOR
DMR/DMC-11 WAS GIVEN IN HARDWARE P TABLE.

TIME OUT WAITING FOR OUTPUT INTERRUPT

SELO SEL2
XXXXXX XXXXXX

THIS MEANS THAT A SOFTWARE TIMER EXPIRED BEFORE
THE DEVICE SET OUTPUT INTERRUPT IN RESPONSE TO
PROGRAM REQUESTING DEVICE TO TRANSMIT OR RECEIVE.

NOTE: PROGRAM RESETS TIMER AND WAITS AGAIN SO AN
EFFECTIVE LOOP ON ERROR IS SET UP.
THIS ERROR WILL OCCUR WHEN ONE NODE IS STARTED

IN RX OR TX MODE AND THE OTHER IS STILL BEING
SET UP. IGNORE THIS ERROR IF PROGRAM CONTINUES
WITHOUT FURTHER ERRORS.

INPUT INTERRUPT WHEN EXPECTING OUTPUT

SEL0 SEL2
XXXXXX XXXXXX

THIS WILL HAPPEN IF THE DEVICE IS BAD. IT MEANS
THAT AFTER THE PROGRAM HAS ISSUED ALL INPUT REQUESTS
TO THE DEVICE, THE DEVICE ISSUES AN INPUT INTERRUPT

ILLEGAL OUTPUT INTERRUPT

SEL2 SEL6
XXXXXX XXXXXX

THIS HAPPENS WHEN THE DEVICE ISSUES AN OUTPUT INTERRUPT
WITHOUT SETTING 'RDO'. IF THIS HAPPENS THE DEVICE IS BAD.

CONTROL OUT INSTEAD OF BA-CC OUT

SEL2 SEL6
XXXXXX XXXXXX MMMMMM

WHERE 'MMMMM' IS ONE OF THE FOLLOWING MESSAGES
THAT RESULT FROM INTERPRETING THE REGISTER CONTENTS
FOR YOU:

PROCEDURE ERROR/HALT
NON EXIST MEM
DDCMP START REC
DISCONNECT
LOST DATA
DDCMP MAINT REC
OVERRUN
TIME OUT
DATA CHECK
RUN SET ILLEAGLly (DMR IN DMR-MODE ONLY)
CD GLITCHED (DMR IN DMR-MODE ONLY)
RX IDLE (DMR IN DMR-MODE ONLY)
CTS FALILED (DMR IN DMR-MODE ONLY)

THIS ERROR OCCURS WHEN THE DEVICE SETS CONTROL OUT
TO INDICATE ERROR CONTIDION. THE PROGRAM EXPECTS A
BACC OUT.

TX BUFF COMPLETED AND SHOULD BE RX

SEL4 SEL6
XXXXXX XXXXXX

THIS ERROR OCCURS WHEN THE THE DEVICE HAS
A BACC OUT WITH TX COMPLETED AND THE PROGRAM
WAS EXPECTING A RX COMPLETED.

RX BUFF COMPLETED AND SHOULD BE TX

SEL4 SEL6
XXXXXX XXXXXX

THIS ERROR OCCURS WHEN THE THE DEVICE HAS
A BACC OUT WITH RX COMPLETED AND THE PROGRAM
WAS EXPECTING A TX COMPLETED.

WHERE 'XXXXX' IS THE OCTAL CONTENTS OF THAT
DEVICE REGISTER.

DOWN LINE LOAD ABORTED

THIS ERROR CAN ONLY OCCUR IN A NODE THAT
IS A DLL 'HOST' WHEN IT HAPPENS IT ALSO
PRINTS ONE OF THE FOLLWING QUALIFERS:

TX NOT COMPLETE

HOST DEVICE DID NOT GIVE BACC OUT TX
THIS SHOULD NOT HAPPEN BECAUSE DEVICE
DOES NOT NEED AN ACK FOR MAINT MESGS.

RX NOT COMPLETE

HOST DEVICE DID NOT GIVE BACC OUT RX
THIS CAN HAPPEN IF SATELLITE DOES NOT
SEND THE SEC BOOT REQUEST MESSAGE.

SEC REQ WORD1

HOST RECEIVED A MESSAGE FROM SATELLITE
BUT MESSAGE WAS NOT 1ST WORD OF SEC BOOT REQ.

SEC REQ WORD2

HOST RECEIVED A MESSAGE FROM SATELLITE
BUT MESSAGE WAS NOT 2ND WORD OF SEC BOOT REQ.

CALLED FROM PC. XXXXXX

THIS MESSAGE OCCURS WITH OTHER ERROR MESSAGES
TO INDICATE PC OF CALLING ROUTINE.

4.0 PERFORMANCE AND PROGRESS REPORTS

DCLT USES IT'S OWN METHOD FOR DETERMINING AN 'END OF PASS' WHICH IS CALLED A 'DCLT END OF PASS'. THE NUMBER OF 'DCLT PASSES' TO BE RUN IS SPECIFIED BY THE '/PASS=XXX' SWITCH ON THE DCLT RUN COMMAND. THE TOTAL NUMBER OF 'DCLT ERRORS' IS REPORTED WHEN 'X' NUMBER OF DCLT PASSES ARE COMPLETED.

4.1 PRINTING OF EVENT LOG

SIGNIFICANT EVENTS OR CHECK-POINTS WILL BE LOGGED IN A 'CIRCULAR QUEUE' STORAGE AREA CALLED THE EVENT LOG. THE LAST 'N' EVENTS ARE KEPT LOGGED AND CAN BE LISTED ON THE OPERATORS CONSOLE BY GIVING A 'PRINT' COMMAND AT THE 'DR>' (DIAGNOSTIC SUPERVISOR) OR 'DCLT>' (DCLT) LEVEL. THIS WILL TAKE YOU TO THE RPT> LEVEL. NOW GIVE THE 'LOG' COMMAND. THE EVENTS ARE PRINTED IN A 'LAST-IN FIRST-OUT' ORDER.

EVENT TIME IS TYPED OUT AS MMM:SS:TT (LIKE 254:36:07) WHERE MMM,SS,TT REPRESENT THE NUMBER OF MINUTES, SECONDS, CLOCK TICKS SINCE THE LAST START OR RESTART. IT SHOULD BE NOTED THAT THE TIMES ARE RELATIVE SINCE WHILE THE PROCESSOR IS RUNNING AT PRIORITY 7 THE CLOCK CAN'T INTERRUPT TO KEEP TIME. THIS IS THE CASE WHILE THE PROGRAM IS FETCHING DCLT COMMANDS FROM THE OPERATOR. IT SHOULD ALSO BE NOTED THAT THERE ARE ONLY 8 BITS AVAILABLE TO STORE RELATIVE MINUTES SO 'TIME' WILL WRAP TO 000:00:00 AFTER 256:59:59.

A START OR RESTART COMMAND AT THE 'DR>' LEVEL INITIALIZES THE EVENT LOG. THEREFORE IT IS WISE TO DO A 'PRINT' AT THE 'DR>' LEVEL BEFORE GIVING A 'START' OR 'RESTART'.

THE TYPES OF EVENTS KEPT IN THE EVENT LOG ARE:

TRANSMIT MESSAGE QUEUED:

EVENT TIME, ADDRESS OF 1ST BYTE OF MESSAGE,
TOTAL NO. OF BYTES, MODEM STATUS AT THAT TIME.

TRANSMIT MESSAGE COMPLETED:

EVENT TIME, ADDRESS OF 1ST BYTE OF MESSAGE,
TOTAL NO. OF BYTES, MODEM STATUS AT THAT TIME.

RECEIVE SPACE QUEUED:

EVENT TIME, ADDRESS OF 1ST BYTE OF MESSAGE,
TOTAL NO. OF BYTES, MODEM STATUS AT THAT TIME.

RECEIVE MESSAGE COMPLETED:

EVENT TIME, ADDRESS OF 1ST BYTE OF MESSAGE,
TOTAL NO. OF BYTES, MODEM STATUS AT THAT TIME.

DATA COMPARISON STARTED:

EVENT TIME, ADDRESS OF 1ST BYTE OF RECEIVED MSG.,
TOTAL NO. OF BYTES IN RCV. MSG., TOTAL NO. OF BYTES
IN EXPECT MSG.

DATA COMPARISON DATA ERROR:

EVENT TIME, ADDRESS OF 1ST BYTE OF RECEIVED MSG.,
TOTAL NO. OF BYTES IN RCV. MSG., TOTAL NO. OF

COMPARISON FAILURES
DATA COMPARISON LENGTH ERROR:
EVENT TIME, ADDRESS OF 1ST BYTE OF RECEIVED MSG.,
TOTAL NO. OF BYTES IN RCV. MSG., TOTAL NO. OF BYTES
IN EXPECT MSG.
DEVICE INIT AND SETUP:
EVENT TIME, MODE OF OPERATION, TYPE OF MAINTENANCE
LOOP, 'DCLT' PASS COUNT, 'RUN' PARAMETERS
DEVICE ERROR:
EVENT TIME, DEVICE ERROR MESSAGE, CONTENTS OF TWO
REGISTERS RELATING TO THE ERROR.
END OF PASS:
EVENT TIME, 'DCLT' PASS COUNT, 'DCLT' ERROR COUNT,
NO. OF 'NOBUFF'S'(NO. OF CONTROL-OUTS WITH THE
NO-BUFFER SET SINCE THE LAST 'DCLT RUN' COMMAND.)

NOTE: IF THE NODES ON THE LINK ARE SIMILAR WITH
RESPECT TO CONSOLE SPEED AND SETUP, THE
NUMBER OF 'NOBUFFS' SHOULD BE NEAR ZERO.

4.2 OPERATOR STATUS MESSAGES

THE '/STATUS, /NOSTATUS' QUALIFIERS FOR THE DCLT 'RUN' COMMAND
ENABLES/DISABLES THE PRINTING OF PROGRAM STATUS MESSAGES TO THE
OPERATOR. THESE MESSAGES ARE INTENDED TO TELL THE OPERATOR WHAT
THE DCLT PROGRAM IS CURRENTLY DOING. BELOW ARE THE MESSAGES THAT
MIGHT BE PRINTED AND THEIR MEANING:

MESSAGE	MEANING
TXQ	DEVICE IS ABOUT START TRANSMITTING A MESSAGE
TXC	TRANSMISSION OF MESSAGE COMPLETED
RXQ	DEVICE HAS QUEUED SPACE TO RECEIVE/ COMPLETED RECEIVE
ERR	DEVICE ERROR HAS OCCURRED
INI	DEVICE ABOUT TO BE INITIALIZED
MSC	ABNORMAL MODEM STATUS CHANGE
CMP	ABOUT TO DO DATA CHECKING OF RECVD VS. EXPTD DATA
CML	LENGTH ERROR OCCURRED DURING DATA COMPARISON
CMD	DATA ERROR OCCURRED DURING DATA COMPARISON
EOP	END OF PASS

4.3 PRINTING OF DMR/DMC-11 BASE TABLE

AT THE 'DCLT>' OR 'DR>' LEVEL, GIVE THE PRINT COMMAND. THIS WILL
TAKE YOU TO THE 'RPT>' LEVEL. YOU NOW HAVE THE OPTION OF PRINTING
ONLY ERROR LOCATIONS, ENTIRE BASE TABLE OR A SINGLE LOCATION.
YOU ONLY HAVE TO INPUT ENOUGH OF THE COMMAND TO MAKE IT UNIQUE.
THE ENTIRE BASE TABLE IN LOCAL PDP11 MEMORY IS UPDATED BY THE DMC
OR DMR, WHENEVER A FATAL ERROR OCCURS. THE ERROR COUNTER LOCATIONS
OF THE BASE TABLE ARE UPDATED EVERY SECOND BY THE DMC OR DMR IN
DMC MODE. IF THE DMR IS IN DMR MODE, THE ENTIRE BASE TABLE WILL BE
UPDATED AT 'END OF DCLT PASS'.

4.3.1 PRINTING ERROR LOCATIONS

TO PRINT ERROR COUNTER LOCATIONS, INPUT 'BASE/ERROR'. FOR A DMC LOCATIONS BASE+3..BASE+12 WILL BE DISPLAYED. IF A DMR, LOCATIONS BASE+3..BASE+41 WILL BE DISPLAYED. THE BASE ADDRESS IN THIS PROGRAM IS ALWAYS 17370.

EXAMPLE - DEVICE IS DMC

RPT> (A) ? B/E

LOCATION	CONTENTS	DESCRIPTION
17373	004	NAKS-MSG NO BUFFERS CUMUL
.	.	.
17402	007	REPS RECD CUMUL

4.3.2 PRINTING ENTIRE BASE TABLE

TO PRINT THE ENTIRE BASE TABLE, INPUT 'BASE/FULL'. IF A DMC 256 BYTES WILL BE DISPLAYED. IF A DMR, 128 BYTES WILL BE DISPLAYED. IN ORDER TO SAVE PROGRAM SPACE IN MEMORY, NOT ALL LOCATIONS WILL HAVE A DESCRIPTIVE MESSAGE. WHEN IN DOUBT SEE THE DMC OR DMR TECHNICAL MANUALS FOR A FULL DESCRIPTION.

EXAMPLE - DEVICE IS DMR IN DMC MODE;

RPT> (A) ? BASE/FULL

LOCATION	CONTENTS	DESCRIPTION
17370	000	BASE TABLE UPDATE INDEX POINTER
17371	000	BASE TABLE UPDATE LIMIT
17372	000	BEGINNING OF BASE TABLE DATA
17373	000	NAKS RCVD..BUFFER TEMP UNAVAILABLE
.
17567	.	SEE DMR MANUAL FOR DESCRIPTION

4.3.3 PRINTING SINGLE LOCATION

TO EXAMINE A SINGLE LOCATION, INPUT 'BASE/OFFSET=NNN'. FOR A DMC NNN IS A OCTAL NUMBER BETWEEN 0-377. FOR A DMR, NNN IS A OCTAL NUMBER BETWEEN 0-177. IF THE OFFSET VALUE IS NOT WITHIN THIS RANGE AN ERROR MESSAGE WILL BE PRINTED.

EXAMPLE - DEVICE IS DMR

RPT> (A) ? B/O=27

LOCATION	CONTENTS	DESCRIPTION
17417	006	STREAMING TIME OUT COUNT

RPT> (A) ?

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. [10]) 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	1	:[0] FULL OR HALF DUPLEX FLAG (BIT0=1 IF FULL)
.WORD	160170	:[2] CSR ADDRESS
.WORD	300	:[4] INTERRUPT VECTOR
.WORD	240	:[6] INTERRUPT PRIORITY (5)
.WORD	0	:[10] SPARE
.WORD	0	:[12] OPTION TYPE (0=DMC,5=DMR-DMC MODE,7=DMR)

6.0 MODE AND MESSAGE DESCRIPTIONS

6.1 MODE DESCRIPTIONS

BECAUSE THE DMC-11 AND DMR-11 SUPPORT DDCMP OPERATION IN THE FIRMWARE, THE PDP11 DCLT PROGRAM IS UNABLE TO CONTROL OR KNOW EXACTLY WHAT IS BEING TRANSMITTED OR RECEIVED AT ANY GIVEN TIME. ALL DATA MESSAGES ARE ENCLOSED IN A DDCMP ENVELOPE AND THEREFORE CONTROL MESSAGES (ACKS,NAKS...) ARE ALSO BEING TRANSMITTED AND RECEIVED.

6.1.1 TRANSMIT MODE

A LIST OF MESSAGES IS TRANSMITTED WITHOUT EXPECTING ANY DATA TO BE RECEIVED.

6.1.2 RECEIVE MODE

SPACE IS QUEUED FOR THE DEVICE TO RECEIVE MESSAGES. AFTER RECEIVING AN 'EXPECTED' NUMBER OF MESSAGES, THE DATA RECEIVED CAN BE COMPARED AGAINST A LIST OF 'EXPECT TO RECEIVE' MESSAGES IF DATA-CHECKING IS ENABLED.

6.1.3 PASSIVE MODE

EVERY TIME A MESSAGE IS RECEIVED, A MESSAGE IS TRANSMITTED.
DATA CHECKING CAN BE DONE ON THE RECEIVED DATA. THE "/ECHO, /NOECHO"
ENABLES/DISABLES THE RETRANSMISSION OF THE DATA RECEIVED.

6.1.4 ACTIVE MODE

A LIST OF MESSAGES IS TRANSMITTED AND MESSAGES ARE RECEIVED.
AFTER RECEIVING AN "EXPECTED" NUMBER OF MESSAGES, THE DATA RECEIVED
CAN BE COMPARED AGAINST A LIST OF "EXPECT TO RECEIVE" MESSAGES
IF DATA-CHECKING IS ENABLED.

NOTE: IF BOTH ENDS OF THE LINK ARE IN ACTIVE MODE, THEN THE
LINK MUST BE A FULL DUPLEX LINK!

6.1.5 DOWN-LINE-LOAD

THE "HOST" OR ORIGINATING STATION REQUESTS THE "SATELLITE" OR
BOOT STATION TO ENTER MOP MODE. THE SATELLITE THEN SENDS A
"SECONDARY BOOT REQUEST MESSAGE". THE "HOST" THEN CHECKS THE
RECEIVED MESSAGE TO SEE THAT IT IS A "SECONDARY BOOT REQUEST".
THEN THE HOST SENDS A "MEMORY LOAD WITH TRANSFER ADDRESS"
THAT CONTAINS IMAGE DATA TO BE LOADED BY THE SATELLITE'S
M9301-YJ/M9312 STARTING AT LOC. 0. THIS IMAGE DATA WILL CONTAIN A
CODE THAT PRINTS A MESSAGE SAYING DOWN-LINE-LOAD WAS SUCCESSFUL.
THE BOOTING PROCESS OVERWRITES PART OF THE "VECTOR" AREA SO THE DCLT
PROGRAM MUST BE RELOADED IN THE "SATELLITE" SYSTEM.

THE SATELLITE WILL ENTER MOP MODE ONLY IF THE PASSWORD WORD
SUPPLIED BY THE USER MATCHES THAT SET IN ITS PASSWORD SWITCH PACK.
INCLUDED IN THE "SECONDARY BOOT MESSAGE", IS THE DEVICE TYPE CODE
THAT IS DECIPHERED AND INCLUDED IN AN IDENTIFICATION MESSAGE.

EXAMPLE DOWNLINE LOAD:

DCLT>R M=D
SATELLITE PASSWORD = NNN ;NNN = OCTAL # BETWEEN 0-376
SECONDARY BOOT REQ FROM XXX DEVICE TYPE = YY

<u>YY</u>	<u>XXX</u>
0	DP
2	DU
4	DL
6	DQ
8	DA
10	DUP
12	DMC
14	DN
16	DLV
18	DMP
20	DTE
22	DV

24
28
30
32
34

DZ
KDP
KDZ
KL
DMV

6.1.6 TALK MODE

THE 'TALK' END OF THE LINK TRANSMITS OPERATOR-TYPED MESSAGES UNTIL A 'EXIT' MESSAGE IS TYPED. AT THAT POINT, THE NODE GOES INTO 'LISTEN' MODE. AN 'EXIT MESSAGE' IS A MESSAGE WHOSE FIRST FOUR CHARACTERS ARE 'EXIT'. SINCE ONLY THE FIRST FOUR CHARACTERS NEED TO BE 'EXIT', MORE CHARACTERS CAN BE ADDED SO THAT A MESSAGE MAY BE SENT AND THE MODE SWITCHED ALL AT ONCE. FOR EXAMPLE:

TLK> EXIT ALL OF THIS LINE IS SENT THEN MODE SWITCHED

6.1.7 LISTEN MODE

THE 'LISTEN' END OF THE LINK PRINTS ALL OF THE MESSAGES RECEIVED BY THE DEVICE ON THE OPERATOR'S CONSOLE. IF THE MESSAGE RECEIVED IS AN 'EXIT' MESSAGE, THEN THE NODE ENTERS 'TALK' MODE. AN 'EXIT MESSAGE' IS A MESSAGE WHOSE FIRST FOUR CHARACTERS ARE 'EXIT'.

6.1.8 MAINTENANCE 'LOOP' MODES

REMEMBER THAT THE WHENEVER A 'RUN' COMMAND IS TYPED, THE DEFAULT IS NO LOOPBACK AND THAT A LOOP MODE MUST BE SPECIFIED BY A '/LOOP=..' IF A LOOP MODE IS DESIRED.
LOOP MODES ARE ONLY VALID IF THE MODE TO RUN IS ACTIVE !

INTERNAL TTL

THE 'LU LOOP' BIT IS SET SO THAT THE UNIT'S SERIAL LINE OUT IS LOOPED BACK TO THE SERIAL LINE IN AT THE TTL LEVEL BEFORE LEVEL CONVERSION.

CABLE

NOT USED BY DMR OR DMC-11 CODE.

LOCALMODEM

FOR DMR-11 IN DMR MODE AND RS449 MODEMS ONLY. CAUSES A 'WRITE MODEM' TO BE DONE TO SET UP LOCAL-LOOPBACK (MAINT1) . ALSO CALLED ANALOG-LOOPBACK.

REMOTEMODEM

FOR DMR-11 IN DMR MODE AND RS449 MODEMS ONLY. CAUSES A 'WRITE MODEM' TO BE DONE TO SET UP REMOTE-LOOPBACK (MAINT2) . ALSO CALLED DIGITAL-LOOPBACK.

THE FOLLOWING TABLE SUMMARIZES THE MODES THAT CAN BE RUN TOGETHER WHEN THE DCLT PROGRAM IS RUNNING ON TWO PROCESSORS (ONE AT EACH END OF THE LINK):

STATION A "HOST" NODE	"/LOOP" ALLOWED?	STATION B "REMOTE" NODE	DUPLEX
TALK	NO	LISTEN*, RECEIVE	HALF OR FULL
LISTEN	NO	TALK*, TRANSMIT	HALF OR FULL
TRANSMIT	NO	RECEIVE*, LISTEN	HALF OR FULL
RECEIVE	NO	TRANSMIT*, TALK	HALF OR FULL
PASSIVE	NO	ACTIVE*	HALF OR FULL
ACTIVE	YES	ACTIVE*	FULL
		PASSIVE*	HALF OR FULL
DOWNLINELOAD	NO	PASSIVE	HALF FORCED

*= MOST LIKELY TO BE IN THAT MODE

6.2 MESSAGE DESCRIPTIONS

NAME	DESCRIPTION
ZEROES	MESSAGE OF ALL 0'S (00000000,00000000,00000000,...)
ONES	MESSAGE OF ALL 1'S (11111111,11111111,11111111,...)
?ALT	MESSAGE OF ALTERNATING 1'S (10101010,10101010,...)
OALT	MESSAGE OF ALTERNATING 0'S (01010101,01010101,...)
CCITT	'CCITT' 512-BIT (VS. 511 BITS) TEST PATTERN
ITEP	'INTERPROCESSOR TEST PROGRAM'S (ITEP)' MESSAGE 1(DP1:) (<177><177>/\$A THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.<15><12><001><177><177><177><177>)
ALPHA	ALPHA-NUMERICS (OR FUTURE COMM TURNAROUND MSG) (#\$!' (AMPERSAND)'()*+,-.0123456789:;<=>?@ABCDEFGHIJK LMNOPQRSTUVWXYZ/[\\]^_`)
'A-Z,0-9,SPACES,TABS'	THESE ARE THAT THE CHARACTERS THAT CAN BE TYPED BETWEEN QUOTATION MARKS ('..') TO SPECIFY A UNIQUE MESSAGE. (CALLED AN OPERATOR SPECIFIED MESSAGE.)

7.0 OTHER INFORMATION

7.1 INTERFACING TO AN "ITEP" NODE

WHEN DCLT IS USED TO INTERFACE TO AN ITEP NODE.
THE TABLE BELOW APPLIES:

ITEP NODE	DCLT NODE
ONE-WAY-OUT	RECEIVE OR LISTEN
ONE-WAY-IN	TRANSMIT OR TALK
INTERNAL LOOP	ACTIVE
EXTERNAL LOOP	ACTIVE OR PASSIVE

NOTE: WHEN INTERFACING TO ITEP IF THE RX BUFFER ON THE
ITEP SIDE IS ONLY 10 BYTES LARGER THAN THE TX BUFFER YOU
HAVE SELECTED, SO BE SURE TO SET THE TX BUFFER ON THE DCLT
NODE ACCORDINGLY.

WHEN ITEP IS IN A MODE THAT IT IS EXPECTING TO BE TRANSMITTED
TO, A SOFT ERROR 'BASE TABLE ERR COUNTS NON-ZERO' WILL OCCUR.
THIS IS DUE TO THE SPEED DIFFERENCES IN THE SOFTWARE.

WHEN DCLT IS IN LISTEN MODE THE RX BUFFER IS ONLY
82 BYTES LONG THEREFORE DO NOT SEND THE DCLT NODE
ITEP MSG. 3 FROM THE ITEP NODE OR A 'LOST DATA' ERROR WILL
OCCUR

BE SURE ITEP NODE HAS INCORPERATED PATCH FROM DEPO# MD-11-DZDMD-A1

ITEP NODE SHOULD ALWAYS BE RUN WITH SW 4 = TO 0

7.2 TROUBLESHOOTING HINTS

LISTED BELOW ARE SOME SETUPS THAT COULD BE USED FOR ISOLATING FAULTS.
THESE ARE BY NO MEANS THE ONLY WAYS DCLT CAN BE USED !!!!!!!
DCLT IS MEANT TO BE A VERY FLEXIBLE TOOL! THIS SECTION IS MEANT TO
GIVE SOMEONE NOT TOO FAMILIAR WITH DCLT A PLACE TO START.

REMEMBER THAT THE PRINTING OF STATUS MESSAGES AND PRINTING OF THE
EVENT LOG CAN PROVIDE A LOT OF INFORMATION ABOUT THE SEQUENCE OF
EVENTS AND HOW THE DEVICE AND LINK ARE BEHAVING.

NOTE: IF BOTH NODES IN ACTIVE AND "/NOCHECK" IS USED,
----- END-OF-PASS IS DEFINED AS RECEIVING 1 MESSAGE
AND COMPLETING THE TRANSMIT LIST. WITH NO DATA
CHECKING, THERE IS NO WAY FOR DCLT TO KNOW HOW
MANY MESSAGES IT SHOULD EXPECT TO RECEIVE.

7.2.1 INTERNAL LOOP AT EACH NODE

RUN EACH END OF THE LINK IN ACTIVE MODE WITH LOOP=INTERNAL.
TRANSMIT TWO OR THREE MESSAGES WITH NO DATA CHECKING.
STATUS PRINTING COULD BE TURNED OFF IF ON, BUT SEEING THE SEQUENCE
OF EVENTS MIGHT BE INFORMATIVE.

A POSSIBLE COMMAND SEQUENCE IS:

```
C E
C T
SE T=ONES/S=20/C=2
R M=A/LO=I/NOCH/STAT
```

WHAT THE ABOVE COMMAND SEQUENCE MEANS:

THE 'C E' AND THE 'C T' INITIALIZES THE 'EXPECT'
LIST AND THE 'TRANSMIT LIST'. THE 'SE T=ONES/S=20/C=2'
SETS THE TRANSMIT LIST TO CONTAIN 3 MESSAGES. THE MESSAGES
CONTAIN DATA OF ALL ONES AND EACH ONE IS 20 BYTES IN LENGTH.
THE 'R M=A/LO=I/NOCH/STAT' SETS THE MODE TO RUN IN TO BE
ACTIVE AND LOOP TYPE TO BE INTERNAL TTL. THE PROGRAM WILL
NOT BE CHECKING DATA SO THERE WAS NO NEED TO SET UP AN
EXPECT LIST. THE PROGRAM WILL BE PRINTING STATUS MESSAGES.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND
IF THINGS ARE RUNNING CORRECTLY :

```
INI RXQ TXQ RXQ TXC TXQ RXQ TXC
TXQ RXQ TXC EOP
MODE=ACTIVE/LOOP=INTERNAL/PASS=00000
/STATUS/NOCHECK/NOECHO/NOMODEM
DCLT> (A) ?
```

THIS GIVES YOU A IDEA IF THE COMM. DEVICE CAN EVEN TRANSMIT AND
RECEIVE. ANY ERRORS REPORTED WILL PROBABLY BE DUE TO INCORRECT
DEVICE ADDRESSES BEING USED OR A FAULTY DEVICE. CHECK ADDRESSES
WITH 'DISPLAY' AND RUN THE PREREQUISITE DIAGNOSTICS FOR THE COMM.
DEVICE.

NOW TRY RUNNING EACH NODE THE SAME WAY WITH DATA CHECKING ENABLED.
A POSSIBLE COMMAND SEQUENCE IS:

```
SE E=T
R M=A/LO=I/CH/PAS=3
```

WHAT THIS SEQUENCE MEANS:

THIS SEQUENCE IS SIMILAR TO THE ONE ABOVE . THE 'SE E=T'
MAKES A COPY OF THE TRANSMIT LIST IN THE EXPECT LIST.
THE EXPECT LIST NOW CONTAINS 3 MESSAGES. THE MESSAGES WILL
HAVE ALL ONES FOR DATA AND BE 20 BYTES EACH IN LENGTH.

THE RUN COMMAND IS THE SAME WITH THE ADDITION OF TWO SWITCHES "/CH/PAS=3". THE "CH" SWITCH TELLS THE PROGRAM TO CHECK THE RECEIVED DATA AGAINST THE 'EXPECTED LIST'. THE "PAS=3" SWITCH TELLS THE PROGRAM TO RUN 3 PASSES BEFORE RETURNING TO THE DCLT> PROMPT.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND IF THINGS ARE RUNNING CORRECTLY :

```
INI RXQ TXQ RXQ TXC TXQ RXQ TXC
TXQ TXC CMP CMP CMP EOP RXQ TXQ
RXQ TXC TXQ RXQ TXC TXQ TXC CMP
CMP CMP EOP RXQ TXQ RXQ TXC TXQ
RXQ TXC TXQ TXC CMP CMP CMP EOP
MODE=ACTIVE/LOOP=INTERNAL/PASS=00000
/STATUS/CHECK/NOECHO/NOMODEM
```

IF A CABLE TURNAROUND CONNECTOR IS AVAILABLE, PUT IT ON THE END OF THE CABLE JUST BEFORE THE MODEM AND RUN IN ACTIVE MODE WITH NO LOOP. POSSIBLE COMMAND SEQUENCE IS:

```
R M=A/CH/PAS=3
```

WHAT THIS SEQUENCE MEANS:

THIS SEQUENCE HAS THE "/LO=I" REMOVED. THIS INFORMS THE DEVICE TO ACT AS IF IT WAS RECEIVING FROM ANOTHER NODE.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND IF THINGS ARE RUNNING CORRECTLY :

```
INI RXQ TXQ TXC RXQ TXQ TXC RXQ
TXQ TXC CMP CMP CMP EOP RXQ TXQ
TXC RXQ TXQ TXC RXQ TXQ TXC CMP
CMP CMP EOP RXQ TXQ TXC RXQ TXQ
TXC RXQ TXQ TXC CMP CMP CMP EOP
MODE=ACTIVE/PASS=00000
/STATUS/CHECK/NOECHO/NOMODEM
DCLT> (A) ?
```

7.2.2 TRANSMIT ON ONE NODE RECEIVE ON THE OTHER

NOW TRY TRANSMITTING FROM ONE END AND RECEIVING ON THE OTHER. MAYBE WITH NO DATA CHECKING AT FIRST TO ESTABLISH IF THE LINK IS WORKING. POSSIBLE COMMAND SEQUENCES ARE:

NODE A	NODE B
-----	-----
C E	C E
C T	C T
SE T=1ALT/S=250	R M=R/NOCH/PAS=3

R M=TR/PAS=3

WHAT THIS SEQUENCE MEANS:

THE 'C E ' AND 'C T' INITIALIZE BOTH THE TRANSMIT AND EXPECT LISTS. THE 'SE T=1ALT/S=250' SETS THE TRANSMIT LIST ON NODE A TO BE 1 MESSAGE WITH A LENGTH OF 250 BYTES AND DATA OF ALTERNATING ONES AND ZEROS. THE 'R M=TR/PAS=3' SETS THE RUN MODE OF NODE A TO BE TRANSMIT AND THE PASS COUNT IS SET TO 3. THE 'R M=R/NOCH/PAS=3' SETS THE RUN MODE OF NODE B TO BE RECEIVE, NO DATA CHECKING IS TO BE DONE, AND THE PASS COUNT IS SET TO THREE.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND IF THINGS ARE RUNNING CORRECTLY :

FOR NODE A:

```
INI TXQ TXC EOP TXQ TXC EOP TXQ
TXC EOP
MODE=TRANSMIT/PASS=00000
/STATUS/NOCHECK/NOECHO/NOMODEM
DCLT> (A) ?
```

FOR NODE B:

```
INI RXQ EOP RXQ EOP RXQ EOP
MODE=RECEIVE/PASS=00000
/STATUS/NOCHECK/NOECHO/NOMODEM
DCLT> (A) ?
```

NOW TRY DOING DATA CHECKING ON THE MESSAGE(S) BEING TRANSMITTED. POSSIBLE COMMAND SEQUENCES ARE:

```
R M=TR/PAS=3
SE E=1ALT/S=250
R M=R/CH/PAS=3
```

WHAT THIS SEQUENCE MEANS:

THE 'SE E=1ALT/S=250' LINE MUST BE ADDED HERE TO SET UP THE 'EXPECT LIST' ON THE RECEIVE NODE SO IT WILL KNOW WHAT TO COMPARE AGAINST. THE CHANGE IN THE RUN COMMAND IS FROM 'NOCH' TO 'CH'. THE 'CH' ENABLES DATA CHECKING.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND IF THINGS ARE RUNNING CORRECTLY:

NODE A: IS THE SAME AS ABOVE.

NODE B:

```
INI RXQ CMP EOP RXQ CMP EOP RXQ CMP EOP
MODE=RECEIVE/PASS=00000
/STATUS/CHECK/NOECHO/NOMODEM
DCLT> (A)?
```

NOW RUN THRU THE SEQUENCE AGAIN WITH NODE A RECEIVING
AND NODE B TRANSMITTING TO CHECK OUT THE OPPOSITE
DIRECTION OF DATA FLOW.

7.2.3 ONE NODE ACTIVE THE OTHER NODE PASSIVE

NOW TRY RUNNING ONE NODE IN ACTIVE MODE WHILE THE OTHER
END RUNS IN PASSIVE. DATA CHECKING SHOULD BE TURNED OFF
IF THE MESSAGE LISTS ARE NOT THE SAME.
POSSIBLE COMMAND SEQUENCES ARE:

NODE A	NODE B
-----	-----
C E	C E
C T	C T
SE T=CCITT/S=10/C=2	SE T=1ALT/S=20/C=2
R M=ACT/NOCH/PAS=3	R M=P/NOCH/PAS=3

WHAT THIS SEQUENCE MEANS:

THE EXECUTION OF THIS SEQUENCE CAUSES THE FOLLOWING
THINGS TO HAPPEN ON NODE A. THE TRANSMIT AND EXPECT
LISTS ARE INITIALIZED THEN THE TRANSMIT LIST IS SET
TO 3 MESSAGES OF 10 BYTES EACH. THE DATA USED IN THE
TRANSMIT MESSAGES IS THE CCITT PATTERN. THEN NODE A
IS RUN IN ACTIVE MODE WITH DATA CHECKING DISABLED AND
THE PASS COUNT SET TO THREE. NOTE STATUS WOULD STILL BE
PRINTED IF THE PREVIOUS SEQUENCES HAD BEEN RUN.
IF YOU ARE RUNNING FROM LOAD TIME YOU WOULD HAVE
TO ADD A '/STA TO THE RUN COMMAND LINE.

NODE B: THE TRANSMIT AND EXPECT LISTS ARE INTIALIZED
THEN THE TRANSMIT LIST IS SET TO 3 MESSAGES OF
20 BYTES EACH. THE DATA FOR EACH MESSAGE IS ALTERNATING
1'S AND 0'S. THE NODE IS THEN RUN IN PASSIVE MODE WITH
DATA CHECKING DISABLED AND THE PASS COUNT SET TO 3.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND
IF THINGS ARE RUNNING CORRECTLY :

FOR NODE A:

```
INI RXQ TXQ TXC TXQ RXQ TXC TXQ
RXQ TXC EOP RXQ TXQ RXC TXC TXQ
RXQ TXC TXQ RXQ TXC EOP RXQ TXQ
RXQ TXC TXQ RXQ TXC TXQ RXQ TXC
EOP
MODE=ACTIVE/PASS=00000
/STATUS/NOCHECK/NOECHO/NOMODEM
DCLT> (A) ?
```

FOR NODE B:

```
INI RXQ TXQ TXC RXQ TXQ TXC RXQ
TXQ TXC EOP RXQ TXQ TXC RXQ TXQ
TXC EOP RXQ TXQ TXC RXQ TXQ TXC
```

RXQ TXQ TXC EOP
MODE=PASSIVE/PASS=00000
/STATUS/NOCHECK/NOECHO/NOMODEM
DCLT> (A) ?

NOW USE DATA CHECKING WITH THE 'EXPECT MESSAGE LISTS' SET UP APPROPRIATELY. ANOTHER VARIATION IS TO HAVE LARGE SIZE MESSAGES ON ONE SIDE WITH SMALL MESSAGES ON THE OTHER.

THEN REVERSE THE SETUP SO THAT THE NODE RUNNING IN ACTIVE IS RUNNING IN PASSIVE AND VICE VERSA.

7.2.4 BOTH NODES ACTIVE

NOW BOTH NODES CAN BE RUN IN ACTIVE WITH DATA CHECKING ON. STATUS PRINTING COULD BE TURNED OFF IF YOU'RE NOT INTERESTED IN THEM.

NODE A	NODE B
-----	-----
C E	C E
C T	C T
SE T=OALT/S=10	SE E=OALT/S=10
SE T=CCITT/S=20	SE E=CCITT/S=20
SE T=ALPHA/S=30	SE E=ALPHA/S=30
SE E=ZERO/S=11	SE T=ZERO/S=11
SE E=ONES/S=21	SE T=ONES/S=21
SE E=ITEP/S=31	SE T=ITEP/S=31
R M=A/CH/NOST/PAS=3	R M=A/CH/NOST/PAS=3

WHAT THIS SEQUENCE MEANS:

NODE A SETS UP IS TRANSMIT LIST TO BE 3 MESSAGES. MESSAGE 1 IS 10 BYTES LONG AND CONTAINS DATA OF ALTERNATING 0'S AND 1'S MESSAGE 2 IS 20 BYTES LONG AND CONTAINS DATA OF THE CCITT PATTERN. MESSAGE THREE IS 30 BYTES LONG AND CONTAINS ALPHANUMERICS FOR DATA. THE EXPECT LIST ALSO CONTAINS 3 MESSAGES. MESSAGE 1 IS 11 BYTES LONG AND CONTAINS 0'S FOR DATA. MESSAGE TWO IS 21 BYTES LONG AND CONTAINS 1'S FOR DATA. MESSAGE 3 IS 31 BYTES LONG AND CONTAINS THE ITEP DATA. NODE B HAS THE SAME MESSAGES EXCEPT THAT THE TRANSMIT MESSAGE LIST IS THE EXPECT MESSAGE LIST AND VICE VERSA. BOTH NODES ARE RUN IN THE ACTIVE MODE WITH DATA CHECKING AND PASS COUNT EQUAL TO THREE.

WHAT YOU SHOULD SEE AFTER ENTERING THE RUN COMMAND IF THINGS ARE RUNNING CORRECTLY :
ON BOTH NODES A AND B:

MODE=ACTIVE/PASS=00000
/NOSTATUS/CHECK/NOECHO/NOMODEM

DCLT> (A) ?

A VARIATION THAT CAN BE USED IS FOR ONE END TO SEND A LOT OF SMALL MESSAGES AND THE OTHER TO SEND A FEW LARGE MESSAGES. THE 'END-OF-PASS' POINT WILL BE OUT OF SYNC BUT THIS IS NOT A PROBLEM.

7.2.5 TALK AND LISTEN MODES FOR COMMUNICATING

TALK AND LISTEN MODES ARE USEFUL IF THE OPERATORS WISH TO COMMUNICATE WITH EACH OTHER. JUST SETUP A TIME THAT EACH WILL GO TO THEIR MODE, TALK OR LISTEN, AND SEND MESSAGES OVER THE LINK. POSSIBLE COMMAND SEQUENCES ARE.

R M=LIS/NOST
LIS>

R M=TA/NOST
TLK>

7.3 EXAMPLES OF COMMANDS

THIS SECTION WILL SHOW A SAMPLING OF COMMANDS AND EXACTLY WHAT TO EXPECT FROM THEM.

7.3.1 EXAMPLES OF MESSAGES COMMANDS

THE CLEAR COMMANDS .

C E
C T

THIS WILL INITIALIZE THE TRANSMIT AND EXPECT LIST TO 1 MESSAGE OF 58 BYTES. THE DATA OF THE MESSAGE WILL BE THE ITEP MESSAGE.

IF THESE COMMANDS ARE FOLLOWED BY A SHOW COMMAND

SH E

SUCH AS THE SHOW EXPECT LIST. WHAT YOU WOULD SEE IS

MSG: TYPE=ITEP/SIZE=58
MODE=ACTIVE/PASS=00001
/NOSTATUS/CHECK/NOECHO/NOMODEM

DCLT> (A) ?

NOW IF YOU DID A SET EXPECT LIST COMMAND SUCH AS:

SE E=A/S=35/C=3

AND FOLLOWED IT WITH A SHOW EXPECT LIST COMMAND

SH E

WHAT YOU WOULD SEE IS

MSG: TYPE=ALPHA/SIZE=35
MSG: TYPE=ALPHA/SIZE=35
MSG: TYPE=ALPHA/SIZE=35
MSG: TYPE=ALPHA/SIZE=35
MODE=ACTIVE/PASS=00001
/NOSTATUS/CHECK/NOECHO/NOMODEM

DCLT> (A) ?

7.3.2 EXAMPLES STATISTICAL COMMANDS

IF YOU TYPE A HELP COMMAND

HELP
WHAT YOU WILL SEE IS

DCLT CMDS:

CLEAR OR SHOW EXPECTLIST OR TRANSMITLIST
PRINT
EXIT
DUMP START-END/B
SET EXPECTMSG OR TRANSMITMSG=TYPE/SIZE=N OR /COPY=N
SET EXPECT=TRANSMIT
TYPE=ONES,ZEROES,1ALT,0ALT,ITEP,CCITT,ALPHA
OR 'OPR SPCD=A-Z,SP,TAB,0-9 IN QUOTES'
RUN MODE=MTYP/LOOP=LTYP/CHECK,STATUS,ECHO,MODEM,PASS=N
MTYP=TRAN,REC,ACT,PAS,TAL,LIS,DOWN
LTYP=INT,CAB,LOC,REM/

DCLT> (A) ?

THE SAME WILL HAPPEN IF YOU USE THE ?

THE DUMP COMMAND WORKS LIKE THIS

DUM 41260-41300

THIS WILL DUMP THE DATA FROM ADDRESSES 41260 TO
41300 IN THE FOLLOWING MANNER

41260 104423 000167 177772 021122 012112 006312 006312 006312
41300 006312

IF YOU HAD USED THE /B SWITCH

DUM 41260-41300/B

WHAT YOU WOULD SEE IS

41260 023 211 167 000 372 377 122 024
41270 112 024 312 014 312 014 312 014
41300 312

7.3.3 EXAMPLES RUN COMMANDS

YOU CAN FIND SEVERAL EXAMPLES OF THE RUN COMMAND IN THE
TROUBLE SHOOTING HINTS SECTION BUT HERE ARE SOME OTHERS.

IF YOU WERE TO EXECUTE THE RUN COMMAND

R M=TR/NOST/CH/PAS=4

WHAT WOULD HAPPEN IS AFTER 4 PASSES THE PROGRAM WOULD RETURN
TO THE DCLT PROMPT AND PRINT

MODE=TRANSMIT/PASS=00000
/NOSTATUS/CHECK/NOECHO/NOMODEM

DCLT> (A) ?

IF YOU WERE TO EXECUTE THE RUN COMMAND

C E
C T

R M=A/LO=I/ST/CH/PAS=3

WHAT YOU WOULD SEE (IF USING DEFAULT TRANSMIT AND EXPECT

.....
.....

THIS GOES ON FOR 45 EVENTS IF THE MODE
PREVIOUSLY EXECUTED HAD THAT MANY
YOU EXIT FROM EVENT LOG PRINTING BY
TYPING A CONTROL C.

7.3.5 EXAMPLE EXIT COMMAND

THE EXIT COMMAND WORKS LIKE THIS. IF YOU
ENTERED THE REPORT LEVEL FROM THE SUPERVISOR
(DR>) THEN TYPING

EXIT

WILL RETURN YOU TO THE SUPERVISOR.

DR>

IF YOU ENTERED REPORT FROM THE DCLT LEVEL
THEN TYPING

EXIT

WILL RETURN YOU TO THE DCLT LEVEL.

DCLT>

7.4 THINGS TO WATCH OUT FOR

IF YOU ARE RUNNING DCLT ON SYSTEMS THAT HAVE CONSOLES
WITH DIFFERENT SPEEDS YOU WILL BE UNABLE TO USE THE
PRINT STATUS FEATURE IN CERTAIN MODES. THE RULE IS
IF IT DOESNT WORK WITH STATUS PRINTING RUN THE MODE
WITH NOSTATUS.

IF YOU ARE USING PASSIVE MODE WITH THE ECHO SWITCH
THEN YOU WILL PROBABLY HAVE TO RE-ENTER THE TRANSMIT
LIST ON THE SIDE WITH THE ECHO SWITCH. THE REASON IS
THAT THE TRANSMIT LIST GETS OVER WRITTEN WITH THE
RECEIVE LIST WHEN USING THE ECHO SWITCH.

4217
4218
4248
4249 002000
4250
4255
4259
4260
4280
4281
4282
4283
4284
4285
4286 002000
4287
4304
4305
4306 002000
(4) 002000
(4) 002000 103
(4) 002001 132
(4) 002002 103
(4) 002003 114
(4) 002004 113
(6) 002005 000
(6) 002006 000
(5) 002007 000
(5) 002010
(4) 002010 102
(5) 002011
(4) 002011 060
(5) 002012
(4) 002012 000000
(5) 002014
(4) 002014 003410
(5) 002016
(4) 002016 046060
(5) 002020
(4) 002020 000000
(5) 002022
(4) 002022 002130
(5) 002024
(4) 002024 000000
(5) 002026
(4) 002026 046442
(5) 002030
(4) 002030 000000
(5) 002032
(4) 002032 000000
(5) 002034
(4) 002034 000000
(5) 002036
(4) 002036 000000
(5) 002040
(4) 002040 002124

.SBTTL PROGRAM HEADER
BGNMOD

+++
: THE PROGRAM HEADER IS THE INTERFACE BETWEEN
: THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.

POINTER BGNRPT,BGNAU,BGNDU

HEADER CZCLK,B,0,1800.,0,#PRI07

LSNAME::
.ASCII /C/
.ASCII /Z/
.ASCII /C/
.ASCII /L/
.ASCII /K/
.BYTE 0
.BYTE 0
.BYTE 0
LSREV::
.ASCII /B/
LSDEPO::
.ASCII /O/
LSUNIT::
.WORD 0
LSTIML::
.WORD 1800.
LSHPCP::
.WORD LSHARD
LSSPCP::
.WORD 0
LSHPTP::
.WORD LSHW
LSSPTP::
.WORD 0
LSLADP::
.WORD LSLAST
LSSTA::
.WORD 0
LSCO::
.WORD 0
LSDTYP::
.WORD 0
LSAPT::
.WORD 0
LSDTP::
.WORD LSDISPATCH

(5) 002042
(4) 002042 000340
(5) 002044
(4) 002044 000000
(5) 002046
(4) 002046 000000
(5) 002050
(4) 002050 003
(3) 002051 003
(5) 002052
(4) 002052 000000
(5) 002054 000000
(5) 002056
(4) 002056 000000
(5) 002060
(4) 002060 012410
(5) 002062
(4) 002062 033732
(5) 002064
(4) 002064 000000
(5) 002066
(4) 002066 000000
(5) 002070
(4) 002070 034702
(5) 002072
(4) 002072 034674
(5) 002074
(4) 002074 000000
(5) 002076
(4) 002076 012424
(5) 002100
(4) 002100 104035
(5) 002102
(4) 002102 000000
(5) 002104
(4) 002104 033746
(5) 002106
(4) 002106 034650
(5) 002110
(4) 002110 034646
(5) 002112
(4) 002112 033740
(5) 002114
(4) 002114 000000
(5) 002116
(4) 002116 000000
(5) 002120
(4) 002120 000000

4307
4314

LSPRIO::
LSENV1:: .WORD #PRI07
LSEXP1:: .WORD 0
LSMREV:: .WORD 0
LSEF:: .BYTE CSREVISION
 .BYTE C\$EDIT
LSSPC:: .WORD 0
LSDEVP:: .WORD LSDVTYP
LSREPP:: .WORD LSRPT
LSEXP4:: .WORD 0
LSEXP5:: .WORD 0
LSAUT:: .WORD LSAU
LSDUT:: .WORD LSDU
LSLUN:: .WORD 0
LSDESP:: .WORD L\$DESC
LSLOAD:: EMT ESLOAD
LSETP:: .WORD 0
LSICP:: .WORD L\$INIT
LSCCP:: .WORD L\$CLEAN
LSACP:: .WORD L\$AUTO
LSPRT:: .WORD L\$PROT
L\$TEST:: .WORD 0
LSDLY:: .WORD 0
LSHIME:: .WORD 0

4326
4327
4328
4329
4330
4331
4332
4333
(4)
(3)
(6)
4334

.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

002122
002122 000001
002124
002124 034710

.WORD 1
LSDISPATCH::
.WORD T1

4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
(3)
(3)
(3)
4352
4362
4363
4364
4365
4366
4367
4368
4369
4370
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
(3)

002126
002126 000010
002130
002130

002130 000001

002132 160170
002134 000300
002136 000240
002140 000000

002142 000000

002144 000004
002146 000000

002150
002150

```
.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::

:INDEPENDENT SECTION
: THE NUMBERS IN BRACKETS ARE THE OFFSET VALUES USED IN THE PARAMETER
: CODING SECTION.

      .WORD  1                ;[0] FULL OR HALF DUPLEX FLAG (BIT0=1 IF FULL)

:DEVICE DEPENDENT SECTION
: ADDING OR REMOVING WORDS FROM THIS TABLE EFFECTS THE 'GET' CALLS IN
: THE HARDWARE PARAMETER CODING SECTION BY CHANGING 'OFFSETS'

      .WORD  160170          ;[2] CSR ADDRESS
      .WORD  300             ;[4] INTERRUPT VECTOR
      .WORD  240             ;[6] INTERRUPT PRIORITY (5)
      .WORD  0               ;[10] DEVICE PARAMETERS WORD
                                ; (ENABLE CRC, STRIP SYNC, COMPATIBLE MODE...)
      .WORD  0               ;[12] DEVICE OPTION TYPE(0=DMC,5=DMR-DMC MODE,
                                ; 7=DMR.
      .WORD  4               ;[14] BAUD RATE (0=2.4K, 1=4.8K, 2=9.6K, 3= 19.2K,
                                ; 4=56K, 5=250K, 6=500K, 7=1 MEGA-BAUD)
      .WORD  0               ;[16] LINE INTERFACE (422, V.35, INT, EIA...)

      ENDPHW

                                L10000:
```

4402
4403
4420
4443
4444
4445
4446
4456
4457
4458
4459
4460
4461
4462
4477
4478 002150

.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
BIT1== BIT01
BIT0== BIT00

:
: EVENT FLAG DEFINITIONS
: EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

:
EF.START== 32.
EF.RESTART== 31.
EF.CONTINUE== 30.
EF.NEW== 29.
EF.PWR== 28.
:

: START COMMAND WAS ISSUED
: RESTART COMMAND WAS ISSUED
: CONTINUE COMMAND WAS ISSUED
: A NEW PASS HAS BEEN STARTED
: A POWER-FAIL/POWER-UP OCCURRED

(1)
(1)
(1)
(1) 100000
(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001
(1)
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001
(1)
(1)
(1)
(1) 000040
(1) 000037
(1) 000036
(1) 000035
(1) 000034
(1)

```
(1) ;  
(1) ; PRIORITY LEVEL DEFINITIONS  
(1) ;  
(1) 000340 PRI07== 340  
(1) 000300 PRI06== 300  
(1) 000240 PRI05== 240  
(1) 000200 PRI04== 200  
(1) 000140 PRI03== 140  
(1) 000100 PRI02== 100  
(1) 000040 PRI01== 40  
(1) 000000 PRI00== 0  
(1) ;  
(1) ; OPERATOR FLAG BITS  
(1) ;  
(1) 000004 EVL== 4  
(1) 000010 LOT== 10  
(1) 000020 ADR== 20  
(1) 000040 IDU== 40  
(1) 000100 ISR== 100  
(1) 000200 UAM== 200  
(1) 000400 BOE== 400  
(1) 001000 PNT== 1000  
(1) 002000 PRI== 2000  
(1) 004000 IXE== 4000  
(1) 010000 IBE== 10000  
(1) 020000 IER== 20000  
(1) 040000 LOE== 40000  
(1) 100000 HOE== 100000  
4479
```

```
4481 ;***** INDEPENDENT EQUATES
4482
4483 001000 BUFLIM=512. ;MAX BUFFER SIZE IN BYTES
4484 ; APPLIES TO TX,RX AND CMP BUFFS
4485 000017 MSGLIM=15. ;MAX NO. OF MESSAGES PER BUFFER
4486 ; (FOR EACH INCREMENT (+1) TO MSGLIM,
4487 ; ADD 6 WORDS TO THE POINTER TABLE
4488 ; (PTRTAB:) SINCE THIS MEANS 2 MORE
4489 ; 'POINTER' WORDS PER BUFFER.
4490
4491 ;MODE OF OPERATION EQUATES
4492 000000 REC=0 ;RECEIVE MODE
4493 000001 TRA=1 ;TRANSMIT MODE
4494 000002 PAS=2 ;PASSIVE MODE
4495 000003 ACT=3 ;ACTIVE MODE
4496 000004 DOW=4 ;DOWN-LINE-LOAD MODE
4497 000005 TAL=5 ;TALK MODE
4498 000006 LIS=6 ;LISTEN MODE
4499
4500 ;MAINT LOOP TYPE EQUATES
4501 000000 NONE= 0 ;NO LOOP
4502 000001 TTL= 1 ;INTERNAL TTL
4503 000002 CABLE= 2 ;CABLE LOOP
4504 000003 MODLOC= 3 ;MODMEM LOCAL
4505 000004 MODREM= 4 ;MODEM REMOTE
4506 000005 MOP= 5 ;MOP
4507
4508 ;CLOCK ENABLE VALUES TO BE LOADED IN CLK'S CSR
4509 000100 LCLKEN= 100 ;L-CLOCK CSR VALUE TO ENABLE THE CLOCK
4510 000111 PCLKEN= 111 ;P-CLOCK CSR VALUE TO ENABLE THE CLOCK
4511 001600 PCLKCT= 1600 ;P-CLOCK COUNT SET REGISTER FOR COUNTER
4512
4513 ;PARAM WORD EQUATES
4514
4515 000001 STATB= BIT0 ;OPERATOR AWAKE ASKED FOR
4516 000002 DATCKB= BIT1 ;DATA CHECK BIT
4517 000004 ECHOB= BIT2 ;ECHO BIT
4518 000010 MOCHK= BIT3 ;MODEM CHECK/NO CHECK ADDED BY EC
4519 000020 CRCB= BIT4 ;CRC CALCUALTE ASKED FOR
4520 000040 PROTOB= BIT5 ;PROTOCOL PROCESSING ASKED FOR
4521
4522 ;OPTION TYPE EQUATES
4523
4524 000000 DMC= 0 ;DMC
4525 000004 DMRC6= 4 ;8206 DMR IN DMC MODE
4526 000005 DMRC7= 5 ;8207 DMR IN DMC MODE
4527 000006 DMR6= 6 ;8206 DMR IN DMR MODE
4528 000007 DMR7= 7 ;8207 DMR IN DMR MODE
4529
4530 ;EVENT LOG MESSAGE TYPES (USED TO LOCATE EVENT DESCRIPTION IN EVENT TABLE
4531 ; AND DISPATCHING TO SEPERATE SECTIONS OF THE EVENT REPORTING SECTION)
4532 000000 TXQ= 0 ;TRANSMIT MESSAGE QUEUED
4533 000002 TXC= 2 ;TRANSMIT COMPLETE
4534 000004 RXQ= 4 ;RECEIVE BUFFER QUEUED
4535 000006 RXC= 6 ;RECEIVE COMPLETE
4536 000010 DER= 10 ;DEVICE INFORMATION
```

```
4537      000012      DVI= 12      ;DEVICE ABOUT TO INIT
4538      000014      DCK= 14      ;DATA COMPARISON RESULTS
4539
4540      000020      DLE= 20      ;DATA COMPARISON LENGH ERROR
4541      000022      DDE= 22      ;DATA COMPARISON DATA ERROR
4542      000024      EOP= 24      ;END OF PASS
4543
4544      ;;;;EQUATES FOR FLAG WORD;;;;;
4545
4546      000001      ININT= 1      ;INPUT INT. REC.
4547      000002      OTINT= 2      ;OUTPUT INT REC
4548      000004      QRX= 4      ;RX QUED /COMPL
4549      000010      QTX= 10      ;TX QUED/COMPL
4550      000020      CTX= 20      ;TX COMPL AND IN TXSEL4 AND TSEL6
4551      000040      CRX= 40      ;RX COMPL AND IN TSEL4 AND TSEL6
4552      000100      ERX= 100      ;EXPECT TO GET A RX COMPLETED
4553      000200      ETX= 200      ;EXPECT TO GET A TX COMPLETED
4554      000400      DLLGA= 400      ;DOWN LINE LOAD GO AHEAD BIT
4555      001000      DMRRUN= 1000      ;DMR RUN MODE EXPECTED
4556      002000      BTUP= 2000      ;BASE TABLE UPDATE REQUESTED
4557
4558      ; SPECIAL CLI CODES FOR 'CHAR' ARGUMENT IN CLI CALLS
4559      ; (COMMAND LINE INTERPRETER DEFINITIONS)
4560      000000      CLIERR= 0
4561      000001      CLIEXI= 1
4562      000002      CLIBR= 2
4563      000003      CLIBIF= 3
4564      000004      CLISPA= 4
4565      000005      CLINUM= 5
4566      000006      CLIALP= 6
4567      000007      CLIALN= 7
4568      000010      CLIOCT= 8.
4569      000011      CLIDEC= 9.
4570      000012      CLISTR= 10.
4571
4572      ; DEFS FOR COMMAND LINE INTERPRETATION ACTION VALUES
4573      000000      NULL=0
4574      000001      CLEAR=1
4575      000002      SHOW=2
4576      000003      CHECK=3
4577      000004      RUN=4
4578      000005      HLP=5
4579      000006      CSHEXP=6
4580      000007      CSHTRN=7
4581      000010      SETEXP=10
4582      000011      SETTRN=11
4583      000012      SIZE=12
4584      000013      QCOPY=13
4585      000014      NUM=14
4586      000015      OPRMSG=15
4587      000016      STATUS=16
4588      000017      ENDQO=17
4589      000020      CMSG0=20
4590      000021      CMSG1=21
4591      000022      CMSG2=22
4592      000023      CMSG3=23
```

```
4593      000024      CMSG4=24
4594      000025      CMSG5=25
4595      000026      CMSG6=26
4596      000027      ATVMOD=27
4597      000030      PASM0D=30
4598      000031      RECM0D=31
4599      000032      LISMOD=32
4600      000033      DLLMOD=33
4601      000034      TRAM0D=34
4602      000035      TALMOD=35
4603      000036      NO=36
4604      000037      ECHO=37
4605      000040      CRC=40
4606      000041      PROTO=41
4607      000042      PASC=42
4608      000043      MOP=43
4609      000044      TTLLOP=44
4610      000045      CBLLOP=45
4611      000046      LMDLOP=46
4612      000047      RMDLOP=47
4613      000050      NOTNUF=50
4614      000051      BADCHR=51
4615      000052      DMPS=52
4616      000053      DMPE=53
4617      000054      DMPQ=54
4618      000055      PRNT=55
4619      000056      MOSC=56      ;MODEM/NOMODEM REV B BY EC
4620      000057      EXIT=57      ;EXIT COMMAND REV B BY EC
4621      000060      SETET=60     ;S E=T COMMAND REV B BY EC
4622      ;FOLLOWING EQUATES USED IN REPORT CLI REV B BY EC
4623      000001      RPHLP=1      ;HELP COMMAND
4624      000002      RPEXT=2     ;EXIT COMMAND
4625      000003      RPLOG=3     ;PRINT EVENT LOG COMMAND
4626      000004      RPSWE=4     ;BASE/ERROR COMMAND
4627      000005      RPSWF=5     ;BASE/FULL COMMAND
4628      000006      RPSWO=6     ;BASE/OFFSET
4629      000007      RNOTNF=7    ;MORE COMMAND NEEDED
4641
4642      ;***** DEVICE DEPENDENT EQUATES
4643      ; MODEM SIGNAL BIT DEFINITONS
4644      ; IF SIGNAL AVAILABLE IN DEVICE, EQUATE NAME TO BIT POSITION,
4645      ; ELSE EQUATE IT TO = 0
4646
4647      000004      CTS=      BIT2      ;CLEAR TO SEND (CIRCUIT CB)
4648      000010      DSR=      BIT3      ;DATA SET READY (CIRCUIT CC)
4649      000001      DCD=      BIT0      ;DATA CARRIER DETECT (CIRCUIT CF)
4650      000040      RTS=      BIT5      ;REQUEST TO SEND (CIRCUIT CA)
4651      000200      RI=       BIT7      ;RING INDICATOR (CIRCUIT CE)
4652      040000      SQD=     BIT14     ;SIGNAL QUALITY DETECT (CIRCUIT CG)
4653      001000      TM=       BIT9      ;MODEM IN TEST MODE (RS 449 ONLY CIRCUIT TM)
4654
4662
4663
4664      ; DEVICE SIGNALS
4665      000040      RQI=     BIT5      ;REQUEST IN
4666      000200      RDI=     BIT7      ;READY IN
```


4667 00020C
4668 000001
4669 040000
4670 004000
4671 000400
4672 002000
4673 000004
4674 000100
4675

RDO= BIT7
BACC= BIT0
MCLR= BIT14
LLOOP= BIT11
MAINTB= BIT8
HALFDB= BIT10
RXBIT= BIT2
IEO= BIT6

:BUFFER ADDR. CHAR COUNT
:MASTER CLEAR
:LINE UNIT LOOP(TTL)
:MAINT MODE BIT
:HALF DUPLEX BIT
:RX BIT
:ENABLE OUTPUT INTERRUPT BIT

4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687 002150
4688 002150 000001
4689 002152 000001
4690 002154 000001
4691 002156 000001
4692 002160 000100
4693 002162 000072
4694 002164 000101
4695 002166 000000
4696 002170 000001
4697 002172 000005
4698 002174 000204
4699
4700
4701
4702 002176
4703 002176 002220
4704 002200 002221
4705 002202 002222
4706 002204 002223
4707 002206 002224
4708 002210 002324
4709 002212 002416
4710 002214 002524
4711 002216 002646
4712
4713 002220 000
4714 002221
4715 002221 377
4716 002222
4717 002222 252
4718 002223
4719 002223 125
4720 002224
4721 002224
4722 002224 177603 157427 031011
002232 047321 163715 105221
002240 143325 142304
4723 002244 040041 014116 052606
002252 172334 105025 123754
002260 111337 111523
4724 002264 030030 145064 137642
002272 143531 063617 135075
002300 066730 026575
4725 002304 052012 053627 070071
002312 151172 165044 031605

.SBTTL GLOBAL DATA SECTION
.SBTTL DEFAULT MESSAGE DEFINITIONS AND TABLES

:++
: THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
: IN MORE THAN ONE TEST.
:--

:MESSAGE BYTE COUNT TABLE

DMSGCT:
MSG0C: .WORD EMSG0-MSG0 :BYTE COUNT OF MESSAGE #0
MSG1C: .WORD EMSG1-MSG1 :BYTE COUNT OF MESSAGE #1
MSG2C: .WORD EMSG2-MSG2 :BYTE COUNT OF MESSAGE #2
MSG3C: .WORD EMSG3-MSG3 :BYTE COUNT OF MESSAGE #3
MSG4C: .WORD EMSG4-MSG4 :BYTE COUNT OF MESSAGE #4
MSG5C: .WORD EMSG5-MSG5 :BYTE COUNT OF MESSAGE #5
MSG6C: .WORD EMSG6-MSG6 :BYTE COUNT OF MESSAGE #6
OPCNT: .WORD 0 :BYTE COUNT FOR OPERATOR SPEC'D MSG.
MSG8C: .WORD EMSG8-MSG8 :BYTE COUNT OF RECEIVE BUFFER FILL PATTERN
DLLM1C: .WORD DLLM1E-DLLM1 :DLL MSG 1 COUNT
DLLM2C: .WORD DLLM2E-DLLM2 :DLL MSG 2 COUNT

:MESSAGE ADDRESS TABLE

DMSGAD:
MSG0 :ADDRESS OF MESSAGE #0
MSG1 :ADDRESS OF MESSAGE #1
MSG2 :ADDRESS OF MESSAGE #2
MSG3 :ADDRESS OF MESSAGE #3
MSG4 :ADDRESS OF MESSAGE #4
MSG5 :ADDRESS OF MESSAGE #5
MSG6 :ADDRESS OF MESSAGE #6
OPBUF :ADDRESS OF OPERATOR SPEC'D MSG.
MSG8 :ADDRESS OF RECEIVE BUFFER FILL PATTERN
MSG0: .BYTE 000 :MESSAGE OF ALL 0'S
EMSG0:
MSG1: .BYTE 377 :MESSAGE OF ALL 1'S
EMSG1:
MSG2: .BYTE 252 :MESSAGE OF ALTERNATING 1'S
EMSG2:
MSG3: .BYTE 125 :MESSAGE OF ALTERNATING 0'S
EMSG3:
MSG4: :''CCITT'' 512-BIT (VS. 511 BITS) TEST PATTERN
.WORD 177603,157427,031011,047321,163715,105221,143325,142304
.WORD 040041,014116,052606,172334,105025,123754,111337,111523
.WORD 030030,145064,137642,143531,063617,135075,066730,026575
.WORD 052012,053627,070071,151172,165044,031605,166632,016741

4726	002320	166632	016741		
4727	002324			EMSG4:	
4728	002324			MSG5:	;'INTERPROCESSOR TEST PROGRAM'S (ITEP)'' MESSAGE
4729	002324	077577	040444	052040	.ASCII <177><177>/SA THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG./
	002332	042510	050440	044525	
	002340	045503	041040	047522	
	002346	047127	043040	054117	
	002354	045040	046525	042520	
	002362	020104	053117	051105	
	002370	052040	042510	046040	
	002376	055101	020131	047504	
	002404	027107			
4730	002406	005015	077401	077577	.ASCIZ <15><12><001><177><177><177><177>
	002414	000177			
4731	002416			EMSG5:	
4732	002416			MSG6:	;ALPHA-NUMERICS (OR FUTURE COMM TURNAROUND MSG)
4733	002416	022043	021041	023040	.ASCII /#S!' &'()*+,-.0123456789:;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ/
	002424	024047	025051	026053	
	002432	027055	030460	031462	
	002440	032464	033466	034470	
	002446	035472	036474	037476	
	002454	040500	041502	042504	
	002462	043506	044510	045512	
	002470	046514	047516	050520	
	002476	051522	052524	053526	
	002504	054530	132		
4734	002507	057	056133	057135	.ASCIZ ?/[\] ^ _ % ?
	002514	022537	000		
4735	002517			EMSG6:	
4736		002520			.EVEN
4737					
4738					; *****
4739					;THESE THREE STORAGE AREAS MUST NOT BE SEPERATED !!!!
4740					
4741	002520	047045	040445	OPBFPT:	.ASCII /%N%/
4742	002524	000122		OPBUF:	.BLKB 82. ;BUFFER FOR OPERATOR SPEC'D MESSAGES
4743	002646			OPEND:	
4744					
4745					; THE ABOVE THREE LINES MUST BE KEPT TOGETHER
4746					; *****
4747					
4748	002646	033		MSG8:	.BYTE 33 ;RECEIVE BUFFER FILL PATTERN
4749	002647			EMSG8:	
4750					
4751					; DOWN-LINE-LOAD MESSAGE DEFINITIONS
4752					;;;ENTER MOP MODE MESSAGE FORMAT
4753					;;;THE NODE WILL ENTER MAINTENANCE MODE ONLY IF THE PASSWORD MATCHES.
4754	002647	006		DLLM1:	.BYTE 6 ;BINARY CODE FOR MAINTENANCE MODE
4755	002650	000		PASS1:	.BYTE 0 ;PASSWORD BYTE #1 LEGAL VALUE 0 - 255
4756	002651	000		PASS2:	.BYTE 0 ;VALUE IN BYTE 1 IS DUPLICATED HERE
4757	002652	000		PASS3:	.BYTE 0 ;AND HERE
4758	002653	000		PASS4:	.BYTE 0 ;AND HERE.
4759	002654			DLLM1E:	;END ENTER MOP MODE MESSAGE FORMAT
4760					;;;MEMORY LOAD WITH TRANSFER ADDRESS MESSAGE FORMAT
4761	002654	000		DLLM2:	.BYTE 0 ;CODE

```
4762 002655 000 .BYTE 0 ;LOAD NUMBER
4763 002656 006 .BYTE 6 ;LOAD ADDRESS LSB
4764 002657 000 .BYTE 0
4765 002660 000 .BYTE 0
4766 002661 000 .BYTE 0 ;LOAD ADDRESS
4767
4768 : : IMAGE DATA
4769 : :
4770 002662 005037 000006 CLR @#6
4771 002666 012706 001000 MOV #1000,SP
4772 002672 012701 177560 MOV #177560,R1 ;SET UP TTY
4773 002676 010700 MOV PC,R0 ;MAKE ADDR.PIC
4774 002700 062700 000034 ADD #<MSG-.>,R0 ;ADDRESS MSG.
4775 002704 105761 000004 1$: TSTB 4(R1) ;TTY READY?
4776 002710 100375 BPL 1$ ;WAIT TIL YES
4777 002712 112061 000006 MOVB (R0)+,6(R1) ;TYPE A CHAR
4778 002716 001372 BNE 1$ ;KEEP GOING
4779 002720 012737 000026 000024 MOV #26,@#24 ;SET UP POWER FAIL
4780 002726 005037 000026 CLR @#26 ;MAKE SURE T BIT CLAER
4781 002732 000777 BR ;JUMP ON YOURSELF
4782 002734 006412 047502 052117 MSG: .ASCII <12><15>/BOOT MESSAGE WAS RECEIVED SUCCESSFULLY -END OF TEST!!/
002742 046440 051505 040523
002750 042507 053440 051501
002756 051040 041505 044505
002764 042526 020104 052523
002772 041503 051505 043123
003000 046125 054514 026440
003006 047105 020104 043117
003014 052040 051505 020524
003022 041
4783 003023 012 027015 027056 .ASCIZ <12><15>/.....RELOAD PROGRAM..../
003030 051056 046105 040517
003036 020104 051120 043517
003044 040522 027115 027056
003052 000056
4784 003054 006 .BYTE 6 ;NEXT FOUR BYTES CONTAINS TRANSFER ADDRESS
4785 003055 000 .BYTE 0 ;:OF PROGRAM JUST DOWNLINE LOADED.
4786 003056 000 .BYTE 0 ;:::THIS PROGRAM STARTS AT ADDRESS 6.
4787 003057 000 .BYTE 0
4788 003060 DLLM2E: ;END MEMORY LOAD MESSAGE FORMAT
4789
4790 .EVEN
4791
```

:COMMAND LINE BUFFER, DATA LOCATIONS AND MESSAGES FOR ACTION ROUTINES

4793
4794
4795 003060 000122
4796 003202 000000
4797
4798 003204 000000
4799 003206 000000
4800 003210 013232
4801 003212 013245
4802 003214 013362
4803 003216 013447
4804 003220 013474
4805 003222 013553
4806 003224 013631
4807 003226 013721
4808 003230
4809
4810 003230 014057
4811 003232 014102
4812 003234 014135
4813 003236 014166
4814 003240 014220
4815 003242 014257
4816 003244 014316
4817 003246 000000
4818
4819 003250 020342
4820 003252 020402
4821 003254 020432
4822 003256 020467
4823 003260 020532
4824 003262 020566
4825 003264 020620
4826 003266 020663
4827 003270 020717
4828 003272 020751
4829 003274 021003
4830 003276 021035
4831 003300 021065
4832 003302 021114
4833 003304 021146
4834 003306 021176
4835 003310 021226
4836 003312 021255
4837 003314 021307
4838 003316 021333
4839 003320 021365
4840 003322 021410
4841 003324 021463
4842 003326 021513
4843 003330 021544
4844 003332 021627
4845 003334 021664
4846 003336 021721
4847 003340 021756
4848 003342 022042

CMDBUF: .BLKB 82. :BUFFER FOR OPERATOR COMMANDS
KEYWD1: .WORD 0 :THIS LOC WILL =1 IF CLEAR TYPED, 2 FOR SHOW,
: A 4 IF RUN WAS TYPED, 5 IF HELP WAS TYPED
QUALFG: .WORD 0 :THIS LOC HOLDS QUALIFIER VALUE (SIZE OR COPY)
QUALVL: .WORD 0
HLPTAB: .WORD HLP1
.WORD HLP2
.WORD HLP3
.WORD HLP3A
.WORD HLP4
.WORD HLP4A
.WORD HLP5
.WORD HLP6
HLPEND:
:INDEX TABLE FOR REPORT 'RPT>' HELP MESSAGES REV B BY EC
RHLPTB: .WORD RHLP1
.WORD RHLP2
.WORD RHLP3
.WORD RHLP4
.WORD RHLP5
.WORD RHLP6
.WORD RHLP7
RHLPEN: .WORD 0 :END OF REPORT HELP TABLE
:INDEX TABLE FOR DMR BASE TABLE DATA DESCRIPTION MESSAGES REV B BY EC
DMRIND: .WORD DMR000
.WORD DMR001
.WORD DMR002
.WORD DMR003
.WORD DMR004
.WORD DMR005
.WORD DMR006
.WORD DMR007
.WORD DMR010
.WORD DMR011
.WORD DMR012
.WORD DMR013
.WORD DMR014
.WORD DMR015
.WORD DMR016
.WORD DMR017
.WORD DMR020
.WORD DMR021
.WORD DMR022
.WORD DMR023
.WORD DMR024
.WORD DMR025
.WORD DMR026
.WORD DMR027
.WORD DMR030
.WORD DMR031
.WORD DMR032
.WORD DMR033
.WORD DMR034
.WORD DMR035

4849 003344 022107 .WORD DMR036
4850 003346 022154 .WORD DMR037
4851 003350 022212 .WORD DMR040
4852 003352 022245 .WORD DMR041
4853 003354 022303 .WORD DMR042
4854 003356 022351 .WORD DMR043
4855 003360 022405 .WORD DMR044
4856 003362 022441 .WORD DMR045
4857 003364 022472 .WORD DMR046
4858 003366 022540 .WORD DMR047
4859 003370 022602 .WORD DMR050
4860 003372 022630 .WORD DMR051
4861 003374 022664 .WORD DMR052
4862 003376 022711 .WORD DMR053
4863 003400 022744 .WORD DMR054
4864 003402 022766 .WORD DMR055
4865 003404 023041 .WORD DMR056
4866 003406 023114 .WORD DMR057
4867 003410 023136 .WORD DMR060
4868 003412 023170 .WORD DMR061
4869 003414 023222 .WORD DMR062
4870 003416 023272 .WORD DMR063
4871 003420 023342 .WORD DMR064
4872 003422 023376 .WORD DMR065
4873 003424 023432 .WORD DMR066
4874 003426 023475 .WORD DMR067
4875 003430 023540 DMREND: .WORD DMR177 ;NO DMR MESSAGES MUST FOLLOW DMREND
4876
4877 ;INDEX TABLE FOR DMC BASE TABLE DATA DESCRIPTION MESSAGES REV B BY EC
4878 003432 020322 DMCIND: .WORD DMUNKN
4879 003434 020322 .WORD DMUNKN
4880 003436 023604 .WORD DMC002
4881 003440 023625 .WORD DMC003
4882 003442 023662 .WORD DMC004
4883 003444 023723 .WORD DMC005
4884 003446 023756 .WORD DMC006
4885 003450 024013 .WORD DMC007
4886 003452 024050 .WORD DMC010
4887 003454 024103 .WORD DMC011
4888 003456 024125 .WORD DMC012
4889 003460 024147 .WORD DMC013
4890 003462 024206 DMCEND: .WORD DMC377 ;NO DMC MESSAGES MUST FOLLOW DMCEND
4891
4892 003464 014464 014473 014500 SHTYTB: .WORD SHTYP0,SHTYP1,SHTYP2,SHTYP3,SHTYP4,SHTYP5,SHTYP6,SHTYP7
003472 014505 014512 014520
003500 014525 014533
4893
4894 ; THE LIST OF BYTES BELOW ARE THE FIRST BYTES OF THE PREDEFINED MESSAGES
4895 ; USED TO 'SHOW' THE TRANSMIT AND COMPARE BUFFER CONTENTS.
4896
4897 003504 000 377 252 SHTAB: .BYTE 0,377,252,125,203,177,043
003507 125 203 177
003512 043
4898 003513 SHTEND:
4899 003514 .EVEN
4900

4901 003514 014544
4902 003516 014554
4903 003520 014565
4904 003522 014575
4905 003524 014604
4906 003526 014621
4907 003530 014626
4908
4909 003532 014635
4910 003534 014645
4911 003536 014656
4912 003540 014664
4913 003542 014677
4914

MODES: .WORD M00 ;ADDRESSES OF MODE TYPES IN ASCII
.WORD M01
.WORD M02
.WORD M03
.WORD M04
.WORD M05
.WORD M06

LOOPS: .WORD LP0 ;ADDRESSES OF LOOP TYPES IN ASCII
.WORD LP1
.WORD LP2
.WORD LP3
.WORD LP4

;COMMAND LINE TRAVERSE LOCATIONS (USED BY 'P\$TRV')

4915
4916
4917 003544 000000
4918 003546 000000
4919 003550 000000
4920 003552 000000
4921 003554 000000
4922 003556 000000
4923 003560 000
4924 003561 000
4925

P\$BUFA: .WORD 0 ;LOC. TO HOLD ADDR. OF CMD LINE BUFFER
P\$TREE: .WORD 0 ;LOC. TO HOLD ADDR. OF PARSING TREE
P\$ACT: .WORD 0 ;LOC. TO HOLD ADDR. OF ACTION ROUTINE
P\$CNT: .WORD 0 ;LOC. TO BE A COUNTER LOCATION
P\$NUM: .WORD 0 ;LOC. TO HOLD NUMERIC VALUE FROM PARSE
P\$RADX: .WORD 0 ;LOC. TO HOLD RADIX USED(LO) AND +/- (HI BYTE)
P\$NNUF: .BYTE 0 ;RETURN =0 IF ENOUGH OF COMMAND FOUND
P\$GDBD: .BYTE 0 ;RETURN CODE 0 IF NO ERROR FOUND

4927		.SBTTL	MESSAGE BUFFERS AND POINTER TABLES
4928			
4929	003562	001000	TXBUF: .BLKB BUFLIM :TRANSMITTER BUFFERS
4930	004562	001000	RXBUF: .BLKB BUFLIM :RECEIVER BUFFERS
4931	005562	001000	CMPBUF: .BLKB BUFLIM :COMPARISON BUFFERS
4932	006562	000264	PTRTAB: .BLKW 180. :TABLE FOR MESSAGE ADDRS. & BYTE COUNTS
4933	007332		PTREND: :END OF MSG. PTR. TABLE
4934			
4935	007332	000000	RXPTR: .WORD 0 :RECEIVER MESSAGE POINTER
4936	007334	000000	TXPTR: .WORD 0 :TRANSMITTER BUFFER POINTER
4937	007336	000000	CMPPTR: .WORD 0 :COMPARISON BUFFER POINTER
4938	007340	000000	CMPTOT: .WORD 0 :CMP MSG TOTAL
4939	007342	000000	CTOTCC: .WORD 0 :COMPARE BUFFER CHAR. COUNT
4940	007344	000000	CCURAD: .WORD 0 :CURRENT ADDR OF CMP BUFF TO ADD AT
4941			
4942	007346	000000	DVTXA: .WORD 0 :DEVICE TX ADDR
4943	007350	000000	DVTCC: .WORD 0 :DEVICE TX CHAR COUNT
4944	007352	000000	DVTCT: .WORD 0 :DEVICE TX MESSAGE COUNT
4945	007354	000000	TXMTOT: .WORD 0 :TX MSG TOTAL
4946	007356	000000	TTOTCC: .WORD 0 :TX BUFFER CHAR. COUNT
4947	007360	000000	TCURAD: .WORD 0 :CURRENT ADDR. OF TX BUFF TO ADD AT
4948			
4949	007362	000000	DVRXA: .WORD 0 :DEVICE RX ADDR
4950	007364	000000	DVRCC: .WORD 0 :DEVICE RX CHAR COUNT
4951	007366	000000	DVRCT: .WORD 0 :DEVICE RX MESSAGE COUNT
4952	007370	000000	RXMTOT: .WORD 0 :RX MSG TOTAL
4953			
4954	007372	000000	LCNCT: .WORD 0 :NUMBER OF OPERATOR AWAKE MSGS
4955	007374	000000	NOBUF: .WORD 0 :NUMBER OF NO BUFFS
4956	007376	000000	PSCNT: .WORD 0 :PASS COUNTER
4957	007400	000000	ERRCNT: .WORD 0 :ERROR COUNTER
4958	007402	000000	STADD: .WORD 0 :START ADDR.
4959	007404	000000	ENADD: .WORD 0 :END ADDR. FOR DUMP
4960	007406	000000	BYTBIT: .WORD 0 :BYTE BIT FOR DUMP ROUTINE
4961			
4962			:OTHER MESSAGE RELATED STORAGE LOCATIONS
4963	007410	000000	MSGTYP: .WORD 0 :TYPE OF DATA 0=0'S,1=1'S,2=10'S,3=01'S :4=CCITT,5=QUICK FOX,6=ALPHA/NUM,7=OPER
4964			
4965	007412	000000	CURCC: .WORD 0 :TX/RX/CMP CHAR COUNT
4966	007414	000000	CPTRR: .WORD 0 :CURRENT RX POINTER
4967	007416	000000	CPTR: .WORD 0 :CURRENT POINTER
4968	007420	000000	CURADD: .WORD 0 :CURRENT TX/RX/CMP START ADDD
4969	007422	000000	TOTCC: .WORD 0 :TOTAL CHAR COUNT NOT MORE THEN 'BUFLIM'
4970	007424	000000	OFFSET: .WORD 0 :OFFSET COUNT
4971	007426	000000	TEMP: .WORD 0 :TEMPORARY LOCATIONS (USED A LOT)
4972	007430	000000	TEMP1: .WORD 0
4973	007432	000000	TEMP2: .WORD 0
4974	007434	000000	TEMP3: .WORD 0
4975	007436	000000	TEMP4: .WORD 0
4976	007440	000000	TEMP5: .WORD 0
4977	007442	000000	CONOTM: .WORD 0 :CONTROL OUT ERROR MSG. ADDRESS
4978	007444	000000	CONTIN: .WORD 0 :WORD FOR CONTORL IN
4979	007446	000	GOOD: .BYTE 0 :BYTE TO HOLD EXPECTED MESSAGE DATA BYTE FOR ERR REPORT
4980	007447	000	BAD: .BYTE 0 :BYTE TO HOLD RECEIVED MESSAGE DATA BYTE FOR ERR REPORT
4981	007450	000000	INDEX: .WORD 0 :WILL CONTAIN POINTER TO DMC OR DMR MESSAGES
4982	007452	000000	INDEXE: .WORD 0 :WILL CONTAIN POINTER TO L/T OF DMC OR DMR MESSAGES


```

4983 007454 000000      BEND:  .WORD  0      ;LAST LOCATION IN BASE TABLE TO BE PRINTED
4984
4985
4986
4987      ;MORE INDEPENDENT CODE STORAGE LOCATIONS
4988 007456 000000      BDATA:  .WORD  0      ;POINTER TO BASE TABLE
4989 007460 000000      LOGUNT: .WORD  0      ;LOC. TO HOLD LOGICAL UNIT NUMBER
4990 007462 000000      PCADD:  .WORD  0      ;LOC. HOLD PC OF CALLIN ROUTINE
4991 007464 000000      DCLFLG: .WORD  0      ;CLEANUP AND EXIT FLAG. 1=DO CLEANUP ROUTINE&EXIT
4992 007466 000000      RESFLG: .WORD  0      ;LOC TO HOLD FLAG (-1) THAT A RESTART WAS GIVEN
4993 007470 000000      MODTYP: .WORD  0      ;DCLT MODE OF OPERATION TYPE
4994      ; (0=REC-ONLY, 1=TX-ONLY, 2=PASSIVE-LOOPBK,
4995      ; 3=ACTIVE-LOOPBK, 4=DOWN L.L., 5=TALK, 6=LISTEN)
4996 007472 000000      MLTYP:  .WORD  0      ;MAINTENANCE LOOP TYPE (0=NONE, 1=INTERNAL TTL,
4997      ; 2=CABLE, 3=MODEM-ANALOG LOOPBK (LOCAL),
4998      ; 4=MODEM-DIGITAL LOOPBK (REMOTE), 5=MOP)
4999 007474 000000      FHDPLX: .WORD  0      ;FULL OR HALF DUPLEX FLAG (1=FULL FROM P-TABLE)
5000 007476 000002      PARAM:  .WORD  2      ;PROGRAM PARAMETERS
5001      ; BIT0= STATUS MSGS TO OPR PRINTED (1=YES)
5002      ; BIT1= DATA CHECKING DONE ON RCVD MSGS (1=YES)
5003      ; BIT2= ECHO (TRANSMIT) RCV'D MSG.(PASSIVE)(1=YES)
5004      ; BIT3= SPARE
5005      ; BIT4= CRC CALC./CHECK DONE (1=YES)
5006      ; BIT5= PROTOCOL EMULATION (1=YES)
5007      ; BIT6= SPARE
5008 007500 000000      RPASS:  .WORD  0      ;PASS NUMBER FROM RUN COMMAND
5009 007502 000000      FLAG:   .WORD  0      ;DEVICE FLAG WORD
5010
5011      ;MODE DISPATCH TABLE
5012 007504 040572      MODE:   .WORD  RXONLY  ;RX ONLY DISPATCH
5013 007506 040624      .WORD  TXONLY  ;TX ONLY DISPATCH
5014 007510 040664      .WORD  PLCK    ;PASSIVE LOOP BACK DISP
5015 007512 040720      .WORD  ALCK    ;ACTIVE LOOP BACK DISP
5016 007514 042050      .WORD  DLL     ;DOWN LINE LOAD DISP
5017 007516 042670      .WORD  TALCK   ;TALK MODE DISPATCH
5018 007520 043110      .WORD  LISCK   ;LISTEN MODE DISPATCH
5019
5020
5021      ;SBTTL      CLOCK TABLES, EVENT LOG AND POINTERS
5022 007522 000000      CLKCSR: .WORD  0      ;CLOCK CSR ADDRESS
5023 007524 000000      CLKBR:  .WORD  0      ;CLOCK INTERRUPT LEVEL
5024 007526 000000      CLKVEC: .WORD  0      ;CLOCK INTERRUPT VECTOR
5025 007530 000074      CLKHZ:  .WORD  60.    ;CLOCK'S HERTZ RATE
5026 007532 000000      CLKEN:  .WORD  0      ;CLOCK'S CSR VALUE TO INTRPT. ENABLE IT
5027
5028 007534 000000      TIMMIN: .WORD  0      ;PLACE TO KEEP TIME-SINCE-START
5029 007536 000000      TIMSEC: .WORD  0
5030 007540 000000      TIMTCK: .WORD  0      ;PLACE TO KEEP # OF TICKS/SEC
5031
5032 007542 000000      TIMER1: .WORD  0      ;EVENT TIMER #1 (TICKS)
5033 007544 000000      TIMER2: .WORD  0      ;EVENT TIMER #2 (TICKS)
5034 007546 000000      TIMERS: .WORD  0      ;EVENT TIMER #3 (SECONDS)
5035

```

```
5037 ;EVENT LOG TABLE AND ITS NEXT ENTRY POINTER
5038 007550 007552 EVTPTN: .WORD EVTLOG ; POINTER TO NEXT FREE SPACE IN EVENT LOG
5039 007552 000341 EVTLOG: .BLKW 225. ; EVENT LOG BUFFER
5040 010454 000001 EVTEND: .BLKW 1. ; APPROXIMATE END OF EVENT TABLE (ALLOWS CIRCULAR QUE)
5041
5042 .SBTTL MODEM DATA SECTION
5043
5044 010456 000000 MODS: .WORD 0 ;MODEM STATUS
5045
5046 ;TABLE OF MODEM SIGNAL BIT DEFINITIONS
5047
5048 010460 000004 MOBITS: .WORD CTS ;CLEAR TO SEND (CIRCUIT CB)
5049 010462 000010 .WORD DSR ;DATA SET READY (CIRCUIT CC)
5050 010464 000001 .WORD DCD ;DATA CARRIER DETECT (CIRCUIT CF)
5051 010466 000040 .WORD RTS ;REQUEST TO SEND (CIRCUIT CA)
5052 010470 000200 .WORD RI ;RING INDICATOR (CIRCUIT CE)
5053 010472 040000 .WORD SQD ;SIGNAL QUALITY DETECT (CIRCUIT CG)
5054 010474 001000 .WORD TM ;MODEM IN TEST MODE (RS 449 ONLY CIRCUIT TM)
5055 010476
5056
5057 ;TABLE OF ADDRESSES OF MODEM SIGNAL MESSAGE POSITIONS
5058
5059 010476 017240 MOMSGS: .WORD EVMCTS ;CLEAR TO SEND (CIRCUIT CB)
5060 010500 017244 .WORD EVMSDR ;DATA SET READY (CIRCUIT CC)
5061 010502 017250 .WORD EVMDCD ;DATA CARRIER DETECT (CIRCUIT CF)
5062 010504 017254 .WORD EVMRTS ;REQUEST TO SEND (CIRCUIT CA)
5063 010506 017260 .WORD EVMRI ;RING INDICATOR (CIRCUIT CE)
5064 010510 017264 .WORD EVMSQD ;SIGNAL QUALITY DETECT (CIRCUIT CG)
5065 010512 017270 .WORD EVMTM ;MODEM IN TEST MODE (RS 449 ONLY CIRCUIT TM)
5066
5067 ;TABLE OF ADDRESSES OF EVENT DESCRIPTION MESSAGES
5068 ; ORDER CORRESPONDS TO MESSAGE TYPE VALUES
5069
5070 010514 015625 EVTLST: .WORD EDTXQ ;TRANSMIT MESSAGE QUEUED
5071 010516 015651 .WORD EDTXC ;TRANSMIT OF MESSAGE COMPLETE
5072 010520 015700 .WORD EDRXQ ;RECEIVE MESSAGE SPACE QUEUED
5073 010522 015725 .WORD EDRXC ;MESSAGE RECEIVED - RECEIVE COMPLETE
5074 010524 015753 .WORD EDDER ;DEVICE INFORMATION
5075 010526 016020 .WORD EDDVI ;DEVICE INITIALIZE STARTED
5076 010530 015770 .WORD EDDCK ;DATA COMPARISON DONE
5077 010532 014635 .WORD LPO ;NULL STRING
5078 010534 016046 .WORD EDDLE ;DATA COMPARE LENGTH ERROR
5079 010536 016103 .WORD EDDDE ;DATA COMPARE DATA ERROR
5080 010540 016136 .WORD EDEOP ;END OF PASS
5081
5082 :::FOLLOWING TABLE USED IN DOWNLINE LOAD ROUTINE.
5083 :::CONTAINS POINTERS TO ASCIZ DEVICE DESCRIPTIONS
5084 010542 020217 DLLIND: .WORD DPM
5085 010544 020222 .WORD DUM
5086 010546 020225 .WORD DLM
5087 010550 020230 .WORD DQM
5088 010552 020233 .WORD DAM
5089 010554 020236 .WORD DUPM
5090 010556 020242 .WORD DMCM
5091 010560 020246 .WORD DNM
5092 010562 020251 .WORD DLVM
```

5093 010564 020255 .WORD DMPM
5094 010566 020261 .WORD DTEM
5095 010570 020265 .WORD DVM
5096 010572 020270 .WORD DZM
5097 010574 020273 .WORD UNKM
5098 010576 020303 .WORD KDPM
5099 010600 020307 .WORD KDZM
5100 010602 020313 .WORD KLM
5101 010604 020316 .WORD DMVM

;LOCATIONS USED DURING EVENT REPORTING

5102
5103
5104
5105 010606 000000 EVTSEC: .WORD 0 ;TEMPORARY LOCS TO KEEP EVENT TIME WHILE REPORTING
5106 010610 000000 EVTMIN: .WORD 0
5107 010612 000000 EVTTCK: .WORD 0
5108 010614 000000 EVTADD: .WORD 0 ;TEMP. LOC. TO HOLD ADDRESS DURING EVENT REPORTING
5109 010616 000000 EVTACT: .WORD 0 ;
5110 010620 000000 EVTTMP: .WORD 0 ;
; " " BYTE COUNT " " "
; " " OTHER DATA " " "

;REPORT CODING DISPATCH TABLE

5111
5112
5113
5114 010622 031014 RPTDSP: .WORD RPTTXQ ;TRANSMIT QUEUED ENTRY DECODING
5115 010624 031014 .WORD RPTTXQ ;TRANSMIT COMPLETE ENTRY DECODING
5116 010626 031014 .WORD RPTTXQ ;RECEIVER QUEUED ENTRY DECODING
5117 010630 031014 .WORD RPTTXQ ;RECEIVER COMPLETE ENTRY DECODING
5118 010632 031066 .WORD RPTDER ;DEVICE ERROR ENTRY DECODING
5119 010634 031162 .WORD RPTDVI ;DEVICE INIT ENTRY DECODING
5120 010636 031356 .WORD RPTDCK ;DATA COMPARISON ENTRY DECODING
5121 010640 030642 .WORD RPT ;PLACE HOLDER
5122 010642 031356 .WORD RPTDLE ;DATA COMPARISON LENGH ERROR
5123 010644 031302 .WORD RPTDDE ;DATA COMPARISON DATA ERROR
5124 010646 031226 .WORD RPTTEOP ;END OF PASS

5125
5126
5127 010650 000000 DEV1: .WORD 0 ;TEMP LOCS TO HOLD DATA FOR EVENT REPORTING
5128 010652 000000 DEV2: .WORD 0 ; AND SHOW MODE,... SUBROUTINE
5129 010654 000000 DEV3: .WORD 0
5130 010656 000000 DEV4: .WORD 0
5131

5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148 010660
5149
5150
5151 010660
5152 010664
5153 010670
5154 010672
5155 010706
5156 010710
5157 010724
5158 010726
5159 010742
5160 010744
5161 010756
5162 010762
5163 010776
5164 011002
5165 011016
5166 011022
5167 011026
5168 011040
5169 011044
5170 011056
5171 011062
5172
5173
5174
5175 011064
5176 011070
5177 011104
5178 011110
5179 011126
5180 011132
5181 011150
5182 011154
5183 011172
5184 011176
5185 011214
5186 011220
5187 011244
5188 011250

.SBTTL COMMAND LINE ACTION TREE

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

```
-----  
: ! ACTION ! CHAR CODE !  
-----  
: ! MISS DISPLACEMENT ! ONLY IF 'MISS' ARGUMENT DEFINED  
-----  
: ! NEXT NODE DISPLMNT ! ONLY IF 'ASCII' ARGUMENT DEFINED  
-----  
: ! ASCIIZ MATCH STRING ! ONLY IF 'ASCII' ARGUMENT DEFINED  
: ! (.EVEN) !  
-----
```

CLITRE:

;FIRST KEYWORD

```
N10$: CLI CLISPA,0,N10$ ;SKIP ANY LEADIN SPACES  
CLI <'?'>,HLP,N42$ ;IS THE FIRST NON-SP CHAR A '?'  
N42$: CLI CLIEXI,0 ; IF YES DO 'HLP' AND EXIT  
CLI CLISTR,HLP,N43$,<'HELP'> ;ELSE, IS FIRST WORD A 'HELP'  
N43$: CLI CLIEXI,0 ; IF YES DO 'HLP' AND EXIT  
CLI CLISTR,PRNT,N44$,<'PRINT'> ;ELSE, IS FIRST WORD A 'PRINT'  
N44$: CLI CLIEXI,0 ; IF YES DO 'PRINT' AND EXIT  
CLI CLISTR,EXIT,N45$,<'EXIT'> ;ELSE, IS FIRST WORD AN 'EXIT';REV B BY E  
N45$: CLI CLISTR,RUN,N46$,<'RUN'> ; IF YES DO 'EXIT' AND EXIT  
CLI CLIBR,0,N80$ ;ELSE, IS FIRST WORD A 'RUN'  
N46$: CLI CLISTR,NOTNUF,N40$,<'DUMP'> ; IF YES DO 'RUN' & GOTO N80$  
CLI CLIBR,0,N50$ ;ELSE, IS FIRST WORD A 'DUMP'  
N40$: CLI CLISTR,CLEAR,N20$,<'CLEAR'> ; IF YES GOTO N80$  
CLI CLIBR,NOTNUF,N100$ ;ELSE, IS FIRST WORD A 'CLEAR'  
N20$: CLI <'S'>,NOTNUF,N30$ ; IF YES DO 'CLR' & GOTO N100$  
CLI CLISTR,SHOW,N25$,<'HOW'> ;ELSE, IS FIRST CHAR. A 'S'  
N25$: CLI CLIBR,0,N100$ ; IF YES IS REST OF WORD 'HOW'  
CLI CLISTR,0,N30$,<'ET'> ; IF YES, DO 'SHOW',BR N100$  
N30$: CLI CLIBR,0,N110$ ; ELSE, IS REST OF WORD 'ET'  
CLI CLIERR,0 ; IF YES, DO 'SET', BR N110$  
;OTHERWISE 'ILL CMD' - EXIT
```

;SECOND KEYWORD (MODE=) FOR RUN COMMAND

```
N80$: CLI CLISPA,0,N30$ ;SKIP LEADING SPS, IF NONE-ERR  
N81$: CLI CLISTR,NOTNUF,N30$,<'MODE'> ;IS NEXT WORD 'MODE='  
CLI <'='>,0,N30$ ; IF NO, IT'S WRONG -ERR -EXIT  
N82$: CLI CLISTR,ATVMOD,N82$,<'ACTIVE'> ;IS NEXT WORD 'ACTIVE'  
CLI CLIBR,0,N115$ ; IF YES, DO 'ACTIVE',BR N115$  
N83$: CLI CLISTR,PASMOD,N83$,<'PASSIVE'> ;IS NEXT WORD 'PASSIVE'  
CLI CLIBR,0,N115$ ; IF YES, DO 'PASSIVE',BR N115$  
N84$: CLI CLISTR,RECMOD,N84$,<'RECEIVE'> ;IS NEXT WORD 'RECEIVE'  
CLI CLIBR,0,N115$ ; IF YES, DO 'RECVE',BR N115$  
N85$: CLI CLISTR,LISMOD,N85$,<'LISTEN'> ;IS NEXT WORD 'LISTEN'  
CLI CLIBR,0,N115$ ; IF YES, DO 'LISTEN',BR N115$  
N86$: CLI CLISTR,DLLMOD,N86$,<'DOWNLINELOAD'> ;IS NEXT WORD 'DOW'  
CLI CLIBR,0,N115$ ; IF YES, DO 'DWNLL',BR N115$  
CLI <'T'>,0,N30$ ;IS NEXT CHAR A 'T'
```

```

5189 011254      CLI      CLISTR,TRAMOD,N87$,<'RANSMIT'>  ; IS REST OF WORD 'RANSMIT'
5190 011272      CLI      CLIBR,0,N115$                          ; IF YES, DO 'TRANSM',BR N115$
5191 011276      N87$:  CLI      CLISTR,TALMOD,N30$,<'ALK'>      ; IS REST OF WORD 'ALK'
5192 011310      CLI      CLIBR,0,N115$                          ; IF YES, DO 'TALK',BR N115$
5193                                     ; IF NO, ERROR - EXIT
5194
5195      ;SECOND KEYWORD (FOR CLEAR OR SHOW)
5196 011314      N100$: CLI      CLISPA,0,N30$                          ;SKIP LEADING SPACES, NONE=ERR
5197 011320      N102$: CLI      CLISTR,CSHEXP,N104$,<'EXPECT'> ;IS NEXT WORD 'EXPE...'
5198 011336      CLI      CLIEXI,0                          ; IF YES, DO CLR-EXP,EXIT
5199 011340      N104$: CLI      CLISTR,CSHTRN,N30$,<'TRANSMIT'> ;IS NEXT WORD 'TRANS...'
5200 011360      CLI      CLIEXI,0                          ; IF YES, DO CLR-TRN,EXIT
5201                                     ; IF NO - ERROR - EXIT
5202
5203
5204      ;SECOND KEYWORD (FOR SET)
5205 011362      N110$: CLI      CLISPA,0,N30$
5206 011366      N111$: CLI      CLISTR,SETEXP,N112$,<'EXPECT'>
5207 011404      CLI      CLIBR,0,N120$
5208 011410      N112$: CLI      CLISTR,SETTRN,N30$,<'TRANSMIT'>
5209 011430      CLI      CLIBR,0,N120$
5210
5211      ;GET ADDRESSES FOR DUMP COMMAND
5212 011434      N50$:  CLI      CLIALP,0,N51$
5213 011440      N51$:  CLI      CLISPA,0,N52$
5214 011444      N52$:  CLI      CLIOCT,DMP$S,N30$
5215 011450      CLI      <'>,NOTNUF,N125$
5216 011454      CLI      CLIOCT,DMPE,N30$
5217 011460      CLI      <'>,NOTNUF,N125$
5218 011464      CLI      <'B>,DMPQ,N30$
5219 011470      CLI      CLIBR,0,N125$
5220
5221      ;QUALIFIERS FOR THE RUN COMMAND
5222 011474      N115$: CLI      CLIALP,0,N114$
5223 011500      N114$: CLI      <'>,NOTNUF,N125$
5224 011504      CLI      CLISTR,NO,N116$,<'NO'>
5225 011516      N116$: CLI      <'C>,0,N117$
5226 011522      CLI      CLISTR,CHECK,N117$,<'HECK'>
5227 011536      CLI      CLIBR,0,N115$
5228
5229
5230      ;N113$: CLI      CLISTR,CRC,N30$,<'RC16'>
5231      ;      CLI      CLIBR,0,N115$
5232
5233
5234
5235
5236
5237
5238 011542      N117$: CLI      CLISTR,STATUS,N118$,<'STATUS'>
5239 011560      CLI      CLIBR,0,N115$
5240 011564      N118$: CLI      CLISTR,ECHO,N130$,<'ECHO'>
5241 011600      CLI      CLIBR,0,N115$
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256 011604      N130$: CLI      CLISTR,0,N131$,<'PASS'>
5257 011620      CLI      CLIBR,0,N150$
5258 011624      N131$: CLI      CLISTR,0,N132$,<'LOOP'>
5259 011640      CLI      CLIBR,0,N140$
5260
5261 011644      N132$: CLI      CLISTR,MOSC,N30$,<'MODEM'>      ;MODEM ACTION
```

```
5262 011660          CLI      CLIBR,0,N115$          ;;ADDED BY EC
5263
5264
5265 011664          ;GET MESSAGE TYPE FOR SET MESSAGE COMMANDS
5266 N120$: CLI      <'=>,0,N30$
5267
5267 011670          ; LOOK FOR DEFAULT MESSAGE NAME
5268 N60$: CLI      CLISTR,CMSG1,N61$,<'ONES'>
5269 011704          CLI      CLIBR,0,N121$
5270 011710          N61$: CLI      CLISTR,CMSG0,N62$,<'ZEROES'>
5271 011726          CLI      CLIBR,0,N121$
5272 011732          N62$: CLI      CLISTR,CMSG2,N63$,<'1ALT'>
5273 011746          CLI      CLIBR,0,N121$
5274 011752          N63$: CLI      CLISTR,CMSG3,N64$,<'0ALT'>
5275 011766          CLI      CLIBR,0,N121$
5276 011772          N64$: CLI      CLISTR,CMSG5,N65$,<'ITEP'>
5277 012006          CLI      CLIBR,0,N121$
5278 012012          N65$: CLI      CLISTR,CMSG4,N66$,<'CCITT'>
5279 012026          CLI      CLIBR,0,N121$
5280 012032          N66$: CLI      CLISTR,CMSG6,N67$,<'ALPHA'>
5281 012046          CLI      CLIBR,0,N121$
5282 012052          N67$: CLI      CLISTR,SETET,N68$,<'TRANSMIT'> ;REV B BY EC
5283 012072          CLI      CLIBR,0,N125$          ;REV B BY EC
5284
5285          ; LOOK FOR QUOTED MESSAGE
5286 012076          N68$: CLI      <'>,OPRMSG,N30$
5287 012102          N70$: CLI      <'>,ENDQO,N71$
5288 012106          CLI      CLIBR,0,N121$
5289 012112          N71$: CLI      CLISPA,0,N72$
5290 012116          N72$: CLI      CLIALN,0,N73$          ;ONLY A-Z,SP,TAB, OR 0-9 BETWEEN ''S
5291 012122          CLI      CLIBR,0,N70$
5292 012126          N73$: CLI      CLIERR,BADCHR          ;PRINT ERROR IF NONE LEGAL CHAR FOR ''S
5293
5294          ;GET QUALIFIERS (SIZE OR COPY) FOR SET MESSAGE COMMANDS
5295 012130          N121$: CLI      CLIALP,0,N123$
5296 012134          N123$: CLI      <'>,NOTNUF,N125$
5297 012140          CLI      CLISTR,SIZE,N122$,<'SIZE'>
5298 012154          CLI      CLIBR,0,N126$
5299 012160          N122$: CLI      CLISTR,QCOPY,N30$,<'COPY'>
5300 012174          CLI      CLIBR,0,N126$
5301
5302          ;NUMER FOR SIZE OR COPY
5303 012200          N126$: CLI      <'=>,0,N30$
5304 012204          CLI      CLIDEC,NUM,N30$
5305 012210          CLI      CLIBR,0,N121$
5306
5307          ;GET MAINTENANCE LOOP TYPE FOR RUN 'LOOP' QUALIFIER
5308 012214          N140$: CLI      <'=>,0,N30$
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319 012220          N141$: CLI      CLISTR,TTLLOP,N142$,<'INTERNAL TTL'>
5320 012242          CLI      CLIBR,0,N115$
5321 012246          N142$: CLI      CLISTR,CBLLOP,N143$,<'CABLE'>
5322 012262          CLI      CLIBR,0,N115$
5323 012266          N143$: CLI      CLISTR,LMDLOP,N144$,<'LOCALMODEM'>
5324 012310          CLI      CLIBR,0,N115$
5325 012314          N144$: CLI      CLISTR,RMDLOP,N30$,<'REMOTEMODEM'>
```

5326 012336
5327
5328
5329 012342
5330 012346
5331 012352
5332
5333
5334
5335
5336 012356
5337

CLI CLIBR,0,N115\$
:GET LINE NUMBER FOR 'PASS' RUN QUALIFIER
N150\$: CLI <'=>,0,N30\$
CLI CLIDEC,PASC,N30\$
CLI CLIBR,0,N115\$
:END-OF-LINE
N125\$: CLI CLIEXI,0

5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5387
5397
5398

;DEVICE DEPENDENT STORAGE LOCATIONS FOR
; CURRENT DEVICE PARAMETERS

SELO:
BSEL0: .WORD 0
BSEL1: .WORD 0
SEL2:
BSEL2: .WORD 0
BSEL3: .WORD 0
SEL4:
BSEL4: .WORD 0
BSEL5: .WORD 0
SEL6:
BSEL6: .WORD 0
BSEL7: .WORD 0

;ADDRESSES OF REGISTERS SELO THRU BSEL7

INVEC: .WORD 0
OUTVEC: .WORD 0
INTPRI: .WORD 0
OPTYP: .WORD 0

;INPUT INTERRUPT VECTOR ADDRESS
;OUTPUT INTERRUPT VECTOR ADDRESS
;INTERRUPT PRIORITY
;DEVICE OPTION TYPE (0=DMC,5=DMR-DMC MODE
;7=DMR).

; ERR TBL

5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
(4)
(3)
(3)
(2)
5413
5419
5428
5429
5430
5431
5432
(4)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(2)
5433
5434
5441
5442
5443

.SBTTL GLOBAL TEXT SECTION
:++
: THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
: MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
: MORE THAN ONE TEST.
:--

.SBTTL DEVICE SUPPORTED
: NAMES OF DEVICES SUPPORTED BY PROGRAM
: DEVTYP <DMR,DMC-11>

012410			
012410	046504	026122	046504
012410	026503	030461	000
012416	012424		

L\$DVTYP::
.ASCIZ /DMR,DMC-11/
.EVEN

.SBTTL PROGRAM IDENTIFICATION
: TEST DESCRIPTION
:

012424			
012424	055103	046103	041113
012432	020060	046504	026122
012440	042040	041515	030455
012446	020061	040504	040524
012454	041440	046517	027115
012462	046040	047111	020113
012470	042524	052123	000
012476			

DESCRIPT <CZCLKBO DMR, DMC-11 DATA COMM. LINK TEST>
L\$DESC::

.ASCIZ /CZCLKBO DMR, DM

.EVEN

.EVEN

					.SBTTL	GLOBAL FORMAT STATEMENTS, MESSAGES, AND ASCII INFO
5445						
5449						
5450	012476	041504	052114	000076	CLISPM:	.ASCIZ /DCLT>/
5451	012504	050122	037124	000040	CLISRP:	.ASCIZ /RPT> / ;REV B BY EC
5452	012512	047045	040445	044477	CLIERM:	.ASCIZ /%NZA?ILL CMD-BAD SYNTAX?/
5453	012542	047045	040445	044477	CLINUF:	.ASCIZ /%NZA?INCMPLTE CMD?/
5454	012565	045	022516	037501	CLINBG:	.ASCIZ /%NZA?NUM TOO BIG?/
5455	012607	045	022516	037501	CLIBRX:	.ASCIZ /%NZA?BAD RADIX?/
5456	012627	045	022516	037501	CLIBDL:	.ASCIZ /%NZA?'LOOP' VALID ONLY IN ACTIVE?/
5457	012671	045	022516	037501	CLINPS:	.ASCIZ /%NZA?'ECHO' VALID ONLY IN PASSIVE?/
5458	012734	047045	040445	044477	CLIBCR:	.ASCIZ /%NZA?ILL CHR- 'A-Z,0-9,SP,TAB' ONLY?/
5459	013001	045	022516	037501	CLISEO:	.ASCIZ /%NZA?'SIZE=0' NOT VALID?/
5460	013032	047045	040445	052077	CLIPW:	.ASCIZ /%NZA?TRANSMIT & EXPECT LIST MUST BE IDENTICAL FOR LOOP?/ ;REV B BY EC
5461	013122	040523	052124	046105	DLLQ1:	.ASCIZ /SATELLITE PASSWORD= / ;REV B BY EC
5462	013147	045	022516	052101	HLP0:	.ASCIZ /%NZA?THIS IS DCLT. TYPE 'H' OR '?' FOR DETAILS/
5463	013225	045	022516	000124	HLPF:	.ASCIZ /%NXT/
5464	013232	041504	052114	041440	HLP1:	.ASCIZ /DCLT CMDS:/
5465	013245	040	046103	040505	HLP2:	.ASCII / CLEAR OR SHOW EXPECTLIST OR TRANSMITLIST/<15><12>
5466	013321	040	051120	047111		.ASCII / PRINT/<15><12>
5467	013331	040	054105	052111		.ASCII / EXIT/<15><12> ;REV B BY EC
5468	013340	042040	046525	020120		.ASCII ? DUMP START-END/B?
5469	013362	051440	052105	042440	HLP3:	.ASCIZ ? SET EXPECTMSG OR TRANSMITMSG=TYPE/SIZE=N OR /COPY=N?
5470	013447	040	042523	020124	HLP3A:	.ASCIZ / SET EXPECT=TRANSMIT/ ;REV B BY EC
5471	013474	020040	052040	050131	HLP4:	.ASCIZ ? TYPE=ONES,ZEROES,1ALT,0ALT,ITEP,CCITT,ALPHA?
5472	013553	040	020040	020040	HLP4A:	.ASCIZ / OR 'OPR SPCD=A-Z,SP,TAB,0-9 IN QUOTES'/
5473	013631	040	052522	020116	HLP5:	.ASCIZ ? RUN MODE=MTYP/LOOP=LTYP/CHECK,STATUS,ECHO,MODEM,PASS=N?
5474	013721	040	020040	052115	HLP6:	.ASCII / MTYP=TRAN,REC,ACT,PAS,TAL,LIS,DOWN/<15><12>
5475	013770	020040	046040	054524		.ASCIZ / LTYP=INT,CAB,LOC,REM/
5476	014020	047045	040445	054524	RHLP0:	.ASCIZ /%NZA?TYPE 'H' OR '?' FOR HELP !/ ;REV B BY EC
5477	014057	104	046103	020124	RHLP1:	.ASCIZ /DCLT REPORT CMDS :/ ;REV B BY EC
5478	014102	047514	020107	020055	RHLP2:	.ASCIZ /LOG - PRINT DCLT EVENT LOG/ ;REV B BY EC
5479	014135	105	044530	020124	RHLP3:	.ASCIZ /EXIT - EXIT REPORT LEVEL/ ;REV B BY EC
5480	014166	042510	050114	026440	RHLP4:	.ASCIZ /HELP - PRINT THIS MESSAGE/ ;REV B BY EC
5481	014220	040502	042523	042457	RHLP5:	.ASCIZ !BASE/ERROR - PRINT ONLY ERRORS! ;REV B BY EC
5482	014257	102	051501	027505	RHLP6:	.ASCIZ !BASE/FULL - PRINT ENTIRE TABLE! ;REV B BY EC
5483	014316	040502	042523	047457	RHLP7:	.ASCIZ !BASE/OFFSET=NNN - PRINT SINGLE LOCATION!<15><12> ;REV B BY EC
5484	014370	047045	040445	040502	RPTIV:	.ASCIZ /%NZA?BASE OFFSET=%03XA TOO BIG !/ ;REV B BY EC
5485	014430	047045	040445	051515	SHMSG:	.ASCIZ ?%NZA?MSG: TYPE=%TXA/SIZE=%D3?
5486	014464	042532	047522	051505	SHTYP0:	.ASCIZ /ZERONES/
5487	014473	117	042516	000123	SHTYP1:	.ASCIZ /ONES/
5488	014500	040461	052114	000	SHTYP2:	.ASCIZ /1ALT/
5489	014505	060	046101	000124	SHTYP3:	.ASCIZ /0ALT/
5490	014512	041503	052111	000124	SHTYP4:	.ASCIZ /CCITT/
5491	014520	052111	050105	000	SHTYP5:	.ASCIZ /ITEP/
5492	014525	101	050114	040510	SHTYP6:	.ASCIZ /ALPHA/
5493	014533	117	051120	051440	SHTYP7:	.ASCIZ /OPR SPEC/
5494	014544	042522	042503	053111	MO0:	.ASCIZ /RECEIVE/
5495	014554	051124	047101	046523	MO1:	.ASCIZ /TRANSMIT/
5496	014565	120	051501	044523	MO2:	.ASCIZ /PASSIVE/
5497	014575	101	052103	053111	MO3:	.ASCIZ /ACTIVE/
5498	014604	047504	047127	044514	MO4:	.ASCIZ /DOWNLINELOAD/
5499	014621	124	046101	000113	MO5:	.ASCIZ /TALK/
5500	014626	044514	052123	047105	MO6:	.ASCIZ /LISTEN/
5501	014635	000			LP0:	.ASCIZ //
5502	014636	046057	047517	036520	LP00:	.ASCIZ ?/LOOP=?
5503	014645	111	052116	051105	LP1:	.ASCIZ ?INTERNAL?

5504	014656	040503	046102	000105	LP2:	.ASCIZ	?CABLE?	
5505	014664	047514	040503	046514	LP3:	.ASCIZ	?LOCALMODEM?	
5506	014677	122	046505	052117	LP4:	.ASCIZ	?REMOTEMODEM?	
5507	014713	116	117		PNST:	.ASCII	/NO/	
5508	014715	123	040524	052524	PST:	.ASCIZ	/STATUS/	
5509	014724	047516			PNCK:	.ASCII	/NO/	
5510	014726	044103	041505	000113	PCK:	.ASCIZ	/CHECK/	
5511	014734	047516			PNEC:	.ASCII	/NO/	
5512	014736	041505	047510	000	PEC:	.ASCIZ	/ECHO/	
5513								
5524	014743	116	117		PNMS:	.ASCII	/NO/	:ADDED BY EC
5525	014745	115	042117	046505	PMS:	.ASCIZ	/MODEM/	:ADDED BY EC
5526	014753	045	022516	046101	LISP:	.ASCIZ	/?N%ALIS>/	
5527	014764	046124	037113	000	OPRMM:	.ASCIZ	/TLK>/	
5528	014771	124	044510	020123	L5060:	.ASCIZ	/THIS A 50. OR 60. HZ. LSI-11:/	
5529		015030			.EVEN			

:
: FORMAT STATEMENTS USED IN PRINT CALLS
:

5537	015030	047045	040445	047504	DLLCM:	.ASCIZ	/?N%ADOWN LINE LOAD COMPLETED SUCCESSFULLY/	
5538	015102	047045	040445	040502	NOCLK:	.ASCIZ	/?N%ABAD CLOCK - PROGRAM WILL HANG ON 'TIMEOUT'!!!/	
5539	015163	115	054101	020056	TABEX:	.ASCIZ	/?MAX. CHAR. MSG COUNT EXCEEDED -/	
5540	015223	102	043125	042506	BUFEX:	.ASCIZ	/?BUFFER FULL -/	
5541	015241	045	022516	022524	MSGTRN:	.ASCIZ	/?N%T%MSG. NOT BUILT !!!/	
5542	015272	047045	040445	044103	MSGTRU:	.ASCIZ	/?N%ACHAR. COUNT EXCEEDS BUFF LIMIT - MSG TRUNCATED/	
5543	015355	045	022516	032523	SHF0:	.ASCIZ	/?N%S5%AMODE=%T%T%T%/PASS=%Z5?	
5544								
5551	015413	045	022516	032523	SHF1:	.ASCIZ	/?N%S5%S5%S5%A/%T%A/%T%A/%T%A/%T?	
5552								
5553	015453	045	032523	040445	EFM2:	.ASCIZ	/?S5%ATOTAL MISMATCHES IN MSG = %D5/	
5554	015516	047045	051445	022463	PCPM:	.ASCIZ	/?N%S3%ACALLED FROM PC=%O6/	
5555	015550	051445	022465	041501	EFM11:	.ASCIZ	/?S5%ACOMPARE COUNT=%D5%S3%ARECEIVE COUNT=%D5/	

:EVENT DESCRIPTION MESSAGES

5560	015625	124	040522	051516	EDTXQ:	.ASCIZ	/TRANSMIT MSG QUEUED/	
5561	015651	124	040522	051516	EDTXC:	.ASCIZ	/TRANSMIT MSG COMPLETED/	
5562	015700	042522	042503	053111	EDRXQ:	.ASCIZ	/RECEIVE SPACE QUEUED/	
5563	015725	122	041505	044505	EDRXC:	.ASCIZ	/RECEIVE MSG COMPLETED/	
5564	015753	104	053105	041511	EDDER:	.ASCIZ	/DEVICE ERROR/	
5565	015770	040504	040524	041440	EDDCK:	.ASCIZ	/DATA COMPARISON STARTED/	
5566	016020	042504	044526	042503	EDDVI:	.ASCIZ	/DEVICE INIT AND SETUP/	
5567	016046	040504	040524	041440	EDDLE:	.ASCIZ	/DATA COMPARISON LENGTH ERROR/	
5568	016103	104	052101	020101	EDDDE:	.ASCIZ	/DATA COMPARISON DATA ERROR/	
5569	016136	047105	020104	043117	EDEOP:	.ASCIZ	/END OF PASS/	
5570	016152	041101	047516	046522	EDMOS:	.ASCIZ	/ABNORMAL MODEM STATUS CHANGE/	:ADDED BY EC

:*****
:THESE TWO STORAGE AREAS MUST NOT BE SEPERATED !!!!

5571
5572
5573
5574

```
5575 :EVENT REPORTING MESSAGES  
5576 016207 040 041040 051501 BASM1A: .ASCIZ / BASE TABLE/  
5577  
5578 016224 051445 022463 031517 BASM3: .ASCIZ /%S3%03/  
5579 016233 045 031523 047445 BASM2: .ASCIZ /%S3%06/  
5580 016242 047045 047445 000066 BASM1: .ASCIZ /%N%06/  
5581  
5582  
5583  
5584  
5585 016250 047045 040445 044124 NULEVT: .ASCIZ ?%N%THE DCLT EVENT LOG IS EMPTY?  
5586 016310 047045 040445 037076 EVTFO: .ASCIZ ?%N%A>>>DCLT EVENT LOG ENTRY <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<?  
5587 016403 045 022516 032504 EVTf1: .ASCIZ /%N%D5%A:%22%A:%22%S3%T/  
5588 016432 047045 051445 022463 EVTf2: .ASCIZ /%N%S3%AADDR OF MSG=%06%S3%ABYTE COUNT=%D5/  
5589 016504 047045 051445 022463 EVTf3: .ASCIZ /%N%S3%T%N/  
5590 016516 051445 022463 033117 EVTf3C: .ASCIZ /%S3%06%S3%06/  
5591 016533 045 031523 047445 EVTf3D: .ASCIZ /%S3%06%S3%06%S3%T/  
5592 016555 045 022516 031523 EVTf4: .ASCIZ /%N%S3%AADDR OF MSG=%06%S3%ABYTE COUNT=%D5%S3%AND. OF CMP ERRS=%D5/  
5593 016657 045 022516 031523 EVTf4A: .ASCIZ /%N%S3%AADDR OF MSG=%06%S3%ARX BYTES=%D5%S3%ACMPARE BYTES=%D5/  
5594 016755 045 022516 031523 EVTf4B: .ASCIZ /%N%S3%APASS=%D5%S3%AERRORS=%D5%S3%ANOBUFFS=%D5/  
5595 017034 051445 022465 041101 EVTf5A: .ASCIZ /%S5%ABYTE # IN MSG.=%D5%S3%AEXPTD=%03%S3%ARECVD=%03/  
5596  
5597 017120 047045 051445 022471 EVMOCG: .ASCIZ /%N%S9%ACHANGED TO:/  
5598  
5599 : *****  
5600 ;DO NOT SEPERATE THE NEXT LIST OF MESSAGES - MODEM SIGNAL HEADER AND REPORT  
5601  
5602 017143 045 022516 034123 EVMOHD: .ASCIZ /%N%S8%AMODEM STATUS: CTS DSR DCD RTS RI SQD TM/  
5603 017223 045 022516 034523 EVMOST: .ASCII /%N%S9%S9%S5%A/  
5604 017240 130 040 040 EVMCTS: .BYTE 'X,40,40,40  
5605 017244 130 040 040 EVMDSR: .BYTE 'X,40,40,40  
5606 017250 130 040 040 EVMDCD: .BYTE 'X,40,40,40  
5607 017254 130 040 040 EVMRTS: .BYTE 'X,40,40,40  
5608 017260 130 040 040 EVMRI: .BYTE 'X,40,40,40  
5609 017264 130 040 040 EVMSQD: .BYTE 'X,40,40,40  
5610 017270 130 040 040 EVMTM: .BYTE 'X,40,40,40  
5611 017274 000 .BYTE 0  
5612 017276 .EVEN  
5613  
5614  
5615
```

5617
5618
5619
5620
5621
5622
5623
5624 017370
5625 000400
5626 020000
5627

.SBTTL BASE TABLE ADDRESS
;THIS SECTION IS USED BY A M9301-YJ BOOT ROM FOR DOING DOWN-LINE-LOAD.
;MUST BE IN THE AREA OF '017370 + 256. BYTES' + A FEW
:.....!!!! BEWARE !!!!! DO NOT ALLOW THE ABOVE ASCIIZ MESSAGES TO EXPAND INTO
:.....!!!! THIS REGION.
.EVEN
BASE: .=17370
.BLKB 256. ;BASE TABLE ADDRESS
.=20000

5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668

020000 047045 000
020003 045 031523 040445
020014 051445 022463 052101
020025 045 031523 040445
020036 051445 022463 042501
020047 045 031523 040445
020060 051445 022463 044501
020071 045 031523 040445
020102 051445 022463 041501
020113 045 031523 040445
020124 051445 022463 046501

020135 045 022516 051501
020217 104 000120
020222 052504 000
020225 104 000114
020230 050504 000
020233 104 000101
020236 052504 000120
020242 046504 000103
020246 047104 000
020251 104 053114 000
020255 104 050115 000
020261 104 042524 000
020265 104 000126
020270 055104 000
020273 125 045516 047516
020303 113 050104 000
020307 113 055104 000
020313 113 000114
020316 046504 000126

.SBTTL ASCIZ MESSAGES CONTINUED AFTER BASE TABLE REGION

:EXECUTION STATUS MESSAGES TO BE PRINTED TO KEEP OPERATOR AWAKE
CR: .ASCIZ /%N/ ;CR FOR LINES IN A ROW
STXQ: .ASCIZ /%S3%ATXQ/ ;ABOUT TO TRANSMIT
STXC: .ASCIZ /%S3%ATXC/ ;TX COMPLETED
SRXQ: .ASCIZ /%S3%ARXQ/ ;ABOUT TO RECEIVE
SDVE: .ASCIZ /%S3%AERR/ ;DEVICE ERROR
SCM: .ASCIZ /%S3%ACMP/ ;ABOUT TO DO DATA CHECKING OF RECVD VS. EXPTD
SDVI: .ASCIZ /%S3%AINI/ ;DEVICE ABOUT TO BE INITIALIZED
SCML: .ASCIZ /%S3%ACML/ ;COMPARE LENGTH ERROR
SCMD: .ASCIZ /%S3%ACMD/ ;COMPARE DATA ERROR
SEOP: .ASCIZ /%S3%AEOP/ ;END OF PASS
SMSC: .ASCIZ /%S3%AMSC/ ;MODEM STATUS CHANGE ADDED BY EC

:REV B BY EC
:NEXT ASCIZ LINES ARE USED IN SATELLITE ID MESSAGES
SECRM: .ASCIZ /%N%ASECONDARY BOOT REQ FROM %T%A DEVICE-TYPE= %D3/
DPM: .ASCIZ /DP/
DUM: .ASCIZ /DU/
DLM: .ASCIZ /DL/
DQM: .ASCIZ /DQ/
DAM: .ASCIZ /DA/
DUPM: .ASCIZ /DUP/
DMCM: .ASCIZ /DMC/
DNM: .ASCIZ /DN/
DLVM: .ASCIZ /DLV/
DMPM: .ASCIZ /DMP/
DTEM: .ASCIZ /DTE/
DVM: .ASCIZ /DV/
DZM: .ASCIZ /DZ/
UNKM: .ASCIZ /UNKNOWN/
KDPM: .ASCIZ /KDP/
KDZM: .ASCIZ /KDZ/
KLM: .ASCIZ /KL/
DMVM: .ASCIZ /DMV/
.EVEN

```

5670 ;REV B BY EC
5671 .SBTTL DMR BASE TABLE DESCRIPTION MESSAGES
5672 020322 047125 042504 044506 DMRUNKN: .ASCIZ /UNDEFINED DATA / ;LOCATION UNDEFINED BY SPEC
5673 020342 040502 042523 052040 DMR000: .ASCIZ /BASE TABLE UPDATE INDEX POINTER/
5674 020402 040502 042523 052040 DMR001: .ASCIZ /BASE TABLE UPDATE LIMIT/
5675 020432 042502 044507 047116 DMR002: .ASCIZ /BEGINNING OF BASE TABLE DATA/
5676 020467 116 045501 020123 DMR003: .ASCIZ /NAKS RCVD..BUFFER TEMP UNAVAILABLE/
5677 020532 040516 051513 051040 DMR004: .ASCIZ /NAKS RCVD..HEADER BCC ERROR/
5678 020566 040516 051513 051040 DMR005: .ASCIZ /NAKS RCVD..DATA BCC ERROR/
5679 020620 040516 051513 051440 DMR006: .ASCIZ /NAKS SENT..BUFFER TEMP UNAVAILABLE/
5680 020663 116 045501 020123 DMR007: .ASCIZ /NAKS SENT..HEADER BCC ERROR/
5681 020717 116 045501 020123 DMR010: .ASCIZ /NAKS SENT..DATA BCC ERROR/
5682 020751 122 050105 020123 DMR011: .ASCIZ /REPS SENT..CUMUL REP SENT/
5683 021003 122 050105 020123 DMR012: .ASCIZ /REPS RCVD..CUMUL REP RCVD/
5684 021035 116 045501 020123 DMR013: .ASCIZ /NAKS RCVD..REP RESPONSE/
5685 021065 116 045501 020123 DMR014: .ASCIZ /NAKS RCVD..RCV OVERRUN/
5686 021114 040516 051513 051040 DMR015: .ASCIZ /NAKS RCVD..MSG HDR FORMAT/
5687 021146 040516 051513 051040 DMR016: .ASCIZ /NAKS RCVD..MSG TOO LONG/
5688 021176 040516 051513 051440 DMR017: .ASCIZ /NAKS SENT..REP RESPONSE/
5689 021226 040516 051513 051440 DMR020: .ASCIZ /NAKS SENT..RCV OVERRUN/
5690 021255 116 045501 020123 DMR021: .ASCIZ /NAKS SENT..MSG HDR FORMAT/
5691 021307 130 044515 020124 DMR022: .ASCIZ /XMIT UNDERRUN COUNT/
5692 021333 103 046101 020114 DMR023: .ASCIZ /CALL SET UP FAILURE COUNT/
5693 021365 101 052103 020123 DMR024: .ASCIZ /ACTS FAILURE COUNT/
5694 021410 040503 051122 042511 DMR025: .ASCIZ /CARRIER DETECT LOST COUNT(WHILE RECEIVING)/
5695 021463 122 041505 044505 DMR026: .ASCIZ /RECEIVER INACTIVE COUNT/
5696 021513 123 051124 040505 DMR027: .ASCIZ /STREAMING TIME-OUT COUNT/
5697 021544 046530 054502 024124 DMR030: .ASCIZ /XMBYT(LSB) - TOTAL # BYTES XMITTED, 32 BIT COUNTER/
5698 021627 130 041115 052131 DMR031: .ASCIZ ?XMBYT(2/4) - # BYTES XMITTED?
5699 021664 046530 054502 024124 DMR032: .ASCIZ ?XMBYT(3/4) - # BYTES XMITTED?
5700 021721 130 041115 052131 DMR033: .ASCIZ ?XMBYT(MSB) - # BYTES XMITTED?
5701 021756 041522 054502 024124 DMR034: .ASCIZ /RCBYT(LSB) - TOTAL # BYTES RECEIVED, 32 BIT COUNTER/
5702 022042 041522 054502 024124 DMR035: .ASCIZ ?RCBYT(2/4) - # BYTES RECEIVED (CONT)?
5703 022107 122 041103 052131 DMR036: .ASCIZ ?RCBYT(3/4) - # BYTES RECEIVED (CONT)?
5704 022154 041522 054502 024124 DMR037: .ASCIZ /RCBYT(MSB) - # BYTES RECEIVED/
5705 022212 047111 047503 050115 DMR040: .ASCIZ /INCOMPLETE SELECTION COUNT/
5706 022245 116 020117 042522 DMR041: .ASCIZ /NO REPLY TO SELECTION COUNTER/
5707 022303 110 043511 042510 DMR042: .ASCIZ /HIGHEST MESSAGE SUCCESSFULLY RECEIVED/
5708 022351 110 043511 042510 DMR043: .ASCIZ /HIGHEST MESSAGE TRANSMITTED/
5709 022405 110 043511 042510 DMR044: .ASCIZ /HIGHEST MESSAGE ACKNOWLEDGED/
5710 022441 116 054105 020124 DMR045: .ASCIZ /NEXT MESSAGE TO TRANSMIT/
5711 022472 040514 052123 046440 DMR046: .ASCIZ /LAST MESSAGE TO COMPLETE TRANSMISSION/
5712 022540 052503 051122 047105 DMR047: .ASCIZ /CURRENT MESSAGE BEING TRANSMITTED/
5713 022602 051124 047101 046523 DMR050: .ASCIZ /TRANSMIT END OF QUEUE/
5714 022630 051124 047101 046523 DMR051: .ASCIZ /TRANSMIT BEGINNING OF QUEUE/
5715 022664 042522 042503 053111 DMR052: .ASCIZ /RECEIVE END OF QUEUE/
5716 022711 122 041505 044505 DMR053: .ASCIZ /RECEIVE BEGINNING OF QUEUE/
5717 022744 040514 042524 052123 DMR054: .ASCIZ /LATEST NAK REASON/
5718 022766 051120 043517 040522 DMR055: .ASCIZ ?PROGRAMMABLE REP/SELECT-TIMER PRESET VALUE?
5719 023041 111 052123 052122 DMR056: .ASCIZ ?ISTR/ASTRT/REP/SELECT-TIMER COMPARE LEVEL?
5720 023114 041501 044524 042526 DMR057: .ASCIZ /ACTIVE TIME COUNT/
5721 023136 044124 042522 044123 DMR060: .ASCIZ /THRESHOLD LEVEL NAKS RCVD/
5722 023170 044124 042522 044123 DMR061: .ASCIZ /THRESHOLD COUNT NAKS RCVD/
5723 023222 044124 042522 044123 DMR062: .ASCIZ /THRESHOLD LEVEL NAKS SEND EXCEPT NO BUF/
5724 023272 044124 042522 044123 DMR063: .ASCIZ /THRESHOLD COUNT NAKS SEND EXCEPT NO BUF/
5725 023342 044124 042522 044123 DMR064: .ASCIZ /THRESHOLD LEVEL - REPS SENT/
  
```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) 16-JUN-81 14:39 PAGE 13-35
DMR BASE TABLE DESCRIPTION MESSAGES

SEQ 0071

5726	023376	044124	042522	044123	DMR065:	.ASCIZ	/THRESHOLD COUNT - REPS SENT/
5727	023432	044124	042522	044123	DMR066:	.ASCIZ	/THRESHOLD LEVEL - NO BUF AVAILABLE/
5728	023475	124	051110	051505	DMR067:	.ASCIZ	/THRESHOLD COUNT - NO BUF AVAILABLE/
5729	023540	042523	020105	046504	DMR177:	.ASCIZ	/SEE DMR TECH MANUAL FOR DESCRIPTION/
5730							
5731							
5732							

5734					:REV B BY EC
5735					.SBTTL DMC BASE TABLE DATA DESCRIPTION MESSAGES
5736	023604	047101	020120	020055	DMC002: .ASCIZ /ANP - CONSTANT 0/
5737	023625	116	046124	020122	DMC003: .ASCIZ /NTRL - NAKS RCVD..NO BUFFERS/
5738	023662	044116	051104	026440	DMC004: .ASCIZ /NHDR - NAKS RCVD..MSG HEADER BAD/
5739	023723	104	052101	020122	DMC005: .ASCIZ /DATR - NAKS RCVD..DATA BAD/
5740	023756	052116	051514	026440	DMC006: .ASCIZ /NTLS - NAKS SENT..NO BUFFERS/
5741	024013	116	042110	020123	DMC007: .ASCIZ /NHDS - NAKS SENT..BAD HEADER/
5742	024050	040504	051524	026440	DMC010: .ASCIZ /DATS - NAKS SENT..BAD DATA/
5743	024103	122	050105	051503	DMC011: .ASCIZ /REPCS - REPS SENT/
5744	024125	122	050105	051103	DMC012: .ASCIZ /REPCR - REPS RECD/
5745	024147	102	051501	020105	DMC013: .ASCIZ /BASE - CORE TABLE BASE ADDRESS/
5746	024206	042523	020105	046504	DMC377: .ASCIZ /SEE DMC TECH MANUAL FOR DESCRIPTION/

5755
 5756
 5757
 5758
 5759
 5760
 5761
 5762
 5763
 5764
 5765
 5766
 5767
 5768
 5769
 5770
 5771
 5772
 5773
 5774
 5775
 5776
 5777
 5778
 5779
 5780
 5781
 5782
 5783
 5784
 5785
 5786
 5787
 5788
 5789
 5790
 5791
 5792
 5793
 5794
 5795
 5796
 5797
 5808
 5809
 5817
 5818
 5819
 5820

:DEVICE ERROR MESSAGES

024252	044524	042515	047440	DVEM0:	.ASCII	/TIME OUT WAITING FOR RDI TO CLEAR/
024313	015	020012	020040		.ASCIZ	<15><12>/ SEL0 SEL2 /
024340	044524	042515	047440	DVEM1:	.ASCII	/TIME OUT WAITING FOR RDI TO SET/
024377	015	020012	020040		.ASCIZ	<15><12>/ SEL0 SEL2 /
024424	044524	042515	047440	DVEM3:	.ASCII	/TIME OUT WAITING FOR RUN TO SET/
024463	015	020012	020040		.ASCIZ	<15><12>/ SEL0 SEL2 /
024510	044524	042515	047440	DVEM4:	.ASCII	/TIME OUT WAITING FOR OUTPUT INTERRUPT/
024555	015	020012	020040		.ASCIZ	<15><12>/ SEL0 SEL2 /
024602	047111	052520	020124	DVEM5:	.ASCII	/INPUT INTERRUPT WHEN EXPECTING OUTPUT/
024647	015	020012	020040		.ASCIZ	<15><12>/ SEL0 SEL2 /
024674	046111	042514	040507	DVEM6:	.ASCII	/ILLEGAL OUTPUT INTERRUPT/
024724	005015	020040	051440		.ASCIZ	<15><12>/ SEL2 SEL6 /
024751	103	047117	051124	DVEM7:	.ASCII	/CONTROL OUT INSTEAD OF BA-CC OUT/
025011	015	020012	020040		.ASCIZ	<15><12>/ SEL2 SEL6 /
025036	054124	041040	043125	DVEM8:	.ASCII	/TX BUFF COMPLETED AND SHOULD BE RX/
025100	005015	020040	051440		.ASCIZ	<15><12>/ SEL4 SEL6 /
025125	122	020130	052502	DVEM9:	.ASCII	/RX BUFF COMPLETED AND SHOULD BE TX/
025167	015	020012	020040		.ASCIZ	<15><12>/ SEL4 SEL6 /
025214	042040	053517	020116	DLLAB:	.ASCII	/ DOWN LINE LOAD ABORTED/
025243	015	020012	020040		.ASCIZ	<15><12>/ RXBUF TXBUF /
025270	051120	041517	042105	PROEM:	.ASCIZ	/PROCEDURE ERROR/
025310	047516	020116	054105	NXMM:	.ASCIZ	/NON EXIST MEM/
025326	042104	046503	020120	DDCSR:	.ASCIZ	/DDCMP START REC/
025346	044504	041523	047117	DISCOM:	.ASCIZ	/DISCONNECT/
025361	114	051517	020124	LOSDAM:	.ASCIZ	/LOST DATA/
025373	104	041504	050115	DDCMRM:	.ASCIZ	/DDCMP MAINT REC/
025413	124	046511	020105	TIMOM:	.ASCIZ	/TIME OUT/
025424	040504	040524	041440	DATCKM:	.ASCIZ	/DATA CHECK/
025437	122	047125	051440	RUNSBM:	.ASCIZ	/RUN SET ILLEGALLY/
025461	122	020130	042111	RXIDM:	.ASCIZ	/RX IDLE/
025471	103	020104	046107	CDGLM:	.ASCIZ	/CD GLITCHED/
025505	103	051524	043040	CTSFM:	.ASCIZ	/CTS FAILED/
025521	124	020130	047516	TXNC:	.ASCIZ	/TX NOT COMPLETE/
025541	122	020130	047516	RXNC:	.ASCIZ	/RX NOT COMPLETE/
025561	123	041505	051040	RXM1:	.ASCIZ	/SEC REQ ERR WORD 1/
025604	042523	020103	042522	RXM2:	.ASCIZ	/SEC REQ ERR WORD 2/
025630					.EVEN	

5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5848

.SBTTL GLOBAL ERROR REPORT SECTION

:++
: THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
: USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION. PRINTB
: (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
:--

5849 025630
(3) 025630
5850 025630
(10) 025630 005046
(10) 025632 153716 007447
(9) 025636 005046
(9) 025640 153716 007446
(8) 025644 013746 007424
(7) 025650 012746 017034
(6) 025654 012746 000004
(3) 025660 010600
(4) 025662 104414
(4) 025664 062706 000012
5851 025670
(3) 025670
(3) 025670 104423

BGNMSG ERR1

PRINTB #EVTF5A,OFSET,<B,GOOD>,<B,BAD>

ERR1::

:INDIVIDUAL DATA COMPARE ERROR
CLR -(SP)
BISB BAD,(SP)
CLR -(SP)
BISB GOOD,(SP)
MOV OFSET, -(SP)
MOV #EVTF5A, -(SP)
MOV #4, -(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #12,SP

ENDMSG

L10001:

TRAP C\$MSG

5852
5853 025672
(3) 025672
5854 025672
(8) 025672 013746 007436
(7) 025676 012746 015453
(6) 025702 012746 000002
(3) 025706 010600
(4) 025710 104414
(4) 025712 062706 000006
5855 025716
(3) 025716
(3) 025716 104423

BGNMSG ERR2

PRINTB #EFM2,TEMP4

ERR2::

:TOTAL DATA COMPARE FAILS ERROR
MOV TEMP4, -(SP)
MOV #EFM2, -(SP)
MOV #2, -(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

ENDMSG

L10002:

TRAP C\$MSG

5856
5857 025720
(3) 025720
5858 025720
(9) 025720 013746 007434
(8) 025724 010446
(7) 025726 012746 015550
(6) 025732 012746 000003
(3) 025736 010600
(4) 025740 104414
(4) 025742 062706 000010
5859 025746
(3) 025746
(3) 025746 104423

BGNMSG ERR10

PRINTB #EFM11,R4,TEMP3

ERR10::

MOV TEMP3, -(SP)
MOV R4, -(SP)
MOV #EFM11, -(SP)
MOV #3, -(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

ENDMSG

L10003:

TRAP C\$MSG

5860

5870 025750
(3) 025750
5871 025750
(10) 025750 013746 007442
(9) 025754 013746 007436
(8) 025760 013746 007434
(7) 025764 012746 016533
(6) 025770 012746 000004
(3) 025774 010600
(4) 025776 104414
(4) 026000 062706 000012
5872 026004
(8) 026004 013746 007462
(7) 026010 012746 015516
(6) 026014 012746 000002
(3) 026020 010600
(4) 026022 104414
(4) 026024 062706 000006
5873 026030
(3) 026030
(3) 026030 104423
5874
5875 026032
(3) 026032
5876 026032
(9) 026032 013746 007436
(8) 026036 013746 007434
(7) 026042 012746 016516
(6) 026046 012746 000003
(3) 026052 010600
(4) 026054 104414
(4) 026056 062706 000010
5877 026062
(8) 026062 013746 007462
(7) 026066 012746 015516
(6) 026072 012746 000002
(3) 026076 010600
(4) 026100 104414
(4) 026102 062706 000006
5878 026106
(3) 026106
(3) 026106 104423
5879
5880 026110
(3) 026110
5881 026110
(9) 026110 013746 007436
(8) 026114 013746 007434
(7) 026120 012746 016516
(6) 026124 012746 000003
(3) 026130 010600
(4) 026132 104414
(4) 026134 062706 000010
5882 026140
(3) 026140
(3) 026140 104423

BGNMSG ERR8
PRINTB #EVTF3D,TEMP3,TEMP4,CONOTM

PRINTB #PCPM,PCADD

ENDMSG

BGNMSG ERR9
PRINTB #EVTF3C,TEMP3,TEMP4

PRINTB #PCPM,PCADD

ENDMSG

BGNMSG ERR13
PRINTB #EVTF3C,TEMP3,TEMP4

ENDMSG

ERR8::

MOV CONOTM,-(SP)
MOV TEMP4,-(SP)
MOV TEMP3,-(SP)
MOV #EVTF3D,-(SP)
MOV #4,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #12,SP

L10004:

MOV PCADD,-(SP)
MOV #PCPM,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

TRAP C\$MSG

ERR9::

MOV TEMP4,-(SP)
MOV TEMP3,-(SP)
MOV #EVTF3C,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

L10005:

MOV PCADD,-(SP)
MOV #PCPM,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

TRAP C\$MSG

ERR13::

MOV TEMP4,-(SP)
MOV TEMP3,-(SP)
MOV #EVTF3C,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

L10006:

TRAP C\$MSG

5883
5884 026142
(3) 026142
5885 026142
(10) 026142 013746 007442
(9) 026146 013746 007436
(8) 026152 013746 007434
(7) 026156 012746 016533
(6) 026162 012746 000004
(3) 026166 010600
(4) 026170 104414
(4) 026172 062706 000012
5886 026176
(3) 026176
(3) 026176 104423
5887
5888 026200
(4) 026200 000167
(3) 026202 177772
5889
5890

BGNMSG ERR14

PRINTB #EVTF3D,TEMP3,TEMP4,CONOTM

ENDMSG

EXIT MSG

ERR14::

MOV CONOTM,-(SP)
MOV TEMP4,-(SP)
MOV TEMP3,-(SP)
MOV #EVTF3D,-(SP)
MOV #4,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #12,SP

L10007:

TRAP C\$MSG

.WORD JSJMP
.WORD L10007-2-

5892
5893
5894
5895
5896
5897
5898
5975
5976
5977
5978
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

.SBTTL GLOBAL SUBROUTINES SECTION

;++
: THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
: THAT ARE USED IN MORE THAN ONE TEST.
:--

.SBTTL 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, THIS 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

IMPLICIT INPUTS:
: THE SUPERVISOR SPACE WHERE CLOCK INFO WAS RETURNED BY THE "CLOCK" CALL

OUTPUTS:
: "CLKCSR" GETS LOADED WITH THE CLOCK'S 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. (HERTZ RATE) WHICH DETERMINES
: THE NUMBER OF TICKS IN A SECOND

CALLING SEQUENCE:
: JSR PC,CLKSET ;CALL CLOCK SETUP WITH R1 & R2 SETUP
:--

CLKSET:
: MOV (R1)+,(R2)+ ;LOAD CLOCK'S CSR ADDR. INTO "CLKCSR"
: MOV (R1)+,(R2) ;LOAD CLOCK'S INT. LEVEL INTO "CLKBR"
: ASL (R2) ;ADJUST THE INT. LEVEL FOR LOADING INTO
: ; THE PSW WITH A "SETVEC" CALL
: ASL (R2)
: ASL (R2)
: ASL (R2)+
: MOV (R1)+,(R2)+ ;LOAD CLOCK'S INT. VECTOR INTO "CLKVEC"
: MOV (R1)+,(R2)+ ;LOAD CLOCK'S HERTZ RATE INTO "CLKHZ"
: RTS PC

6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
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

```

.SBTTL      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 & SECONDS
TIMER 1,2, & S: CURRENT VALUES OF THE 'EVENT TIMERS'

IMPLICIT OUTPUTS:
NEW VALUE OF EVENT TIMER '1' DECREMENTED BY 1 TICK IF IT WAS NON-ZERO
NEW VALUE OF EVENT TIMER '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

FUNCTIONAL SIDE EFFECTS:
THE CLOCK IS DISABLED UPON ENTRY AND REENABLED WHEN LEAVING

CALLING SEQUENCE:
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 SUPERVISOR 'SETVEC' CALL.
  
```

```

026230      BGNSRV CLKINT
(3) 026230      CLKINT::

026230 005077 161266      CLR      @CLKCSR      ;DISABLE THE CLOCK FORM INTERRUPTING
026234 005337 007540      BEQ      TIMTCK      ;DECREMENT THE # OF TICKS/SEC.
026240 001015              BNE      1$          ;GO CHECK TIMERS (1&2-TICKS, 3-SECONDS)
026242 013737 007530 007540  MOV     CLKHZ,TIMTCK ;RESET THE # OF TICKS/SEC.
026250 005237 007536      INC     TIMSEC      ;INC # OF SECS-SINCE-START
026254 022737 000074 007536  CMP     #60.,TIMSEC ;SEE IF WE'VE COUNTED 60 SECS. YET
026262 001004              BNE      1$          ;IF NOT, GO CHECK TIMERS
026264 005237 007534      INC     TIMMIN      ; ELSE INC MINUTES-SINCE-START
026270 005037 007536      CLR     TIMSEC      ; AND RESTART SECOND COUNTER

026274 005737 007542      1$:    TST     TIMER1      ;SEE IF TIMER #1, TIMING ANYTHING
026300 001402              BEQ     2$          ; IF=0, NOTHING BEING TIMED CHECK NEXT TIMER
026302 005337 007542      DEC     TIMER1      ; ELSE DECREMENT THE TIMER VALUE (BY 1 TICK)
026306 005737 007544      2$:    TST     TIMER2      ;SEE IF TIMER #2, TIMING ANYTHING
026312 001402              BEQ     3$          ; IF=0, NOTHING BEING TIMED CHECK NEXT TIMER
026314 005337 007544      DEC     TIMER2      ; ELSE DECREMENT THE TIMER VALUE (BY 1 TICK)
026320 005737 007546      3$:    TST     TIMERS      ;SEE IF TIMER #3, TIMING ANYTHING
026324 001406              BEQ     4$          ; IF=0, NOTHING BEING TIMED, LEAVE
026326 023737 007530 007540  CMP     CLKHZ,TIMTCK ;SEE IF A SECOND HAS BEEN COUNTED OFF
026334 001002              BNE     4$          ; BR IF NO
026336 005337 007546      DEC     TIMFRS      ; ELSE DECREMENT THE TIMER VALUE (BY 1 SEC.)
  
```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) ^{B 7} 16-JUN-81 14:39 PAGE 13-43
CLOCK INTERRUPT SERVICE ROUTINE

SEQ 0079

6071 026342 013777 007532 161152 4\$:
6072 026350
(3) 026350
(2) 026350 000002

MOV CLKEN,@CLKCSR ;REENABLE THE CLOCK TO INTERRUPT
ENDSRV

L10010:
RTI

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
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129

```
.SBTTL          EVENT LOG SUBROUTINES

:++
:FUNCTIONAL DESCRIPTION:
:THIS SUBROUTINE HAS A DIFFERENT ENTRY POINT
:FOR EACH EVENT TO BE LOGGED AND ALWAYS PRINTS
:THE SHORT 'OPERATOR AWAKE' MESSAGE TO CONSOLE THEN LOGS THE
:EVENT TYPE, TIME, AND THE OTHER 3 WORDS OF INFO PASSED TO THE
:SUBROUTINE AT CALLING TIME

:INPUTS:
:TIMMIN & TIMSEC:  CURRENT VALUE OF 'TIME-SINCE-START'
:TEMP2: WORD #1 OF EVENT LOG INFORMATION (FOR MOST EVENT TYPES)
:TEMP3: WORD #2 OF EVENT LOG INFORMATION
:TEMP4: WORD #3 OF EVENT LOG INFORMATION
:MODS:  CURRENT VALUE OF THE MODEM SIGNALS AVAILABLE FROM THE DEVICE

:OUTPUTS:
:'OPERATOR AWAKE' MESSAGE SENT TO THE CONSOLE
:NEW EVENT LOGGED IN 'EVTLOG' (EVENT LOG)
:UPDATED 'EVTPTN' (EVENT LOG ENTRY POINTER)

:SUBORDINATE ROUTINES USED:
:'DVMODS' THE DEVICE SUBROUTINE THAT RETURNS MODEM STATUS IN 'MODS'
:(FOR SOME EVENT TYPES)

:FUNCTIONAL SIDE EFFECTS:
:TEMP:  USED TO STORE ADDRESS OF 'OPERATOR AWAKE' MESSAGE
:TEMP1: USED TO SETUP THE VALUE OF THE 'EVENT TYPE' BYTE FOR LOGGING

:CALLING SEQUENCE:
:JSR    PC,LOGTXQ      ;CALL THE LOG EVENT SUBROUTINE WITH TEMP,TEMP1,
:          ..          ; TEMP2, TEMP3, AND TEMP4 SETUP
:          ..          ;
:JSR    PC,LOGCMP

:--

LOGTXQ:  MOV    #STXQ,TEMP1  ;SET UP MSG. TO PRINT
:        MOV    #TXQ,TEMP   ;SET UP EVENT TYPE
:        BR     LOGS1       ;GO LOG EVENT AND TIME

LOGTXC:  MOV    #STXC,TEMP1  ;SET UP MSG. TO PRINT
:        MOV    #TXC,TEMP   ;SET UP EVENT TYPE
:        BR     LOGS1       ;GO LOG EVENT AND TIME

LOGRXQ:  MOV    #SRXQ,TEMP1  ;SET UP MSG. TO PRINT
:        MOV    #RXQ,TEMP   ;SET UP EVENT TYPE
:        BR     LOGS1       ;GO LOG EVENT AND TIME

LOGRXC:  MOV    #RXC,TEMP    ;SET UP EVENT TYPE
:        BR     LOGS1       ;GO LOG EVENT AND TIME

LGDVE:

```

```
026352
026352 012737 020003 007430
026360 012737 000000 007426
026366 000510
026370
026370 012737 020014 007430
026376 012737 000002 007426
026404 000501
026406
026406 012737 020025 007430
026414 012737 000004 007426
026422 000472
026424
026424 012737 000006 007426
026432 000466
026434
```

```

6130 026434 012737 020036 007430      MOV      #SDVE,TEMP1      ;SET UP MSG. TO PRINT
6131 026442 012737 000010 007426      MOV      #DER,TEMP      ;SET UP EVENT TYPE
6132 026450 000474                      BR        LOGS3          ;GO LOG EVENT AND TIME
6133
6134 026452                      LOGDVI:
6135 026452 012737 020060 007430      MOV      #SDVI,TEMP1     ;SET UP MSG. TO PRINT
6136 026460 012737 000012 007426      MOV      #DVI,TEMP      ;SET UP EVENT TYPE
6137 026466 113737 007470 007432      MOV      MODTYP,TEMP2
6138 026474 113737 007472 007433      MOV      MLTYP,TEMP2+1
6139 026502 013737 007500 007434      MOV      RPASS,TEMP3
6140 026510 013737 007476 007436      MOV      PARAM,TEMP4    ;SET UP EVNT ENTRIES
6141 026516 000451                      BR        LOGS3          ;GO LOG EVENT AND TIME
6142
6143 026520                      LOGCMP:
6144 026520 012737 020047 007430      MOV      #SCM,TEMP1     ;SET UP MSG. TO PRINT
6145 026526 012737 000014 007426      MOV      #DCK,TEMP      ;SET UP EVENT TYPE
6146 026534 000442                      BR        LOGS3
6147 026536                      LOGCML:
6148 026536 012737 020071 007430      MOV      #SCML,TEMP1    ;SET UP MSG. AND TYPE
6149 026544 012737 000020 007426      MOV      #DLE,TEMP      ;GO LOG EVENT AND TIME
6150 026552 000433                      BR        LOGS3
6151 026554                      LOGCMD:
6152 026554 012737 020102 007430      MOV      #SCMD,TEMP1    ;SET UP MSG. AND TYPE
6153 026562 012737 000022 007426      MOV      #DDE,TEMP      ;GO LOG EVENT AND TIME
6154 026570 000424                      BR        LOGS3
6155 026572                      LOGEOP:
6156 026572 012737 020113 007430      MOV      #SEOP,TEMP1    ;SET UP MSG. AND TYPE
6157 026600 012737 000024 007426      MOV      #EOP,TEMP      ;GO LOG EVENT AND TIME
6158 026606 000415                      BR        LOGS3
6159
6160 026610 013746 007400                      LOGS1: MOV      ERRCNT,-(SP)  ;SAVE CURRENT ERROR COUNT
6161 026614 004737 044004                      JSR      PC,DVMODS      ;GO GET MODEM STATUS
6162 026620 012604                      MOV      (SP)+,R4      ;GET SAVED ERRCNT VALUE
6163 026622 020437 007400                      CMP      R4,ERRCNT     ;WHERE ANY ERRORS FOUND
6164 026626 001402                      BEQ      1$           ;BR IF NONE
6165 026630 000137 027044                      JMP      LOGEX        ;ELSE, LEAVE WITHOUT LOGGING ANYTHING
6166                                     ; BUT THE DEVICE ERROR FROM 'DVMODS'
6167 026634 013737 010456 007436 1$:      MOV      MODS,TEMP4    ;AND PUT IT IN TEMP4
6168
6169 026642                      LOGS3:
6170 026642 022737 000006 007426      CMP      #RXC,TEMP      ;IF RXC DONT PRINT
6171 026650 001434                      BEQ      LOGS5
6172 026652 032737 000001 007476      BIT      #STATB,PARAM  ;IF NO STATUS SELECTED
6173 026660 001430                      BEQ      LOGS5        ;GO TO 5
6174
6175
6176 026662 022737 000010 007372      CMP      #10,LNCNT     ;HAVE WE DONE 10?
6177 026670 001012                      BNE      LOGS4        ;IF NOT GO TO 4
6178 026672 005037 007372      CLR      LNCNT        ;ESLE CLEAR IT
6179
6180 026676                      PRINTF #CR           ;ELSE PRINT CR
(7) 026676 012746 020000                      MOV      #CR,-(SP)
(6) 026702 012746 000001                      MOV      #1,-(SP)
(3) 026706 010600                      MOV      SP,R0
(4) 026710 104417                      TRAP    C$PNTF
(4) 026712 062706 000004                      ADD     #4,SP

```

```
6181 026716
6182 026716 005237 007372
6183 026722
(7) 026722 013746 007430
(6) 026726 012746 000001
(3) 026732 010600
(4) 026734 104417
(4) 026736 062706 000004
6184 026742 010346
6185 026744 013703 007550
6186 026750 113723 007426
6187 026754 013737 007530 007426
6188 026762 163737 007540 007426
6189 026770 113723 007426
6190 026774 113723 007536
6191 027000 113723 007534
6192 027004 013723 007432
6193 027010 013723 007434
6194 027014 013723 007436
6195 027020 020327 010454
6196 027024 103404
6197
6198 027026 012713 177777
6199 027032 012703 007552
6200 027036 010337 007550
6201 027042 012603
6202 027044 000207
6203
6204
```

LOGS4: INC LNCNT ;INC COUNTER OF # OF AWAKE MSGS
PRINTF TEMP1 ;PRINT OPERATOR AWAKE MSG.

LOGS5: MOV R3,-(SP) ;SAVE R3 ON THE STACK
MOV EVTPTR,R3
MOVB TEMP,(R3)+ ;LOG EVENT
MOV CLKHZ,TEMP
SUB TIMTCK,TEMP
MOVB TEMP,(R3)+ ;LOG TIME SINCE START
MOVB TIMSEC,(R3)+
MOVB TIMMIN,(R3)+ ;TICKS,SECS AND MINS.
MOV TEMP2,(R3)+ ;LOG EVNT ENTRY 3
MOV TEMP3,(R3)+ ;LOG EVNT ENTRY 4
MOV TEMP4,(R3)+ ;LOG EVNT ENTRY 5
CMP R3,#EVTEND
BLO LOGS2 ;IF EVENT LOG FULL GO
;CONTINUE;ELSE GO TO 2
MOV #-1,(R3) ;LOG A TABLE END
MOV #EVTLOG,R3 ;PUT R3 TO START OF TABLE
LOGS2: MOV R3,EVTPTR ;RESTORE POINTER
MOV (SP)+,R3 ;RESTORE R3
LOGEX: RTS PC

MOV TEMP1,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTF
ADD #4,SP

```
6206 .SBTTL REPORT BASE TABLE OR EVENT LOG
6207 ;REV B BY EC
6208 ;THE FOLLOWING COMMANDS ADDED TO REVISION B CZCLK
6209 ;:DMR/DMC DCLT PROGRAM
6210 ;:RPT> LOG
6211 ;: BASE/ERROR
6212 ;: BASE/FULL
6213 ;: BASE/OFFSET=NN
6214 ;: HELP
6215 ;: EXIT
6216
6217 REPORT: MOV R2,-(SP) ;SAVE R2,R3,R4 ON THE STACK
6218 MOV R3,-(SP)
6219 MOV R4,-(SP)
6220
6221 ;PRINT HELP MESSAGE
6222 PRINTF #RHLPO ;BASIC HELP MESSAGE
(7) 027054 012746 014020 MOV #RHLPO,-(SP)
(6) 027060 012746 000001 MOV #1,-(SP)
(3) 027064 010600 MOV SP,R0
(4) 027066 104417 TRAP C$PNTF
(4) 027070 062706 000004 ADD #4,SP
6223
6224 GETRCL: CLRB P$GDBD ;INIT GOOD/BAD FLAG -1=BAD INPUT
6225 CLRB P$NNUF ;INIT MORE COMMAND LINE INPUT NEEDED
6226
6227 ;PRINT PROMPT 'RPT>'
6228 GMANID CLISRP,CMDBUF,A,0,1,72.,NO
(3) 027104 104443 TRAP C$GMAN
(3) 027106 000406 BR 10000$
(4) 027110 003060 .WORD CMDBUF
(5) 027112 000142 .WORD T$CODE
(5) 027114 012504 .WORD CLISRP
(5) 027116 000000 .WORD 0
(5) 027120 000001 .WORD T$LOLIM
(5) 027122 000110 .WORD T$HILIM
(3) 027124
6229 027124 012737 003060 003544 MOV #CMDBUF,P$BUFA ;INPUT BUFFER
6230 027132 012737 027266 003546 MOV #CLIRT,P$TREE ;REPORT CLI TREE
6231 027140 012737 027464 003550 MOV #CLIRAC,P$ACT ;ACTION ROUTINES
6232 027146 005037 003204 CLR QUALFG
6233 027152 004737 032642 JSR PC,P$TRV ;GO PARSE COMMAND LINE
6234 027156 105737 003561 TSTB P$GDBD ;COMMAND OK ?
6235 027162 001412 BEQ 1$ ;YES,BRANCH
6236 027164 PRINTF #CLIERM ;PRINT INVALID INPUT MESSAGE
(7) 027164 012746 012512 MOV #CLIERM,-(SP)
(6) 027170 012746 000001 MOV #1,-(SP)
(3) 027174 010600 MOV SP,R0
(4) 027176 104417 TRAP C$PNTF
(4) 027200 062706 000004 ADD #4,SP
6237 027204 000137 027074 JMP GETRCL ;TRY AGAIN
6238
6239 1$: TSTB P$NNUF ;MORE COMMAND NEEDED ?
6240 027210 105737 003560 BEQ 10$ ;NO,BRANCH
6241 027214 001412 PRINTF #CLINUF ;INCOMPLETE MESSAGE
(7) 027216 012746 012542 MOV #CLINUF,-(SP)
```

(6)	027222	012746	000001						MOV	#1,-(SP)
(3)	027226	010600							MOV	SP,R0
(4)	027230	104417							TRAP	C\$PNTF
(4)	027232	062706	000004						ADD	#4,SP
6242	027236	000137	027074			JMP	GETRCL			:TRY AGAIN
6243										
6244	027242	023727	003202	000002	10\$:	CMP	KEYWD1,#RPEXT			:EXIT COMMAND ?
6245	027250	001402				BEQ	20\$:YES,BRANCH
6246	027252	000137	027074			JMP	GETRCL			:GET ANOTHER COMMAND
6247	027256	012604			20\$:	MOV	(SP)+,R4			:RESTORE R4
6248	027260	012603				MOV	(SP)+,R3			:RESTORE R3
6249	027262	012602				MOV	(SP)+,R2			:RESTORE R2
6250	027264	000207				RTS	PC			:RETURN

```
6252 .SBTTL COMMAND LINE PARSING TREE FOR REPORT
6253 027266 CLIRT: CLI CLISPA,0,R10$ ;SKIP SPACES IN COMMAND LINE
6254 027272 R10$: CLI <'?'>,RPHLP,R11$ ;IF INPUT = ? THEN PRINT HELP MESSAGE
6255 027276 CLI CLIEXI,0 ;AND EXIT PARSER
6256 027300 R11$: CLI CLISTR,RPHLP,R12$,<'HELP'> ;IF INPUT = 'HELP' THEN PRINT HELP
6257 027314 CLI CLIEXI,0 ;MESSAGE AND EXIT PARSER
6258 027316 R12$: CLI CLISTR,RPEXT,R13$,<'EXIT'> ;IF INPUT = 'EXIT' THEN SET KEYWORD =
6259 027332 CLI CLIEXI,0 ;RPEXT AND EXIT PARSER
6260 027334 R13$: CLI CLISTR,RPLOG,R14$,<'LOG'> ;IF INPUT = 'LOG' THEN GO PRINT EVENT
6261 027346 CLI CLIEXI,0 ;LOG AND EXIT PARSER
6262 027350 R14$: CLI CLISTR,RNOTNF,R30$,<'BASE'>;IF INPUT = 'BASE' THEN MORE COMMAND
6263 027364 CLI CLIBR,0,R15$ ;LINE IS NEEDED
6264 027370 R15$: CLI <'/'>,RNOTNF,R125$ ;IF INPUT = '/' THEN LOOK FOR MORE
6265 027374 CLI CLISTR,RPSWE,R16$,<'ERROR'> ;IF INPUT = 'ERROR' THEN GO PRINT
6266 027410 CLI CLIEXI,0 ;ERROR INFORMATION
6267 027412 R16$: CLI CLISTR,RPSWF,R17$,<'FULL'> ;IF INPUT = 'FULL' THEN GO PRINT
6268 027426 CLI CLIEXI,0 ;ENTIRE BASE TABLE
6269 027430 R17$: CLI CLISTR,RNOTNF,R30$,<'OFFSET'>;IF INPUT = 'OFFSET' THEN LOOK FOR
6270 027446 CLI <'='>,0,R30$ ;'='
6271 027452 CLI CLIOCT,RPSWO,R30$ ;IF INPUT = OCTAL VALUE THEN GO
6272 027456 CLI CLIEXI,0 ;PRINT SINGLE BASE TABLE ITEM
6273 027460 R30$: CLI CLIERR,0
6274 027462 R125$: CLI CLIEXI,0
```

```

6276 .SBTTL CLI ACTION DISPATCHER AND ROUTINES
6277 027464 006302 CLIRAC: ASL R2 ;SET UP INDEX
6278 027466 016202 027502 MOV 10$(R2),R2 ;
6279 027472 062702 027502 ADD #10$,R2 ;
6280 027476 004712 JSR PC,(R2) ;GO DO ACTION
6281 027500 000207 RTS PC ;RETURN
6282 027502 000026 10$: .WORD ACTRNL-10$ ;NULL
6283 027504 000030 .WORD ACTRHL-10$ ;HELP ROUTINE
6284 027506 000074 .WORD ACTREX-10$ ;EXIT ROUTINE
6285 027510 000104 .WORD ACTRLG-10$ ;REPORT EVENT LOG ROUTINE
6286 027512 000120 .WORD ACTSWE-10$ ;REPORT ERRORS ROUTINE
6287 027514 000262 .WORD ACTSWF-10$ ;REPORT ENTIRE BASE TABLE
6288 027516 000374 .WORD ACTSWO-10$ ;REPORT SINGLE BASE ADDRESS
6289 027520 000020 .WORD ACTRNF-10$ ;MORE COMMAND NEEDED
6290
6291 ;MORE COMMAND NEEDED
6292 027522 112737 177777 003560 ACTRNF: MOVB #-1,P$NNUF ;MORE COMMAND NEEDED
6293 027530 000207 ACTRNL: RTS PC ;NULL
6294
6295 ;PRINT HELP MESSAGE
6296 027532 012702 003230 ACTRHL: MOV #RHLPTB,R2 ;INDEX FOR HELP MESSAGES
6297 027536 1$: PRINTF #HLPF,(R2)+ ;PRINT IT
(8) 027536 012246
(7) 027540 012746 013225
(6) 027544 012746 000002
(3) 027550 010600
(4) 027552 104417
(4) 027554 062706 000006
6298 027560 020227 003246 CMP R2,#RHLPEN ;LAST MESSAGE ?
6299 027564 001364 BNE 1$ ;NO,BRANCH
6300 027566 012737 000001 003202 MOV #RPHLP,KEYWD1 ;SET KEYWORD
6301 027574 000207 RTS PC ;RETURN
6302
6303 ;EXIT REPORT LEVEL
6304 027576 012737 000002 003202 ACTREX: MOV #RPEXT,KEYWD1 ;SET KEYWORD AND RETURN
6305 027604 000207 RTS PC
6306
6307 ;PRINT ERROR LOG
6308 027606 004737 030574 ACTRLG: JSR PC,REPLOG ;GO PRINT EVENT LOG
6309 027612 012737 000003 003202 MOV #RPLOG,KEYWD1 ;SET KEYWORD
6310 027620 000207 RTS PC ;RETURN
6311
6312 ;PRINT ONLY ERROR LOCATIONS
6313 027622 005737 012406 ACTSWE: TST OPTYP ;DMR ?
6314 027626 001026 BNE 10$ ;YES,BRANCH
6315 027630 012737 003432 007450 MOV #DMCIND,INDEX ;SETUP DMC MESSAGES
6316 027636 062737 000006 007450 ADD #6,INDEX ;POINT TO CORRECT MESSAGE
6317 027644 012737 003462 007452 MOV #DMCEND,INDEXE ;LAST DMC ADDRESS
6318 027652 012737 017370 007454 MOV #BASE,BEND ;SET UP LAST ADDRESS
6319 027660 062737 000012 007454 ADD #12,BEND ;;TO BE PRINTED
6320 027666 012737 017370 007456 MOV #BASE,BDATA ;BASE TABLE START ADDRESS
6321 027674 062737 000003 007456 ADD #3,BDATA ;ERROR START ADDRESS
6322 027702 000425 BR 20$ ;
6323 027704 012737 003250 007450 10$: MOV #DMRIND,INDEX ;SETUP FOR DMR MESSAGES
6324 027712 062737 000006 007450 ADD #6,INDEX ;POINT TO FIRST ERROR MESSAGE
6325 027720 012737 003430 007452 MOV #DMREND,INDEXE ;LAST DMR MESSAGE

```

```

MOV (R2)+,-(SP)
MOV #HLPF,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

```

6326	027726	012737	017370	007454		MOV	#BASE,BEND	:SETUP LAST ADDRESS	
6327	027734	062737	000041	007454		ADD	#41,BEND	:TO BE PRINTED	
6328	027742	012737	017370	007456		MOV	#BASE,BDATA	:START ADDRESS BASE TABLE	
6329	027750	062737	000003	007456		ADD	#3,BDATA	:START ADDRESS ERRORS	
6330	027756	004737	030336		20\$:	JSR	PC,RPBASE	:GO PRINT DATA	
6331	027762	000207				RTS	PC	:RETURN	
6332									
6333									
6334	027764	005737	012406						
6335	027770	001020			ACTSWF:	TST	OPTYP	:DMR ?	
6336	027772	012737	003432	007450		BNE	10\$:YES,BRANCH	
6337	030000	012737	003462	007452		MOV	#DMCIND,INDEX	:SETUP DMC MESSAGES	
6338	030006	012737	017370	007454		MOV	#DMCEND,INDEXE	:LAST MESSAGE	
6339	030014	062737	000377	007454		MOV	#BASE,BEND	:TABLE START ADDRESS	
6340	030022	012737	017370	007456		ADD	#377,BEND	:PRINT 256. BYTES OF DATA	
6341	030030	000417				MOV	#BASE,BDATA	:FIRST ADDRESS TO PRINT	
6342	030032	012737	003250	007450	10\$:	BR	20\$		
6343	030040	012737	003430	007452		MOV	#DMRIND,INDEX	:SETUP DMR MESSAGES	
6344	030046	012737	017370	007454		MOV	#DMREND,INDEXE	:LAST DMR MESSAGE	
6345	030054	062737	000177	007454		MOV	#BASE,BEND	:TABLE START ADDRESS	
6346	030062	012737	017370	007456		ADD	#177,BEND	:PRINT 128. BYTES OF DATA	
6347	030070	004737	030336		20\$:	MOV	#BASE,BDATA	:FIRST ADDRESS TO PRINT	
6348	030074	000207				JSR	PC,RPBASE	:GO PRINT DATA	
6349						RTS	PC	:RETURN	
6350									
6351	030076	105037	003560		ACTSWO:	CLRB	PSNUF	:INIT NOT ENOUGH FLAG	
6352	030102	005737	012406			TST	OPTYP	:DMR?	
6353	030106	001004				BNE	5\$:YES,BRANCH	
6354	030110	012737	000377	007454		MOV	#377,BEND	:BASE TABLE FOR DMC = 256 BYTES	
6355	030116	000403				BR	7\$:BRANCH	
6356	030120	012737	000177	007454	5\$:	MOV	#177,BEND	:BASE TABLE FOR DMR = 128 BYTES	
6357	030126	023737	003554	007454	7\$:	CMP	PSNUM,BEND	:DMC = 256 BYTES DMR = 128 BYTES	
6358	030134	101416				BLOS	10\$:YES,BRANCH	
6359	030136					PRINTF	#RPTIV,PSNUM	:PRINT ILLEGAL VALUE	
(8)	030136	013746	003554						MOV PSNUM,-(SP)
(7)	030142	012746	014370						MOV #RPTIV,-(SP)
(6)	030146	012746	000002						MOV #2,-(SP)
(3)	030152	010600							MOV SP,R0
(4)	030154	104417							TRAP C\$PNTF
(4)	030156	062706	000006						ADD #6,SP
6360	030162	112737	177777	003561		MOVB	#-1,PSGDBD	:SET BAD DATA	
6361	030170	000461				BR	30\$:RETURN	
6362	030172	013701	003554		10\$:	MOV	PSNUM,R1	:OFFSET VALUE	
6363	030176	006301				ASL	R1	:MULTIPLY BY 2	
6364	030200	005737	012406			TST	OPTYP	:DMR ?	
6365	030204	001025				BNE	15\$:YES,BRANCH	
6366	030206	012737	003432	007450		MOV	#DMCIND,INDEX	:DMC MESSAGES	
6367	030214	060137	007450			ADD	R1,INDEX	:GET RIGHT MESSAGE	
6368	030220	012737	003462	007452		MOV	#DMCEND,INDEXE	:LAST DMC MESSAGE	
6369	030226	012737	017370	007454		MOV	#BASE,BEND	:TABLE ADDRESS	
6370	030234	063737	003554	007454		ADD	PSNUM,BEND	:LAST ADDRESS	
6371	030242	012737	017370	007456		MOV	#BASE,BDATA	:BASE ADDRESS	
6372	030250	063737	003554	007456		ADD	PSNUM,BDATA	:ADD OFFSET	
6373	030256	000424				BR	20\$:GO PRINT DATA	
6374	030260	012737	003250	007450	15\$:	MOV	#DMRIND,INDEX	:SETUP FOR DMR MESSAGES	
6375	030266	060137	007450			ADD	R1,INDEX	:GET CORRECT MESSAGE	


```
6376 030272 012737 003430 007452      MOV      #DMREND,INDEXE ;LAST DMR MESSAGE
6377 030300 012737 017370 007454      MOV      #BASE,BEND    ;TABLE ADDRESS
6378 030306 063737 003554 007454      ADD      P$NUM,BEND    ;LAST ADDRESS
6379 030314 012737 017370 007456      MOV      #BASE,BDATA   ;TABLE ADDRESS
6380 030322 063737 003554 007456      ADD      P$NUM,BDATA   ;ADD OFFSET
6381 030330 004737 030336      20$:    JSR      PC,RPBASE ;GO PRINT SINGLE LOCATION
6382 030334 000207      30$:    RTS      PC      ;RETURN
```

42

6384
 6385
 6386
 6387
 6388
 6389
 6390
 6391
 6392
 6393
 6394
 6395
 6396
 6397
 6398
 6399
 6400
 6401
 6402
 6403 030336 010146
 6404 030340 010246
 6405 030342
 (7) 030342 012746 030476
 (6) 030346 012746 000001
 (3) 030352 010600
 (4) 030354 104417
 (4) 030356 062706 000004
 6406 030362 013702 007450
 6407 030366 013701 007456
 6408 030372 010137 007434
 6409 030376 112137 007430
 6410 030402 020237 007452
 6411 030406 002402
 6412 030410 013702 007452
 6413 030414 012237 007432
 6414 030420
 (10) 030420 013746 007432
 (9) 030424 005046
 (9) 030426 153716 007430
 (8) 030432 013746 007434
 (7) 030436 012746 030547
 (6) 030442 012746 000004
 (3) 030446 010600
 (4) 030450 104417
 (4) 030452 062706 000012
 6415 030456 020137 007454
 6416 030462 101743
 6417 030464 105037 003560
 6418 030470 012602
 6419 030472 012601
 6420 030474 000207
 6421
 6422 030476 047045 040445 042101
 6423 030547 045 022516 030523
 6424 030574

```

::PRINT BASE TABLE SUBROUTINE
:FUNCTIONAL DESCRIPTION - THIS ROUTINE IS USED TO PRINT DATA
: STORED IN THE BASE TABLE AREA IN MEMORY. THIS BASE
: TABLE IS UPDATED BY THE DMR OR DMC. THE USER HAS THE
: OPTION OF PRINTING THE FULL TABLE, PRINTING THE FIRST
: FEW ERROR LOCATIONS OR A SINGLE LOCATION.

:DEFINITIONS
INDEX - CONTAINS POINTER TO DMR OR DMC DATA
: DESCRIPTION MESSAGES.
INDEXE - CONTAINS POINTER TO LAST DMR OR DMC
: DESCRIPTION MESSAGES.
BEND - LAST LOCATION IN TABLE TO BE PRINTED.
BDATA - ADDRESS OF DATA TO BE PRINTED.

THE ABOVE VARIABLES MUST BE ASSIGNED THE CORRECT VALUES
BEFORE THIS SUBROUTINE IS CALLED.
  
```

```

RPBASE: MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
PRINTF #BTHEAD ;PRINT BRIEF HEADER MESSAGE

MOV #BTHEAD,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #4,SP

MOV INDEX,R2 ;POINTER TO MESSAGES
MOV BDATA,R1 ;ADDRESS OF DATA
10$: MOV R1,TEMP3 ;SAVE CURRENT ADDRESS OF DATA
MOV (R1)+,TEMP1 ;READ DATA
CMP R2,INDEXE ;END OF MESSAGES?
BLT 20$ ;NO BRANCH
MOV INDEXE,R2 ;'SEE MANUAL' MESSAGE
20$: MOV (R2)+,TEMP2 ;READ MESSAGE ADDRESS
PRINTF #DMFMT,TEMP3,<B,TEMP1>,TEMP2 ;PRINT DATA AND MESSAGE

MOV TEMP2,-(SP)
CLR -(SP)
BISB TEMP1,(SP)
MOV TEMP3,-(SP)
MOV #DMFMT,-(SP)
MOV #4,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #12,SP

CMP R1,BEND ;LAST ADDRESS ?
BLOS 10$ ;NO BRANCH
CLRB PSNUF ;CLEAR ENOUGH FLAG
MOV (SP)+,R2 ;RESTORE R2
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

BTHEAD: .ASCIZ /%N%ADDRESS%S2%ACONTENTS%S6%ADESCRIPTION/
DMFMT: .ASCIZ /%N%S1%06%S5%03%S5%T/
.EVEN
  
```

```

6426 .SBTTL PRINT EVENT LOG
6427 :PRINT THE EVENT LOG
6428 030574 010246 REPROG: MOV R2,-(SP) ;SAVE R2
6429 030576 010346 MOV R3,-(SP) ;SAVE R3
6430 030600 010446 MOV R4,-(SP) ;SAVE R4
6431 030602 013702 007550 MOV EVTPTR,R2 ;MAKE R2 A POINTER TO EVENT TABLE
6432 030606 023727 007552 177777 CMP EVTLOG,#-1 ;SEE IF EVENT TABLE IS EMPTY
6433 030614 001034 BNE RPT0 ;BR IF NO
6434 030616 PRINTS #NULEVT ;IF EMPTY TELL OPERATOR.
(7) 030616 012746 016250 MOV #NULEVT,-(SP)
(6) 030622 012746 000001 MOV #1,-(SP)
(3) 030626 010600 MOV SP,RO
(4) 030630 104416 TRAP C$PNTS
(4) 030632 062706 000004 ADD #4,SP
6435 030636 000137 031432 JMP ENDEVT ;AND END
6436
6437 030642 162702 000012 RPT: SUB #12,R2 ;NOW POINT BACK TO TOP OF ENTRY U
6438 ;JUST PRINTED
6439
6440 030646 020227 007552 CMP R2,#EVTLOG ;POINTING TO TOP OF EVNT LOG QUEUE?
6441 030652 001010 BNE RPT1 ;BR IF NO
6442 030654 012702 010454 MOV #EVTEND,R2 ;SET R2 TO POINT TO BOTTOM OF LOG
6443 030660 026227 177776 177777 CMP -2(R2),#-1
6444 030666 001007 BNE RPT0 ;IF END OF LOG IS NOT EMPTY
6445 030670 000137 031432 JMP ENDEVT ;CONTINUE...ELSE EXIT
6446
6447 030674 020237 007550 RPT1: CMP R2,EVTPTR ;ARE WE BACK TO POINTER?
6448 030700 001002 BNE RPT0 ;IF NOT CONTINUE
6449 030702 000137 031432 JMP ENDEVT ;IF SO EXIT....
6450
6451 030706 162702 000012 RPT0: SUB #12,R2 ;POINT R2 TO START OF ENTRY
6452 030712 RPTAA: PRINTS #EVTFO ;PRINT EVENT ENTRY HEADER
(7) 030712 012746 016310 MOV #EVTFO,-(SP)
(6) 030716 012746 000001 MOV #1,-(SP)
(3) 030722 010600 MOV SP,RO
(4) 030724 104416 TRAP C$PNTS
(4) 030726 062706 000004 ADD #4,SP
6453 030732 112203 MOV (R2)+,R3 ;PUT EVENT TYPE INTO R3
6454 030734 112237 010612 MOV (R2)+,EVTTC ;PUT EVENT TIME (TICKS,SECS,MINS IN TEMP LOC.S)
6455 030740 112237 010606 MOV (R2)+,EVTSEC
6456 030744 112237 010610 MOV (R2)+,EVTMIN
6457 030750 PRINTS #EVTFO,EVTMIN,EVTSEC,EVTTC,EVTLS(R3) ;PRINT EVENT TIME AND DESCRIPT.
(11) 030750 016346 010514 MOV EVTLS(R3),-(SP)
(10) 030754 013746 010612 MOV EVTTCK,-(SP)
(9) 030760 013746 010606 MOV EVTSEC,-(SP)
(8) 030764 013746 010610 MOV EVTMIN,-(SP)
(7) 030770 012746 016403 MOV #EVTFO,-(SP)
(6) 030774 012746 000005 MOV #5,-(SP)
(3) 031000 010600 MOV SP,RO
(4) 031002 104416 TRAP C$PNTS
(4) 031004 062706 000014 ADD #14,SP
6458 031010 000173 010622 JMP @RPTDSP(R3) ;DISPATCH TO DECODING SECTION FOR SPECIFIC TYPE
6459
6460 031014 012237 010614 RPTTXQ: MOV (R2)+,EVTADD ;STORE MESSAGE ADDRESS FOR PRINTING
6461 031020 012237 010616 MOV (R2)+,EVTBCT ;STORE BYTE COUNT FOR PRINTING
6462 031024 012203 MOV (R2)+,R3 ;STORE MODEM STATUS FOR PRINTING
  
```

```

6463 031026          PRINTS #EVTF2,EVTADD,EVTBCT      ;PRINT ADDR,BYTE CNT
(9) 031026 013746 010616          MOV      EVTBCT,-(SP)
(8) 031032 013746 010614          MOV      EVTADD,-(SP)
(7) 031036 012746 016432          MOV      #EVTF2,-(SP)
(6) 031042 012746 000003          MOV      #3,-(SP)
(3) 031046 010600          MOV      SP,R0
(4) 031050 104416          TRAP    C$PNTS
(4) 031052 062706 000010          ADD     #10,SP
6464 031056 004737 031442          JSR     PC,RPTMSB      ;GO PRINT MODEM STATUS
6465 031062 000137 030642          JMP     RPT            ;GO BACK FOR NEXT EVENT ENTRY
6466
6467 031066 012237 010620          RPTDER: MOV    (R2)+,EVTTMP      ;GET ADDRESS OF DEVICE INFO MESSAGE
6468 031072 012237 010650          MOV    (R2)+,DEV1          ;STORE DEVICE REG CONTENTS FOR PRINTING
6469 031076 012237 010652          MOV    (R2)+,DEV2
6470 031102          PRINTS #EVTF3,EVTTMP      ;PRINT DEVICE REG CONTENTS.
(8) 031102 013746 010620          MOV      EVTTMP,-(SP)
(7) 031106 012746 016504          MOV      #EVTF3,-(SP)
(6) 031112 012746 000002          MOV      #2,-(SP)
(3) 031116 010600          MOV      SP,R0
(4) 031120 104416          TRAP    C$PNTS
(4) 031122 062706 000006          ADD     #6,SP
6471 031126          PRINTS #EVTF3C,DEV1,DEV2
(9) 031126 013746 010652          MOV      DEV2,-(SP)
(8) 031132 013746 010650          MOV      DEV1,-(SP)
(7) 031136 012746 016516          MOV      #EVTF3C,-(SP)
(6) 031142 012746 000003          MOV      #3,-(SP)
(3) 031146 010600          MOV      SP,R0
(4) 031150 104416          TRAP    C$PNTS
(4) 031152 062706 000010          ADD     #10,SP
6472 031156 000137 030642          JMP     RPT            ;GO BACK FOR NEXT EVENT ENTRY
6473
6474 031162 005037 010650          RPTDVI: CLR    DEV1
6475 031166 005037 010652          CLR    DEV2          ;CLEAR UPPER BYTES OF DEV1 & DEV2 BEFORE USE
6476 031172 112237 010650          MOV    (R2)+,DEV1      ;STORE SETUP OPERATION PARAMETERS FOR PRINTING
6477 031176 112237 010652          MOV    (R2)+,DEV2
6478 031202 012237 010654          MOV    (R2)+,DEV3
6479 031206 012237 010656          MOV    (R2)+,DEV4
6480 031212 010246          MOV    R2,-(SP)      ;SAVE R2 ON THE STACK
6481 031214 004737 032340          JSR    PC,SHWOP      ;GO PRINT MODE, MAINT-LOOP TYPE, PARAMTERS.
6482 031220 012602          MOV    (SP)+,R2      ;RESTORE R2
6483 031222 000137 030642          JMP    RPT            ;GO BACK FOR NEXT EVENT ENTRY
6484 031226 012237 010614          RPTTEOP: MOV   (R2)+,EVTADD
6485 031232 012237 010616          MOV   (R2)+,EVTBCT
6486 031236 012237 010620          MOV   (R2)+,EVTTMP
6487 031242          PRINTS #EVTF4B,EVTADD,EVTBCT,EVTTMP      ;PRINT ADDR,RXBYTES,CMPBYTES.
(10) 031242 013746 010620          MOV      EVTTMP,-(SP)
(9) 031246 013746 010616          MOV      EVTBCT,-(SP)
(8) 031252 013746 010614          MOV      EVTADD,-(SP)
(7) 031256 012746 016755          MOV      #EVTF4B,-(SP)
(6) 031262 012746 000004          MOV      #4,-(SP)
(3) 031266 010600          MOV      SP,R0
(4) 031270 104416          TRAP    C$PNTS
(4) 031272 062706 000012          ADD     #12,SP
6488
6489 031276 000137 030642          JMP     RPT            ;THEN GO GET NEXT EVENT ENTRY
6490

```

```

6491
6492 031302 012237 010614      RPTDDE: MOV      (R2)+,EVTADD      ;STORE MESSAGE ADDRESS FOR PRINTING
6493 031306 012237 010616      MOV      (R2)+,EVTBCT      ;STORE BYTE COUNT FOR PRINTING
6494 031312 012237 010620      MOV      (R2)+,EVTTMP      ;STORE TOTAL # OF CMP ERRORS
6495 031316      PRINTS #EVT4,EVTADD,EVTBCT,EVTMP ;PRINT ADDR, BYTE CNT, # CMP ERRS
(10) 031316 013746 010620      MOV      EVTMP,-(SP)
(9) 031322 013746 010616      MOV      EVTBCT,-(SP)
(8) 031326 013746 010614      MOV      EVTADD,-(SP)
(7) 031332 012746 016555      MOV      #EVT4,-(SP)
(6) 031336 012746 000004      MOV      #4,-(SP)
(3) 031342 010600      MOV      SP,R0
(4) 031344 104416      TRAP     C$PNTS
(4) 031346 062706 000012      ADD     #12,SP
6496 031352 000137 030642      JMP     RPT      ;THEN GO GET NEXT EVENT ENTRY
6497
6498 031356      RPTDLE:
6499 031356 012237 010614      RPTDCK: MOV      (R2)+,EVTADD      ;STORE MSG ADDR FOR PRINT
6500 031362 012237 010616      MOV      (R2)+,EVTBCT      ;STORE BYTE COUNT
6501 031366 012237 010620      MOV      (R2)+,EVTTMP      ;STORE BYTE COUNT COMP
6502 031372      PRINTS #EVT4A,EVTADD,EVTBCT,EVTMP ;PRINT ADDR,RXBYTES,CMPBYTES.
(10) 031372 013746 010620      MOV      EVTMP,-(SP)
(9) 031376 013746 010616      MOV      EVTBCT,-(SP)
(8) 031402 013746 010614      MOV      EVTADD,-(SP)
(7) 031406 012746 016657      MOV      #EVT4A,-(SP)
(6) 031412 012746 000004      MOV      #4,-(SP)
(3) 031416 010600      MOV      SP,R0
(4) 031420 104416      TRAP     C$PNTS
(4) 031422 062706 000012      ADD     #12,SP
6503
6504 031426 000137 030642      JMP     RPT      ;THEN GO GET NEXT EVENT ENTRY
6505
6506 031432 012604      ENDEVT: MOV      (SP)+,R4      ;RESTORE R4,R3,R2
6507 031434 012603      MOV      (SP)+,R3
6508 031436 012602      MOV      (SP)+,R2
6509 031440 000207      RTS     PC      ;RETURN TO CALLING ROUTINE
6510
6511
6512      ;REPORT MODEM STATUS SUBROUTINE
6513      ; PART OF STATISICAL REPORTING (DUMPING EVENT LOG)
6514
6515 031442      RPTMSB: PRINTS #EVMOHD      ;PRINT MODEM STATUS HEADER
(7) 031442 012746 017143      MOV      #EVMOHD,-(SP)
(6) 031446 012746 000001      MOV      #1,-(SP)
(3) 031452 010600      MOV      SP,R0
(4) 031454 104416      TRAP     C$PNTS
(4) 031456 062706 000004      ADD     #4,SP
6516 031462 012704 010460      MOV      #MOBITS,R4      ;MAKE R4 A POINTER TO MODEM SIG. BIT DEF. TABLE
6517 031466 012705 010476      MOV      #MOMSGS,R5      ;MAKE R5 A POINTER TO MODEM MSG. POSITION TABLE
6518 031472 005714      6$: TST      (R4)      ;SEE IF BIT AVAIABLE FROM DEVICE
6519 031474 001004      BNE     7$      ;BR IF THAT MODEM SIG. AVAIABLE
6520 031476 112735 000130      MOV     #'X,a(R5)+      ;ELSE PUT 'X' IN REPORT IF SIGNAL NOT AVAILABLE
6521 031502 005724      TST      (R4)+      ;BUMP R4 TO POINT TO NEXT BIT DEFINITION
6522 031504 000407      BR     9$      ;GO SEE IF CHECKED ALL MODEM SIGNALS
6523 031506 032403      7$: BIT      (R4)+,R3      ;IF THERE, SEE IF THAT BIT IN DEVICE'S ENTRY=1
6524 031510 001403      BEQ     8$      ;BR IF BIT (SIGNAL) VALUE =0
6525 031512 112735 000061      MOV     #'1,a(R5)+      ;IF=1, PUT '1' IN REPORT MESSAGE

```

```
6526 031516 000402          BR      9$          ;GO SEE IF ALL MODEM SIGNALS CHECKED
6527 031520 112735 000060    8$:     MOVB     #'0,a(R5)+ ;IF BIT(SIGNAL)=0, PUT '0' IN REPORT MESSAGE
6528 031524 020427 010476    9$:     CMP      R4,#MOBITE ;SEE IF ALL BITS(SIGNALS) CHECKED.
6529 031530 002760          BLT     6$          ;LOOP UNTIL ALL SIGNALS(BITS) CHECKED
6530 031532          PRINTS #EVMOST ;THEN PRINT MODEM SIGNAL VALUE MESSAGE
(7) 031532 012746 017223          MOV     #EVMOST,-(SP)
(6) 031536 012746 000001          MOV     #1,-(SP)
(3) 031542 010600          MOV     SP,R0
(4) 031544 104416          TRAP   C$PNTS
(4) 031546 062706 000004          ADD    #4,SP
6531 031552 000207          RTS     PC          ;RETURN TO EVENT DECODING
6532
6533
```

6535
 6536
 6537
 6538
 6539
 6540
 6541
 6542
 6543
 6544
 6545
 6546
 6547
 6548
 6549
 6550
 6551
 6552
 6553
 6554
 6555
 6556
 6557
 6558
 6559
 6560
 6561
 6562 031554 013702 007402
 6563 031560 005003
 6564 031562
 (8) 031562 010246
 (7) 031564 012746 016242
 (6) 031570 012746 000002
 (3) 031574 010600
 (4) 031576 104417
 (4) 031600 062706 000006
 6565 031604 005737 007406
 6566 031610 001416
 6567 031612 112237 007426
 6568 031616
 (8) 031616 005046
 (8) 031620 153716 007426
 (7) 031624 012746 016224
 (6) 031630 012746 000002
 (3) 031634 010600
 (4) 031636 104417
 (4) 031640 062706 000006
 6569 031644 000411
 6570 031646
 (8) 031646 012246
 (7) 031650 012746 016233
 (6) 031654 012746 000002
 (3) 031660 010600
 (4) 031662 104417
 (4) 031664 062706 000006
 6571 031670 020237 007404

.SBTTL DUMP BYTES OR WORDS

```

++
: FUNCTIONAL DESCRIPTION:
: DUMPSR - DUMP BYTES OR WORDS SUBROUTINE
:
: THIS SUBROUTINE PRINTS THE CONTENTS OF THE LOCATIONS BETWEEN
: A STARTING AND END ADDRESS IN LOCS. 'STADD' AND 'ENADD'.
: THE WORD OR BYTE CONTENTS ARE PRINTED 8 TO A LINE WITH THE
: ADDRESS OF THE FIRST BYTE AS THE FIRST 6 OCTAL CHARS. FOLLOWED
: BY A SEMICOLON.
:
: INPUTS:
: STADD= STARTING ADDRESS (FIRST LOC. TO PRINT)
: ENADD= END ADDRESS (LAST LOCATION TO DUMP)
: BYTBIT= 1 IF SUPPOSED TO PRINT 'BYTES'
:         0 IF SUPPOSED TO PRINT 'WORDS'
:
: OUTPUTS:
: CONTENTS OF A RANGE OF LOC.S PRINTED ON THE OPERATORS CONSOLE.
:
: CALLING SEQUENCE:
: JSR PC,DUMPSR ;CALL DUMP BYTES SUBROUTINE
--

```

```

DUMPSR: MOV STADD,R2 ;SET R2 UP TO STARTING ADDR.
DUM4: CLR R3 ;CLEAR R3
PRINTF #BASM1,R2 ;PRINT ADDRESS

MOV R2,-(SP)
MOV #BASM1,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

DUM3: TST BYTBIT ;IS THIS BYTE OR WORD
BEQ DUM1 ;BR IF WORD
MOVB (R2)+,TEMP ;MOV BYTE TO TEMP
PRINTF #BASM3,<B,TEMP> ;PRINT BYTE

CLR -(SP)
BISB TEMP,(SP)
MOV #BASM3,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

DUM1: BR DUM2
PRINTF #BASM2,(R2)+ ;PRINT WORD

MOV (R2)+,-(SP)
MOV #BASM2,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #6,SP

DUM2: CMP R2,ENADD ;COMPARE FOR LAST ADD

```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) ^{E 8} 16-JUN-81 14:39 PAGE 13-59
DUMP BYTES OR WORDS

SEQ 0095

6572 031674 003005
6573 031676 005203
6574 031700 022703 000010
6575 031704 001725
6576 031706 000736
6577
6578 031710 000207
6579

BGT DUMEX
INC R3
CMP #8.,R3
BEQ DUM4
BR DUM3

DUMEX: RTS PC

:IF DONE EXIT
:ELSE BUMP R3
:HAVE WE PRINTED 8 ACCROSS
:IF SO GO BACK TO 4
:ELSE GO BACK AND PRINT ANOTHER
:BYTE OR WORD
:RETURN TO CALLER


```

6581      .SBTTL          UPDATE TOTAL CHAR. COUNT SUBROUTINE
6582
6583      :++
6584      : FUNCTIONAL DESCRIPTION:
6585      : UPDATES TOTAL CHAR. COUNT TOTCC BASED ON CURCC.
6586      : LAST MESSAGE IS TRUNCATED TO FIT INTO THE
6587      : BUFFER IF TOTAL CHAR. COUNT EXCEEDS 'BUFLIM' A MESSAGE
6588      : IS PRINTED TELLING THE OPERATOR THE TRUNCATION OCCURED.
6589
6590      : INPUTS:
6591      : CURCC= CHAR. COUNT OF MESSAGE BEING ADDED
6592      : TOTCC= TOTAL CHAR COUNT OF BUFFER ITS BEING ADDED TO
6593
6594      : OUTPUTS:
6595      : MESSAGE TO OPERATOR IF MESSAGE TRUNCATED TO FIT
6596
6597      : FUNCTIONAL SIDE EFFECTS:
6598      : LOCATION 'TEMP' USED FOR CALCULATIONS
6599
6600      : CALLING SEQUENCE:
6601      : JSR      PC,ADCC          ;UPDATED TOTAL CHAR. COUNT
6602      :--
6603
6604      031712 063737 007412 007422 ADDCC: ADD    CURCC,TOTCC      ;ADD CURRENT TO TOTAL
6605      031720 022737 001000 007422      CMP    #BUFLIM,TOTCC    ; COMPARE TO 'BUFLIM'
6606      031726 103027          ADDC1      BHIS   ADDC1          ;IF NOT MORE THEN 'BUFLIM' EXIT
6607
6608      ; PRINT MESSAGE AND TRUNCATE COUNT
6609
6610      031730          PRINTF #MSGTRU
6611      (7) 031730 012746 015272          MOV    #MSGTRU,-(SP)
6612      (6) 031734 012746 000001          MOV    #1,-(SP)
6613      (3) 031740 010600          MOV    SP,R0
6614      (4) 031742 104417          TRAP  C$PNTF
6615      (4) 031744 062706 000004          ADD    #4,SP
6616      031750 163737 007412 007422      SUB    CURCC,TOTCC      ;SUB CURRENT FROM TOTAL
6617      031756 012737 001000 007426      MOV    #BUFLIM,TEMP    ;MOV 'BUFLIM' TO TEMP
6618      031764 163737 007422 007426      SUB    TOTCC,TEMP      ;SUB TOTAL FROM 'BUFLIM'
6619      031772 013737 007426 007412      MOV    TEMP,CURCC      ;AND ESTABLISH NEW CURRENT
6620      032000 063737 007412 007422      ADD    CURCC,TOTCC     ;ADD 'ADJUSTED CURRENT' TO TOTAL CHAR. CNT.
6621      032006 000207          ADDC1: RTS    PC        ;RETURN TO CALLER

```

6619
 6620
 6621
 6622
 6623
 6624
 6625
 6626
 6627
 6628
 6629
 6630
 6631
 6632
 6633
 6634
 6635
 6636
 6637
 6638
 6639
 6640
 6641
 6642
 6643 032010
 6644 032010 010246
 6645 032012 010346
 6646 032014 013702 007416
 6647
 6648 032020 013722 007420
 6649 032024 013722 007412
 6650 032030 010237 007416
 6651 032034 013702 007410
 6652 032040 006302
 6653 032042 013737 007420 007426
 6654 032050 063737 007412 007426
 6655 032056 013703 007420
 6656 032062 016237 002150 007432
 6657 032070 016204 002176
 6658 032074 060437 007432
 6659 032100 112423
 6660 032102 020337 007426
 6661 032106 001404
 6662 032110 020437 007432
 6663 032114 001762
 6664 032116 000770
 6665 032120 063737 007412 007420
 6666 032126 012603
 6667 032130 012602
 6668 032132 000207
 6669

```
.SBTTL          BUILD MESSAGE BUFFERS SUBROUTINE

:++
: FUNCTIONAL DESCRIPTION:
: BLDBUF-- BUILD POINTER TABLE AND BUFFERS
:
: THIS SUBROUTINE ADDS A MESSAGE TO THE TRANSMIT OR EXPECT LIST
: USING THE POINTER, BYTE COUNT, AND ADDRESS PASSED TO IT.
:
: INPUTS:
: CURCC= CHAR. COUNT OF MESSAGE TO BE ADDED
: CURADD= ADDRESS OF MESSAGE TO BE ADDED
: CPTR= ADDRESS OF POINTER TABLE WORD WHERE MESSAGE POINTERS ARE
:       TO BE BUILT
: MSGTYP= VALUE TO USE AS AN INDEX TO FIND SOURCE OF MESSAGE DATA
:         INDEX INTO DMSGCT() AND DMSGAD().
:
: OUTPUTS:
: A MESSAGE ADDED TO EITHER TXBUF OR CMPBUF
: APPROPRIATE POINTERS IN PTRTAB POINTER TABLE
:
: CALLING SEQUENCE:
: JSR PC,BLDBUF          ;BUILD MESSAGE IN BUFFER AND ADD PTRS.
:--

BLDBUF:
  MOV     R2,-(SP)      ;SAVE R2 AND R3 ON THE STACK
  MOV     R3,-(SP)
  MOV     CPTR,R2

BLDB1:  MOV     CURADD,(R2)+ ;PUT CURRENT ADD ON POINTER TAB
        MOV     CURCC,(R2)+ ;PUT CURRENT CC ON POINTER TAB
        MOV     R2,CPTR     ;PUT UPDATED R2 BACK TO CURRENT POINT
        MOV     MSGTYP,R2  ;GET MESSAGE TYPE TO USE AS INDEX
        ASL     R2         ;DOUBLE FOR WORD INDEX
        MOV     CURADD,TEMP ;MOVE CURRENT ADD TO TEMP
        ADD     CURCC,TEMP  ;ADD CHAR COUNT TO IT TO GET END
        MOV     CURADD,R3  ;SET R3 TO CURRENT START ADD
BLDB2:  MOV     DMSGCT(R2),TEMP2 ;GET BYTE COUNT
        MOV     DMSGAD(R2),R4 ;PUT STARTING FROM ADD IN R4
        ADD     R4,TEMP2    ;ADD IT TO TEMP2 TO GET END OF FROM
BLDB3:  MOV     (R4)+,(R3)+ ;MOV BYTE FROM PATTERN TO BUFFER
        CMP     R3,TEMP    ;ALL DONE?
        BEQ     BLDBEX    ;IF SO EXIT
        CMP     R4,TEMP2  ;IS PATTERN COUNT EXPIRED
        BEQ     BLDB2    ;IF SO GO START AGAIN
        BR     BLDB3     ;IF NOT GET ANOTHER BYTE
BLDBEX: ADD     CURCC,CURADD ;BUMP CURADD
        MOV     (SP)+,R3  ;RESTORE R3 AND R2
        MOV     (SP)+,R2
        RTS     PC       ;RETURN TO CALLER
```

6671
6672
6673
6674
6675
6676
6677
6678
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726

```
.SBTTL CREATE FACSIMILE OF TX BUFFER AND MESSAGE LIST

:++
:FUNCTIONAL DESCRIPTION:
:   FACSIMILE: THIS ROUTINE IS USED TO CREATE A FACSIMILE OF THE
:   OF THE TRANSMIT LIST AND TRANSMIT BUFFER IN THE
:   EXPECTED LIST AND EXPECTED BUFFER. THE ROUTINE IS
:   NORMALLY CALLED WHEN USER COMMAND 'SET E [XPECT]=
:   T [TRANSMIT] IS ENTERED.

:   CALLING SEQUENCE: JSR PC,FACSIMILE

:--
:DEFINITIONS  CMPBUF = EXPECTED DATA BUFFER HOLDS MAX 512 BYTES
:              TXBUF  = TRANSMIT DATA BUFFER HOLDS MAX 512 BYTES
:              TTOTCC = NUMBER OF BYTES IN TXBUF
:              PTRTAB = TOP OF MESSAGE LIST POINTER TABLE
:              CTOTCC = NUMBER OF BYTES IN EXPECT MESSAGE
:              CMPTOT = NUMBER OF EXPECTED MESSAGES
:              CMPPTR = EXPECTED MESSAGE LIST POINTER
:              TXPTR  = TRANSMIT MESSAGE LIST POINTER
:              TXMTOT = NUMBER OF TRANSMIT MESSAGES
:              CCURAD = STORAGE ADDRESS OF MESSAGE IN CMPBUF
:              MSGLIN = MAXIMUM NUMBER OF MESSAGES THAT CAN BE STORED

:BEGIN FACSIMILE ROUTINE
:(*COPY TXBUF ==> CMPBUF*)
:..SAVE R1
:..INIT R1
:..REPEAT
:....[CMPBUF]R1=[TXBUF]R1
:....R1=R1+1
:..UNTIL R1 = BUFLIM

:(*NOW CALCULATE EXPECT LIST MESSAGE POINTER*)
:..CMPPTR = PTRTAB + (2 * MSGLIN)

:(*NOW PRIME THE WHILE - DO LOOP*)
:..TXPTR = PTRTAB
:..CCURAD = CMPBUF
:..TXPTR = TXPTR + 2
:..CTOTCC = [TXPTR]
:..CMPTOT = 0
:..WHILE TXMTOT <> CMPTOT DO
:....[CMPPTR] = CCURAD
:....CMPPTR = CMPPTR + 2
:....[CMPPTR] = CTOTCC
:....TXPTR = TXPTR + 4
:....CCURAD = CCURAD + CTOTCC
:....CTOTCC = [TXPTR]
:....CMPPTR = CMPPTR + 2
:....CMPTOT = CMPTOT + 1
:..END WHILE DO
:..CTOTCC = TTOTCC
:END FACSIMILE ROUTINE
```

```

6727
6728 032134
6729 032134 010146
6730 032136 005001
6731 032140 116161 003562 005562 10$:
6732 032146 005201
6733 032150 020127 001000
6734 032154 001371
6735
6736 032156 012701 000017
6737 032162 006301
6738 032164 006301
6739 032166 012737 006562 007336
6740 032174 060137 007336
6741 032200 005001
6742
6743
6744 032202 012737 006562 007334
6745 032210 012737 005562 007344
6746 032216 062737 000002 007334
6747 032224 017737 155104 007342
6748 032232 005037 007340
6749
6750
6751 032236 023737 007354 007340 30$:
6752 032244 001430
6753 032246 013777 007344 155062
6754 032254 062737 000002 007336
6755 032262 013777 007342 155046
6756 032270 062737 000004 007334
6757 032276 063737 007342 007344
6758 032304 017737 155024 007342
6759 032312 062737 000002 007336
6760 032320 005237 007340
6761 032324 000744
6762
6763 032326 013737 007356 007342 40$:
6764
6765
6766 032334 012601
6767 032336 000207
6768
6769

```

FACSIMILE:

```

MOV R1,-(SP) ;SAVE R1
CLR R1 ;INIT R1
MOV#B TXBUF(R1),CMPBUF(R1) ;COPY TX BUFFER TO EXPECTED BUFFER
INC R1 ;BUMP INDEX
CMP R1,#BUFLIM ;ALL DATA COPIED ?
BNE 10$ ;NO,BRANCH

20$:
MOV #MSGLIM,R1 ;MESSAGE LIMIT
ASL R1 ;MULTIPLY BY 2
ASL R1 ;MULTIPLY BY 2
MOV #PTRTAB,CMPPTR ;TOP OF POINTER TABLE
ADD R1,CMPPTR ;START OF EXPECTED POINTER TABLE
CLR R1 ;INIT R1

;SET UP WHILE - DO LOOP
MOV #PTRTAB, TXPTR ;TX POINTER NOW AT TOP OF TABLE
MOV #CMPBUF,CCURAD ;TRANSFER ADDRESS OF 1ST MESSAGE
ADD #2, TXPTR ;BUMP POINTER
MOV @TXPTR,CTOTCC ;BYTE COUNTER 1ST MESSAGE
CLR CMPTOT ;INIT EXPECTED MESSAGE COUNT

;WHILE TX MESSAGE TOTAL <> EXPECTED MESSAGE TOTAL DO
CMP TXMTOT,CMPTOT ;ALL MESSAGES COPIED ?
BEQ 40$ ;YES,BRANCH
MOV CCURAD,@CMPPTR ;TRANSFER ADDRESS OF MESSAGE
ADD #2,CMPPTR ;BUMP POINTER
MOV CTOTCC,@CMPPTR ;BYTE COUNT OF MESSAGE
ADD #4, TXPTR ;BUMP TX MESSAGE POINTER
ADD CTOTCC,CCURAD ;CALC. TRANSFER ADDRESS
MOV @TXPTR,CTOTCC ;BYTE COUNT NEXT MESSAGE
ADD #2,CMPPTR ;BUMP POINTER
INC CMPTOT ;INCREMENT MESSAGE COUNT
BR 30$ ;DO IT AGAIN

;END WHILE - DO
MOV TTOTCC,CTOTCC ;COPY TOTAL CHARACTER COUNT

;END ROUTINE
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

```

```

6771          .SBTTL          SHOW MODE OF OPERATION, LOOP TYPE AND QUALIFIERS
6772
6773          :++
6774          : FUNCTIONAL DESCRIPTION:
6775          :   SHWOP - SHOW MODE OF OPERATION, LOOP, QULAIIFIERS
6776          :   PRINTED ON THE OPERATOR'S CONSOLE.
6777
6778          : INPUTS:
6779          :   DEV1=  MODE TYPE (MODTYP)
6780          :   DEV2=  MAINT LOOP TYPE (MLTYP)
6781          :   DEV3=  'RUN PASS' COUNT (RPASS) - COUNT DOWN
6782          :   DEV4=  PARAMTERS WORD (PARAM)
6783
6784          : IMPLICIT INPUTS:
6785          :   MODES= TABLE OF ADDRESSES OF MODE NAME STRINGS
6786          :   LOOPS= TABLE OF ADDRESSES OF LOOP TYPE NAMES
6787
6788          : CALLING SEQUENCE:
6789          :   JSR PC,SHWOP
6790          :--
6791
6792 032340 013702 010650          SHWOP:  MOV    DEV1,R2          ;GET THE MODE TYPE IN R2
6793 032344 006302                ASL    R2              ;MAKE IT A WORD TABLE OFFSET
6794 032346 016237 003514 007426  MOV    MODES(R2),TEMP      ;GET ADDRESS OF MODE-IN-ASCII
6795 032354 013702 010652                MOV    DEV2,R2          ;GET MAINTENANCE LOOP TYPE
6796 032360 006302                ASL    R2
6797 032362 012737 014636 007434  MOV    #LP00,TEMP3        ;LOAD TEMP3 TO POINT TO '/LOOP='
6798 032370 005702                TST   R2              ;SEE IF /LOOP=XXXXX OR NONE
6799 032372 001003                BNE   10$             ;BR IF /LOOP= OF SOME KIND
6800 032374 012737 014635 007434  MOV    #LP0,TEMP3         ;IF NO LOOP THEN DON'T PRINT '/LOOP='
6801 032402 016237 003532 007430 10$:  MOV    LOOPS(R2),TEMP1     ;GET ADDRESS OF LOOP-IN-ASCII
6802 032410 013737 010654 007432  MOV    DEV3,TEMP2        ;GET NUMBER OF PASSES
6803 032416                PRINTS #SHF0,TEMP,TEMP3,TEMP1,TEMP2
6804 (11) 032416 013746 007432                MOV    TEMP2,-(SP)
6805 (10) 032422 013746 007430                MOV    TEMP1,-(SP)
6806 (9) 032426 013746 007434                MOV    TEMP3,-(SP)
6807 (8) 032432 013746 007426                MOV    TEMP,-(SP)
6808 (7) 032436 012746 015355                MOV    #SHF0,-(SP)
6809 (6) 032442 012746 000005                MOV    #5,-(SP)
6810 (3) 032446 010600                MOV    SP,R0
6811 (4) 032450 104416                TRAP  C$PNTS
6812 (4) 032452 062706 000014                ADD   #14,SP
6813
6814          CLR    R2              ;NOW SET UP FOR QUALIFIERS IN ASCII
6815 032456 005002                MOV    #PST,TEMP
6816 032460 012737 014715 007426  BIT    #STATB,DEV4        ;SEE IF /STATUS OR /NOSTATUS
6817 032466 032737 000001 010656  BNE   1$              ;BR IF /STATUS
6818 032474 001003                MOV    #PNST,TEMP
6819 032476 012737 014713 007426 1$:  MOV    #PCK,TEMP1
6820 032504 012737 014726 007430  BIT    #DATCKB,DEV4        ;SEE IF /CHECK OR /NOCHECK
6821 032512 032737 000002 010656  BNE   2$              ;BR IF /CHECK
6822 032520 001003                MOV    #PNCK,TEMP1
6823 032522 012737 014724 007430 2$:  MOV    #PEC,TEMP2
6824 032530 012737 014736 007432  BIT    #ECHOB,DEV4        ;SEE IF /ECHO OR /NOECHO
6825 032536 032737 000004 010656  BNE   3$              ;BR IF /ECHO
6826 032544 001003                MOV    #PNEC,TEMP2
6827 032546 012737 014734 007432

```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

K 8
MACY11 30A(1052) 16-JUN-81 14:39 PAGE 13-65
SHOW MODE OF OPERATION, LOOP TYPE AND QUALIFIERS

SEQ 0101

```
6818 032554 012737 014745 007440 3$: MOV #PMS,TEMP5 ;ASSUME /MODEM ;REV B EC
6819 032562 032737 000010 010656 BIT #MOCHK,DEV4 ;MODEM CHECK ? ;REV B EC
6820 032570 001003 BNE 5$ ;YES,BRANCH ;REV B EC
6821 032572 012737 014743 007440 MOV #PNMS,TEMP5 ;'/NOMODEM' MESSAGE ;REV B EC
6822
6840
6841 032600 5$: PRINTS #SHF1,TEMP,TEMP1,TEMP2,TEMP5 ;,TEMP3,TEMP4 **:SEE NOTE ABOVE
(11) 032600 013746 007440 MOV TEMP5,-(SP)
(10) 032604 013746 007432 MOV TEMP2,-(SP)
(9) 032610 013746 007430 MOV TEMP1,-(SP)
(8) 032614 013746 007426 MOV TEMP,-(SP)
(7) 032620 012746 015413 MOV #SHF1,-(SP)
(6) 032624 012746 000005 MOV #5,-(SP)
(3) 032630 010600 MOV SP,R0
(4) 032632 104416 TRAP C$PNTS
(4) 032634 062706 000014 ADD #14,SP
6842 032640 000207 RTS PC ;RETURN
6843
6844
```

```

6846      .SBTTL          TRAVERSE COMMAND LINE SUBROUTINES
6847
6848      :++
6849      :                P$TRV SUBROUTINE
6850      :
6851      :PARSE THE COMMAND LINE SUBROUTINE
6852      :TAKE ACTIONS (VIA ACTION TREE) AS PARCING LINE
6853      :PARSING DIRECTIONS FROM "CLI PARSING NODES"
6854      :                REGS USED:
6855      :
6856      :                R1,R5=SCRATCH                                P$NUM=NUMERIC CODE FROM DATA
6857      :                R2=ACTION CODE PARAMETER FROM TREE
6858      :                R3=PARSE TREE POINTER
6859      :                R4=INPUT STRING POINTER
6860      :CALLING SEQUENCE:
6861      :                JSR      PC,P$TRV
6862      :--
6863
6864      P$TRV:
6865      MOV      P$BUFA,R4
6866      MOV      P$TREE,R3
6867      P$TR5:  TSTB     (R4)                                ;SEE IF ANY CHARS LEFT IN INPUT STRING
6868      BEQ      P$EXIT                                     ;BR IF NO
6869      CMPB     (R3),#11.                                  ;SEE IF SPECIAL CLI CHAR CODE OR ASCII
6870      BGT      20$                                        ;BR IF REGULAR ASCII CHAR.
6871      MOVB     (R3),R5                                    ;GET SPECIAL CHAR CODE INTO R5
6872      ASL      R5
6873      MOV      10$(R5),R5                                ;BUILD TRAVERSE ROUTINE ADDRESS
6874      ADD      #10$,R5
6875      JSR      PC,(R5)                                    ;JSR TO SPECIAL CLI TRAVERSE ROUTINE
6876      BR      P$TR5                                       ;GO SEE IF MORE OF STRING LEFT
6877
6878      :TRAVERSE TABLE FOR "CLI FUNCTIONS"
6879      10$:  .WORD    TRVERR-10$                            ;TAKE ERROR ACTION
6880      .WORD    TRVEXI-10$                                ;TAKE EXIT ACTION
6881      .WORD    TRVBR-10$                                  ;TAKE BRANCH ACTION
6882      .WORD    TRVBIF-10$                                ;TEST P$GDBD & TAKE BRANCH
6883      .WORD    TRVSPA-10$                                ;SKIP SPACES OR TABS IN CMD LINE
6884      .WORD    TRVNUM-10$                                ;TRAVERSE NUMERIC FIELD
6885      .WORD    TRVALP-10$                                ;TRAVERSE ALPHABETICS
6886      .WORD    TRVALN-10$                                ;TRAVERSE ALPHANUMERICS
6887      .WORD    TRVOCT-10$                                ;SAME AS TRVNUM
6888      .WORD    TRVDEC-10$                                ;SAME AS CLINUM BUT DECIMAL
6889      .WORD    TRVSTR-10$                                ;FIND ASCIZ MATCH IN CMD LINE
6890
6891      :NOT A SPECIAL CODE
6892
6893      20$:  CMPB     (R3),(R4)                              ;SEE IF FIRST CHAR OF STRING IS A MATCH
6894      BEQ      22$                                        ;BR IF A MATCH
6895      JSR      PC,TRVBRC                                  ;IF NOT A MATCH, GO TAKE MISS BRANCH
6896      BR      P$TR5                                       ; THEN GO BACK PT'G TO MISS NODE
6897      22$:  JSR      PC,TRVACT                              ;IF A MATCH, GO DO ACTION DEFINED BY
6898      ADD      #4,R3                                       ; ACTION CODE IN CLI NODE, THEN
6899      :                ADJUST PTR TO NEXT CLI NODE
6900      INC      R4
6901      BR      P$TR5                                       ;ADJUST BUF PTR TO NEXT CHAR IF MATCH
  
```

```

6902
6903 032760 000207 PSEXIT: RTS PC ;RETURN FROM PARSER
6904
6905 ;-----
6906
6907 ;GOTO USER ACTION ROUTINE
6908 032762 116302 000001 TRVACT: MOV 1(R3),R2 ;GET ACTION CODE FROM CLI NODE
6909 032766 042702 177400 BIC #177400,R2 ;CLEAR ANY SIGN EXTENSION
6910 032772 013705 003550 MOV P$ACT,R5 ;GET ADDRESS OF CLI ACTION ROUTINE
6911 032776 004715 JSR PC,(R5) ;GO DO ACTION DEFINED BY CODE
6912 033000 000207 RTS PC ;RETURN TO CALLING CODE
6913
6914 ;TAKE BRANCH IN TREE
6915 033002 016305 000002 TRVBRC: MOV 2(R3),R5 ;GET BRANCH DISPLACEMENT FROM TREE
6916 033006 060503 ADD R5,R3 ;AND POINT R3 TO THE 'MISS' NODE
6917 033010 000207 RTS PC ;RETURN TO P$TRV
6918
6919 ;NO BRANCH TAKEN
6920 033012 062703 000004 TRVNOB: ADD #4,R3 ;THINGS OK, UPDATE R3 TO POINT TO NEXT
6921 033016 000207 RTS PC ;NODE AND RETURN TO P$TRV
6922
6923 ;-----
6924 033020 004737 032762 TRVERR: JSR PC,TRVACT ;TAKE ERROR ACTION
6925 033024 112737 177777 003561 MOVB #-1,P$GDBD ;SET ERROR RETURN FLAG
6926 033032 005726 TST (SP)+ ;GET RID OF 'JSR PUSH TO TRVERR'
6927 033034 000137 032760 JMP PSEXIT ;RETURN DIRECT TO EXIT OF P$TRV ROUTINE
6928
6929 033040 004737 032762 TRVEXI: JSR PC,TRVACT ;TAKE EXIT ACTION
6930 033044 105037 003561 CLRB P$GDBD ;SET GOOD/BAD FLAG TO 'SUCCESS (0)'
6931 033050 005726 TST (SP)+ ;GET RID OF 'JSR PUSH TO TRVEXI'
6932 033052 000137 032760 JMP PSEXIT ;RETURN DIRECT TO EXIT OF P$TRV ROUTINE
6933
6934 033056 004737 032762 TRVBR: JSR PC,TRVACT ;GO TAKE BRANCH ACTION
6935 033062 000137 033002 JMP TRVBRC
6936
6937 033066 004737 032762 TRVBIF: JSR PC,TRVACT
6938 033072 105737 003561 TSTB P$GDBD ;SEE IF P$GDBD SET OR CLEARED BY ACTION
6939 033076 001402 BEQ 1$ ;IF CLEAR FALL THRU TO NEXT NODE
6940 033100 000137 033002 JMP TRVBRC ;ELSE TAKE THE 'MISS' BRANCH
6941 033104 000137 033012 1$: JMP TRVNOB ;JUST UPDATE TO NEXT NODE IF THINGS OK
6942
6943 033110 005005 TRVSPA: CLR R5 ;CLEAR 'SPACE OR TAB FOUND' FLAG
6944 033112 121427 000011 1$: CMPB (R4),#11 ;SEE IF CHAR. IN CMD LINE= TAB
6945 033116 001003 BNE 2$ ;BR IF NO, NOT A TAB
6946 033120 005204 INC R4 ;INC INPUT STRING POINTER
6947 033122 005205 INC R5 ;INDICATE A TAB FOUND
6948 033124 000772 BR 1$ ;GO CHECK NEXT CHAR
6949
6950 033126 121427 000040 2$: CMPB (R4),#40 ;SEE IF CHAR. IN CMD LINE= SPACE
6951 033132 001003 BNE 10$ ;BR IF NO, NON-SPACE OR NON-TAB CHAR.
6952 033134 005204 INC R4 ;INC INPUT STRING POINTER
6953 033136 005205 INC R5 ;INDICATE A SPACE FOUND
6954 033140 000764 BR 1$ ;GO CHECK NEXT CHAR
6955 033142 005705 10$: TST R5 ;SEE IF ANY SPACES OR TABS FOUND
6956 033144 001404 BEQ 15$ ;BR IF NO, TAKE NO ACTION
6957 033146 004737 032762 JSR PC,TRVACT ;GO TAKE ACTION IF ANY FOUND

```


6958	033152	000137	033012		JMP	TRVNOB		:JUST GO UPDATE R3 TO NEXT NODE IF OK
6959	033156	000137	033002	15\$:	JMP	TRVBRC		:TAKE BRANCH (MISS) IF NONE FOUND
6960								
6961								
6962	033162	012737	000012	003556	TRVDEC:	MOV #10.,PSRADX		:USE DECIMAL AS RADIX AND ASSUME +
6963	033170	000137	033202		JMP	TRVNMA		
6964	033174				TRVOCT:	;(SAME AS TRVNUM SINCE DEFAULT RADIX IS OCTAL)		
6965	033174	012737	000010	003556	TRVNUM:	MOV #8.,PSRADX		:USE OCTAL AS RADIX AND ASSUME +
6966	033202	005005			TRVNMA:	CLR R5		:CLEAR DIGIT COUNTER
6967	033204	121427	000053		CMPB	(R4),#'+		:SEE IF THERE'S A + SIGN THERE
6968	033210	001001			BNE	10\$: BR IF NO
6969	033212	000406			BR	11\$: ELSE PSRADX ALREADY SAYS +, JUST BR
6970	033214	121427	000055	10\$:	CMPB	(R4),#'-		:SEE IF THERE'S A - SIGN THERE
6971	033220	001004			BNE	1\$: BR IF NO
6972	033222	112737	177777	003557	MOVB	#-1,PSRADX+1		:SET 'MINUS FLAG' (HI BYTE OF PSRADX)
6973	033230	005204		11\$:	INC	R4		:BUMP R4 TO POINT TO FIRST CHAR
6974								
6975	033232	121427	000060	1\$:	CMPB	(R4),#60		:SEE IF CHAR. LESS THAN A '0'
6976	033236	002434			BLT	2\$:BR IF YES (NOT NUMERIC)
6977	033240	121427	000067		CMPB	(R4),#67		:SEE IF CHAR. GREATER THAN A '7'
6978	033244	003426			BLE	13\$: BR IF YES
6979	033246	123727	003556	000012	CMPB	PSRADX,#10.		:SEE IF IN DECIMAL MODE
6980	033254	001417			BEQ	12\$: BR IF YES (CAN USE HIGHER LIMIT)
6981	033256	121427	000071		CMPB	(R4),#71		:SEE IF DIGIT WAS A 8 OR 9
6982	033262	003022			BGT	2\$:BR IF NON-NUMERIC
6983	033264				PRINTF	#CLIBRX		:ELSE WAS A 8 OR 9 WHEN IN OCTAL RADIX
(7)	033264	012746	012607				MOV #CLIBRX,-(SP)	
(6)	033270	012746	000001				MOV #1,-(SP)	
(3)	033274	010600					MOV SP,R0	
(4)	033276	104417					TRAP C\$PNTF	
(4)	033300	062706	000004				ADD #4,SP	
6984	033304	112737	177777	003561	MOVB	#-1,P\$GDBD		:SET ERROR RETURN FLAG
6985	033312	000474			BR	5\$: PRINT ERROR AND TAKE MISS
6986								
6987	033314	121427	000071	12\$:	CMPB	(R4),#71		:SEE IF CHAR. GREATER THAN A '9'
6988	033320	003003			BGT	2\$:BR IF YES (NOT NUMERIC)
6989	033322	005204		13\$:	INC	R4		:UPDATE CMD LINE PTR TO NEXT CHAR.
6990	033324	005205			INC	R5		:INDICATE A NUMERIC FOUND
6991	033326	000741			BR	1\$:GO LOOK AT NEXT CHAR.
6992								
6993	033330	005705		2\$:	TST	R5		:SEE IF FOUND ANY NUMERICS
6994	033332	001464			BEQ	5\$:BR IF NO, TAKE 'MISS' BRANCH
6995	033334	010401			MOV	R4,R1		:GET POINTER TO START OF NUMERIC STRING
6996	033336	160501			SUB	R5,R1		
6997	033340	005037	003554		CLR	PSNUM		:CLEAR LOC. WHERE VALUE WILL BE STORED
6998	033344	112102		3\$:	MOVB	(R1)+,R2		:GET ASCII CHAR AND CONVERT IT TO A #
6999	033346	162702	000060		SUB	#60,R2		
7000	033352	006337	003554		ASL	PSNUM		:SHIFT CURRENT VALUE TO MAKE ROOM
7001	033356	103437			BCS	7\$:ERROR IF NUMBER TOO BIG
7002	033360	013737	003554	003552	MOV	PSNUM,P\$CNT		:SAVE FOR LATER IN CASE DECIMAL RADIX
7003	033366	006337	003554		ASL	PSNUM		
7004	033372	103431			BCS	7\$:ERROR IF NUMBER TOO BIG
7005	033374	006337	003554		ASL	PSNUM		
7006	033400	103426			BCS	7\$:ERROR IF NUMBER TOO BIG
7007	033402	123727	003556	000012	CMPB	PSRADX,#10.		:SEE IF DECIMAL RADIX
7008	033410	001004			BNE	4\$:BR IF NOT EQUAL

7009	033412	063737	003552	003554	ADD	P\$CNT,P\$NUM		
7010	033420	103416			BCS	7\$:ERROR IF NUMBER TOO BIG
7011	033422	060237	003554		ADD	R2,P\$NUM	4\$:	
7012	033426	103413			BCS	7\$:ERROR IF NUMBER TOO BIG
7013	033430	005305			DEC	R5		
7014	033432	001344			BNE	3\$		
7015	033434	105737	003557		TSTB	P\$RADX+1		:SEE IF NUM WAS PRECEDED BY A - SIGN
7016	033440	001402			BEQ	15\$: BR IF NO
7017	033442	005437	003554		NEG	P\$NUM		: ELSE NEGATE THE NUMBER BEFORE LEAVING
7018	033446	004737	032762		JSR	PC,TRVACT	15\$:	:SINCE NUMERIC FOUND, GO TAKE ACTION
7019	033452	000137	033012		JMP	TRVNOB		:GO POINT R3 TO NEXT NODE
7020								
7021	033456						7\$:	:PRINT NUMBER TOO BIG ERROR
(7)	033456	012746	012565					MOV #CLINBG,-(SP)
(6)	033462	012746	000001					MOV #1,-(SP)
(3)	033466	010600						MOV SP,R0
(4)	033470	104417						TRAP C\$PNTF
(4)	033472	062706	000004					ADD #4,SP
7022	033476	112737	177777	003561	MOVB	#-1,P\$GDBD		:SET ERROR RETURN FLAG
7023	033504	000137	033002		JMP	TRVBRC	5\$:	:TAKE 'MISS' BRANCH
7024								
7025								
7026	033510	005005			TRVALP:	CLR R5		:CLEAR ALPHA FOUND FLAG
7027	033512	121427	000101		1\$:	CMPB (R4),#101		:SEE IF CHAR. LESS THAN A 'A'
7028	033516	002406				BLT 2\$:BR IF YES (NOT ALPHA)
7029	033520	121427	000132			CMPB (R4),#132		:SEE IF CHAR. GREATER THAN A 'Z'
7030	033524	003003				BGT 2\$:BR IF YES (NOT ALPHA)
7031	033526	005204				INC R4		:UPDATE CMD LINE PTR TO NEXT CHAR
7032	033530	005205				INC R5		:INDICATE AN ALPHA WAS FOUND
7033	033532	000767				BR 1\$:GO LOOK AT NEXT CHAR.
7034	033534	005705			2\$:	TST R5		:SEE IF ANY ALPHA'S WERE FOUND
7035	033536	001404				BEQ 3\$:BR IF NO
7036	033540	004737	032762			JSR PC,TRVACT		:IF ANY FOUND TAKE ACTION
7037	033544	000137	033012			JMP TRVNOB		:THEN UPDATE R3 TO NEXT NODE -NO BRANCH
7038	033550	000137	033002		3\$:	JMP TRVBRC		:NONE FOUND, TAKE MISS BRANCH
7039								
7040	033554	005005			TRVALN:	CLR R5		:CLEAR ALPHANUM FOUND FLAG
7041	033556	121427	000060		10\$:	CMPB (R4),#60		:SEE IF CHAR. LESS THAN A '0'
7042	033562	002417				BLT 2\$:BR IF YES (NOT NUMERIC OR ALPHA)
7043	033564	121427	000072			CMPB (R4),#72		:SEE IF CHAR. GREATER THAN A '9'
7044	033570	003003				BGT 1\$:BR IF YES (NOT NUMERIC)
7045	033572	005204				INC R4		:UPDATE CMD LINE PTR TO NEXT CHAR.
7046	033574	005205				INC R5		:INDICATE A NUMERIC FOUND
7047	033576	000767				BR 10\$:GO LOOK AT NEXT CHAR.
7048	033600	121427	000101		1\$:	CMPB (R4),#101		:SEE IF CHAR. LESS THAN A 'A'
7049	033604	002406				BLT 2\$:BR IF YES (NOT ALPHA)
7050	033606	121427	000132			CMPB (R4),#132		:SEE IF CHAR. GREATER THAN A 'Z'
7051	033612	003003				BGT 2\$:BR IF YES (NOT ALPHA)
7052	033614	005204				INC R4		:UPDATE CMD LINE PTR TO NEXT CHAR
7053	033616	005205				INC R5		:INDICATE AN ALPHA FOUND
7054	033620	000756				BR 10\$:GO LOOK AT NEXT CHAR.
7055	033622	005705			2\$:	TST R5		:SEE IF ANY ALPHANUM'S WERE FOUND
7056	033624	001404				BEQ 3\$:BR IF NO
7057	033626	004737	032762			JSR PC,TRVACT		:IF ANY FOUND TAKE ACTION
7058	033632	000137	033012			JMP TRVNOB		:THEN UPDATE R3 TO NEXT NODE -NO BRANCH
7059	033636	000137	033002		3\$:	JMP TRVBRC		:NONE FOUND, TAKE MISS BRANCH

```
7060
7061
7062
7063 033642 010401          TRVSTR: MOV      R4,R1          ;POINT R1 TO CMD STRING
7064 033644 010305          MOV      R3,R5
7065 033646 062705 000006  ADD      #6,R5          ;POINT R5 TO MATCH STRING FROM CLI NODE
7066 033652 005037 003552  CLR      PSCNT          ;CLEAR CHAR MATCH COUNT
7067 033656 105715          2$:  TSTB   (R5)          ;SEE IF END OF MATCH STRING YET
7068 033660 001411          BEQ     10$            ;BR IF YES
7069 033662 105711          TSTB   (R1)          ;SEE IF END OF CMD LINE YET
7070 033664 001407          BEQ     10$            ;BR IF YES
7071 033666 121115          CMPB   (R1),(R5)      ;SEE IF CHARACTERS MATCH
7072 033670 001005          BNE     10$            ;BR IF NO
7073 033672 005237 003552  INC     PSCNT          ;MATCH -INCREMENT MATCH COUNT
7074 033676 005201          INC     R1            ;UPDATE STRING POINTERS
7075 033700 005205          INC     R5
7076 033702 000765          BR      2$            ;BR TO CONTINUE CHECKING CHARS.
7077
7078 033704 005737 003552  10$:  TST   PSCNT          ;WHEN DONE SEE IF ANY MATCHES FOUND
7079 033710 001406          BEQ     15$            ;BR IF NO, GO TAKE THE MISS BRANCH
7080 033712 010104          MOV     R1,R4          ;POINT CMD POINTER TO END OF STRING &
7081 033714 004737 032762  JSR    PC,TRVACT        ;IF A MATCH FOUND, GO DO MATCH ACTION
7082 033720 066303 000004  ADD    4(R3),R3        ;UPDATE R3 TO NEXT NODE (NO BRANCH)
7083 033724 000207          RTS     PC            ; (NO RETURN THRU TRVNOB SINCE DIFFERNT
7084                                     ; DISPLACEMENT DUE TO MATCH STRING)
7085 033726 000137 033002  15$:  JMP    TRVBRC        ; GO TAKE BRANCH
7086
7087                                     ; (PARSED OK), -1 IF ILL CMD.....
7088 -----
7089
```

7091
7092
7093
7094
7095
7096
7097
7098
7099
(3)
7100
7112
7113
7114
7115
7122
7123
7130
7131
(3)
(3)

033732
033732

033732 004737 027046

033736
033736
033736 104425

.SBTTL REPORT CODING SECTION

;++
: THE REPORT CODING SECTION CONTAINS THE
: 'PRINTS' CALLS THAT GENERATE STATISTICAL REPORTS.
:--

BGNRPT

L\$RPT::

JSR PC,REPORT

;CALL SUBROUTINE TO DUMP EVENT LOG
; AND BASE TABLE

ENDRPT

L10011: TRAP C\$RPT

7133
7134
7135
7136
7137
7138
7139
7140
(3)
7141
7142
7143
7144
7145
7146
7147

.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

```

7162          .SBTTL INITIALIZE SECTION
7163
7164          :++
7165          : THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
7166          : AT THE BEGINNING OF EACH PASS.
7167          :--
7168
7169 033746      BGNINIT
7170 (3) 033746      LSINIT::
7171
7194          ::ADDED BY EC
7195          ::REVISION CHANGE B
7196 033746 005737 007464      TST      DCLFLG      ;CLEANUP & EXIT ?
7197 033752 001403              BEQ      INIT1      ;NO BRANCH
7198 033754 005037 007464      CLR      DCLFLG      ;CLEAR FLAG
7199 033760              DOCLN      ;GO CLEANUP
7200 (3) 033760 104444              TRAP      C$DCLN
7201 033762 012737 177777 007466 INIT1: MOV      #-1,RESFLG      ;SET RESTART FLAG
7202 033770              READEF   #EF.START      ;IF HERE CAUSE OF START,DO SOME INIT
7203 (3) 033770 012700 000040              MOV      #EF.START,RO
7204 (3) 033774 104447              TRAP      C$REFG
7205 033776              BCOMPLETE      START
7206 (2) 033776 103417              READEF   #EF.RESTART      ;IF HERE CAUSE OF RESTART, DO SOME INIT
7207 034000              BCS      START
7208 (3) 034000 012700 000037              MOV      #EF.RESTART,RO
7209 (3) 034004 104447              TRAP      C$REFG
7210 034006              BCOMPLETE      RESTRT
7211 (2) 034006 103515              READEF   #EF.CONTINUE      ;SEE IF WE'RE HERE CAUSE OF A CONTINUE
7212 034010              BCS      RESTRT
7213 (3) 034010 012700 000036              MOV      #EF.CONTINUE,RO
7214 (3) 034014 104447              TRAP      C$REFG
7215 034016              BNCOMPLETE      S1
7216 (2) 034016 103002              JMP      ENDIT
7217 034020 000137 034530      S1:  READEF   #EF.NEW      ;JMP IF HERE CAUSE OF A CONTINUE
7218 034024              BCC      S1
7219 (3) 034024 012700 000035              MOV      #EF.NEW,RO
7220 (3) 034030 104447              TRAP      C$REFG
7221 034032              BCOMPLETE      NEW
7222 (2) 034032 103523              BR      GETPRM      ;IF YES, BR AROUND LOGUNIT # SETUP
7223 034034 000525              BCS      NEW
7224 034036 005037 007466      START: CLR      RESFLG      ;CLEAR RESTART FLAG SINCE HERE ON START
7225 034042 005037 007526      CLR      CLKVEC      ;CLEAR CLK VECTOR PTR. AS A FLAG IN
7226 034046 012702 007522      MOV      #CLKCSR,R2      ;NO CLOCK IS FOUND.
7227 034052 012700 000114      CLOCK   L,R1      ;SETUP R2 AS A PTR. TO CLOCK INFO BLOCK
7228 (3) 034052 012700 000114              ;LOOK FOR A LINE CLOCK
7229 (3) 034056 104462              MOV      #L,RO
7230 (3) 034060 010001              TRAP      C$CLK
7231 034062              BNCOMPLETE      S2      ; IF NONE THERE GO LOOK FOR A P-CLOCK
7232 (2) 034062 103006              MOV      RO,R1
7233 034064 004737 026204      JSR      PC,CLKSET      ; GO SET UP CLOCK INFO TABLE & CLK VEC.
7234 034070 012737 000100 007532      MOV      #LCLKEN,CLKEN
7235 034076 000461              BR      RESTRT      ;SETUP THE ENABLE LINE CLOCK DATA
7236

```

```

7223 034100          S2:  CLOCK P,R1          ;LOOK FOR A P-CLOCK SINCE NO LINE CLOCK
(3) 034100 012700 000120          MOV #P,R0
(3) 034104 104462          TRAP C$CLCK
(3) 034106 010001          MOV R0,R1
7224 034110          BNCOMPLETE S3          ; IF NONE THERE GO SEE IF THIS IS LSI
(2) 034110 103017          BCC S3
7225 034112 004737 026204          JSR PC,CLKSET          ; ELSE GO SET UP CLOCK INFO & VECTOR
7226 034116 062737 000002 007522          ADD #2,CLKCSR          ;POINT CLKCSR TO P-CLK COUNT SET REG.
7227 034124 012777 001600 153370          MOV #PCLKCT,@CLKCSR          ;LOAD CLK SET REG. WITH COUNT VALUE
7228 034132 162737 000002 007522          SUB #2,CLKCSR          ;POINT CLKCSR BAC TO P-CLK CSR
7229 034140 012737 000111 007532          MOV #PCLKEN,CLKEN          ;SETUP THE ENABLE THE P-CLK DATA
7230 034146 000435          BR RESTRT
7231
7232 034150          S3:  READBUS          ;READ BUS TYPE TO SEE IF ON AN LSI
(3) 034150 104407          TRAP C$RDBU
7233 034152          BNCOMPLETE S4          ;BR IF NOT, NO CHANCE OF A CLOCK
(2) 034152 103021          BCC S4
7234 034154 012737 000100 007526          MOV #100,CLKVEC          ;LOAD 100 AS CLK VECTOR
7235 034162 005037 007524          CLR CLKBR          ;LOAD 0 AS CLK INT. LEVEL
7236 034166 012737 007532 007522          MOV #CLKEN,CLKCSR          ;KLUDGE UP THE CSR & ENABLE DATA LOCS
7237 034174          GMANID L5060,CLKHZ,D,377,50.,60.,YES
(3) 034174 104443          TRAP C$GMAN
(3) 034176 000406          BR 10000$
(4) 034200 007530          .WORD CLKHZ
(5) 034202 000052          .WORD T$CODE
(5) 034204 014771          .WORD L5060
(5) 034206 000377          .WORD 377
(5) 034210 000062          .WORD T$LLOLIM
(5) 034212 000074          .WORD T$HILIM
(3) 034214          10000$:
7238 034214 000412          BR RESTRT
7239
7240 034216          S4:  PRINTF #NOCLK          ;INFORM OPR. NO CLOCK, & EXIT INIT
(7) 034216 012746 015102          MOV #NOCLK,-(SP)
(6) 034222 012746 000001          MOV #1,-(SP)
(3) 034226 010600          MOV SP,R0
(4) 034230 104417          TRAP C$PNTF
(4) 034232 062706 000004          ADD #4,SP
7241 034236          EXIT INIT          TRAP C$EXIT
(3) 034236 104432          (3) 034240 000404          .WORD L10013-.
7242
7243 034242 005037 007534          RESTRT: CLR TIMMIN          ;CLEAR TIME SINCE START LOCATIONS
7244 034246 005037 007536          CLR TIMSEC
7245 034252 013737 007530 007540          MOV CLKHZ,TIMTCK          ;LOAD TICKS/SEC
7246 034260 012702 007552          MOV #EVTLOG,R2          ;INIT EVENT TABLE TO ALL 1'S AFTER EACH
7247 034264 010237 007550          MOV R2,EVTPTR          ; START OR RES AND INIT TABLE POINTER
7248 034270 012722 177777          1$:  MOV #-1,(R2)+          ;SEE IF REACHED END OF TABLE
7249 034274 020227 010454          CMP R2,#EVTEND          ;LOOP UNTIL DONE
7250 034300 001373          BNE 1$
7251
7252 034302 012737 177777 007460          NEW:  MOV #-1,LOGUNT          ;INITIALIZE LOGICAL UNIT #
7253
7254 034310 005237 007460          GETPRM: INC LOGUNT          ;POINT TO NEXT LOGICAL UNIT
7255 034314 023737 007460 002012          CMP LOGUNT,L$UNIT          ;SEE IF PAST MAX. LOG. UNIT #
7256 034322 002367          BGE NEW          ;BR IF YES, AND START OVER

```

```

7257
7258 034324          GPHARD LOGUNT,R1          ;GET THE P-TABLE FOR THIS LOG. UNIT
      (3) 034324 013700 007460          MOV LOGUNT,RO
      (3) 034330 104442          TRAP C$GPHRD
      (3) 034332 010001          MOV RO,R1
7259 034334          BNCOMPLETE GETPRM          ;IF NO P-TABLE AVAIL., GO GET NEXT ONE
      (2) 034334 103365          BCC GETPRM
7260
7261 034336 011137 007474          MOV (R1),FHDPLX          ;PUT FULL OR HALF DUPLEX ANSWER IN LOC.
7262
7273          ;DEVICE DEPENDENT PART OF GETTING INFO FROM P-TABLE
7274
7275
7276 034342 016137 000002 012360          MOV 2(R1),SEL0          ;STORE AWAY CSR ADDRESSES
7277 034350 016137 000002 012362          MOV 2(R1),BSEL1
7278 034356 005237 012362          INC BSEL1
7279 034362 016137 000002 012364          MOV 2(R1),SEL2
7280 034370 062737 000002 012364          ADD #2,SEL2
7281 034376 016137 000002 012366          MOV 2(R1),BSEL3
7282 034404 062737 000003 012366          ADD #3,BSEL3
7283 034412 016137 000002 012370          MOV 2(R1),SEL4
7284 034420 062737 000004 012370          ADD #4,SEL4
7285 034426 016137 000002 012372          MOV 2(R1),BSEL5
7286 034434 062737 000005 012372          ADD #5,BSEL5
7287 034442 016137 000002 012374          MOV 2(R1),SEL6
7288 034450 062737 000006 012374          ADD #6,SEL6
7289 034456 016137 000002 012376          MOV 2(R1),BSEL7
7290 034464 062737 000007 012376          ADD #7,BSEL7
7291
7292 034472 016137 000004 012400          MOV 4(R1),INVEC          ;STORE AWAY INPUT INTERRUPT VECTOR
7293 034500 016137 000004 012402          MOV 4(R1),OUTVEC
7294 034506 062737 000004 012402          ADD #4,OUTVEC          ;BUILD OUTPUT INTERRUPT VECTOR
7295 034514 016137 000006 012404          MOV 6(R1),INTPRI          ;STORE AWAY INTERRUPT PRIORITY
7296 034522 016137 000012 012406          MOV 12(R1),OPTYP          ;STORE AWAY DEVICE OPTION TYPE
7297
7298          ENDIT:
7299          SETVEC CLKVEC,#CLKINT,#340          ;SETUP CLOCK VECTOR
      (7) 034530 012746 000340          MOV #340,-(SP)
      (6) 034534 012746 026230          MOV #CLKINT,-(SP)
      (5) 034540 013746 007526          MOV CLKVEC,-(SP)
      (4) 034544 012746 000003          MOV #3,-(SP)
      (3) 034550 104437          TRAP C$SVEC
      (2) 034552 062706 000010          ADD #10,SP
7300
7301          ;DEVICE DEPENDENT VECTOR SETUP
7302
7311 034556          SETVEC INVEC,#DVINS,INTPRI          ;SETUP INPUT INTERRUPT VECTOR
      (7) 034556 013746 012404          MOV INTPRI,-(SP)
      (6) 034562 012746 044620          MOV #DVINS,-(SP)
      (5) 034566 013746 012400          MOV INVEC,-(SP)
      (4) 034572 012746 000003          MOV #3,-(SP)
      (3) 034576 104437          TRAP C$SVEC
      (2) 034600 062706 000010          ADD #10,SP
7312 034604          SETVEC OUTVEC,#DVOUTS,INTPRI          ;SETUP OUTPUT INTERRUPT VECTOR
      (7) 034604 013746 012404          MOV INTPRI,-(SP)
      (6) 034610 012746 044630          MOV #DVOUTS,-(SP)
  
```


(5)	034614	013746	012402			MOV	OUTVEC,-(SP)
(4)	034620	012746	000003			MOV	#3,-(SP)
(3)	034624	104437				TRAP	C\$SVEC
(2)	034626	062706	000010			ADD	#10,SP
7313							
7314	034632			SETPRI	#PRI00		
(3)	034632	012700	000000				
(3)	034636	104441					
7315	034640			EXIT	INIT		
(3)	034640	104432					
(3)	034642	000002					
7316							
7328							
7329							
7330				.EVEN			
7331	034644			ENDINIT			
(3)	034644						
(3)	034644	104411					

L10013: TRAP C\$INIT

7333
7334
7335
7336
7337
7338
7339
7340
7341
7342 034646
(3) 034646
7343
7350
7351 034646
(3) 034646
(3) 034646 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::

ENDAUTO

L10014: TRAP C\$AUTO

7353
7354
7355
7356
7357
7358
7359

.SBTTL CLEANUP CODING SECTION

;++
: THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
: AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
:--

7360 034650
(3) 034650

BGNCLN

L\$CLEAN::

7370 034650 004737 045714
7371 034654 005077 152642

JSR PC,DVBTUP
CLF @CLKCSR
SETPRI #PRI07

;GO UPDATE BASE TABLE
;DISABLE CLOCK
;SET PROCESSOR PRIORITY BACK TO 7

7372 034660
(3) 034660 012700 000340
(3) 034664 104441

MOV #PRI07,R0
TRAP C\$SPRI

7373
7374 034666
(3) 034666 104432
(3) 034670 000002

EXIT CLN

TRAP C\$EXIT
.WORD L10015-

7375
7387
7388
7389

.EVEN

7390 034672
(3) 034672
(3) 034672 104412

ENDCLN

L10015: TRAP C\$CLEAN

7392
7393
7394
7395
7396
7397
7398
7399 034674
(3) 034674
7400
7409
7410 034674
(4) 034674 000167
(3) 034676 000000
7411
7423
7424
7425
7426 034700
(3) 034700
(3) 034700 104453

.SBTTL DROP UNIT SECTION

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

BGNDU

LSDU::

EXIT DU

.WORD JSJMP
.WORD L10016-2-

.EVEN

ENDDU

L10016: TRAP C\$DU

7428
7429
7430
7431
7432
7433
7434
7435
7436
(3)
7437
7446
7447
(4)
(3)
7448
7460
7461
7462
7463
(3)
(3)
7464
7465

.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

LSAU::

EXIT AU

.WORD JSJMP
.WORD .L10017-2-

.EVEN

ENDAU

L10017: TRAP CSAU

034702
034702

034702
034702 000167
034704 000000

034706
034706
034706 104452

7467
 7468
 7469
 7470
 7471
 7472
 7473
 7474
 7475
 7482
 7488
 7489 034710
 (3) 034710
 7490
 7496
 7497
 7498
 7499 034710 013777 007532 152604
 7500
 7501 034716
 7502 034716 005001
 7503 034720 012737 000001 007542
 7504 034726 005737 007542
 7505 034732 001412
 7506 034734 005301
 7507 034736 001373
 7508 034740
 (7) 034740 012746 015102
 (6) 034744 012746 000001
 (3) 034750 010600
 (4) 034752 104417
 (4) 034754 062706 000004
 7509
 7510 034760 005737 007466
 7511 034764 001117
 7512
 7513
 7514
 7515 034766 005037 007422
 7516 034772 005037 007356
 7517 034776 005037 007342
 7518 035002 012701 006562
 7519 035006 010137 007334
 7520 035012 005037 007332
 7521 035016 012702 000017
 7522 035022 006302
 7523 035024 006302
 7524 035026 010137 007336
 7525 035032 060237 007336
 7526
 7527 035036 012737 000005 007410
 7528 035044 013737 002162 007412
 7529 035052 012737 003562 007360
 7530 035060 012737 005562 007344
 7531
 7532 035066 013737 007360 007420

.SBTTL TEST 1: SETUP AND MODES OF OPERATION

```

:++
: TEST TO DETECT FAULTS IN THE DATA COMMUNICATION LINK. THIS TEST WILL
: THE PROVIDE COVERAGE NECESSARY TO ISOLATE FAILURES TO THE COMPUTER
: EQUIPMENT, THE COMMUNICATION LINK, OR THE MODEM.
:--
  
```

BGNTST

T1::

.SBTTL PROGRAM SETUP SECTION

```

MOV CLKEN,@CLKCSR ;ENABLE THE CLOCK

GTXRXB:
GTRA2: CLR R1
MOV #1,TIMER1 ;SET TIMER TO COUNT 1 TICK
1$: TST TIMER1 ;CHECK FOR IT TO BE COUNTED OFF
BEQ GTRA3 ;BRANCH IF CLOCK EXISTS (COUNTED A TICK)
DEC R1
BNE 1$ ;KEEP CHECKING UNTIL R1 DOES FULL COUNTDOWN
PRINTF #NOCLK ;PRINT BAD CLK MSG AND WARN OF HANG IF TIMEOUT
MOV #NOCLK,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C$PNTF
ADD #4,SP

GTRA3: TST RESFLG ;SEE IF HERE AFTER A RESTART.
BNE GTRA5 ;BR IF HERE CAUSE OF A RESTART

; CLEAR COUNTS AND SET UP DEFAULTS

GTRA4: CLR TOTCC ;CLEAR TOTAL CHAR. COUNT TEMP. LOC.
CLR TTOTCC ; CLEAR TOTAL CHAR. COUNT FOR TX BUFF
CLR CTOTCC ; CLEAR TOTAL CHAR. COUNT FOR CMP BUFF
MOV #PTRTAB,R1 ;INIT TRANSMIT MESSAGE POINTER
MOV R1, TXPTR
CLR RXPTR ; ZERO RX POINTER
MOV #MSGLIM,R2
ASL R2
ASL R2
MOV R1, CMPPTR
ADD R2, CMPPTR ;INIT COMPARE MESSAGE POINTER

MOV #5,MSGTYP ;SET UP DEFAULT MSG TYPE (QUICK FOX - ITEP MSG)
MOV MSG5C,CURCC ;SET UP DEFAULT CHAR COUNT
MOV #TXBUF,TCURAD ;SET UP CURRENT ADD TO START OF TX BUFFER
MOV #CMPBUF,CCURAD ;SET UP CURRENT ADD TO START OF CMP BUFFER

MOV TCURAD,CURADD ;SETUP CURRENT ADDR TO START OF TXBUF
  
```

```

7533 035074 013737 007334 007416 MOV TXPTR,CPTR ;SETUP CURRENT POINTER TABLE POINTER FOR TXBUF
7534 035102 004737 032010 JSR PC,BLDBUF ; GO BUILD POINTER TABLE AND BUFFER
7535 035106 012737 000001 007354 MOV #1, TXMTOT ;BUMP TOTAL MESSAGE COUNT
7536
7537 035114 013737 007336 007416 MOV CMPPTR,CPTR ;SET UP START OF COMPARE POINTER TABLE
7538 035122 013737 007344 007420 MOV CCURAD,CURADD ;SET UP CURRENT ADDR. TO START OF CMPBUF
7539 035130 012737 000005 007410 MOV #5,MSGTYP
7540 035136 013737 002162 007412 MOV MSG5C,CURCC
7541 035144 004737 032010 JSR PC,BLDBUF ;PUT DEFAULT MESSAGE INTO CMPBUF
7542 035150 012737 000001 007340 MOV #1,CMPTOT ;BUMP THE COMP MESSG COUNT
7543 035156 012737 000003 007470 MOV #ACT,MODTYP ;SET DEFAULT MODE= ACTIVE
7544 035164 005037 007472 CLR MLTYP ;SET DEFAULT MAINTENANCE LOOP MODE =NONE
7545 035170 012737 000001 007500 MOV #1,RPASS ;SET UP DEFAULT 'RUN PASS' COUNT TO 1
7546 035176 012737 000002 007476 MOV #2,PARAM ;SET UP PROG. PARAMETERS - DATACHECKING ENABLED
7547 ; OPERATOR STATUS MSGS. PRINT OFF
7548 035204 PRINTF #HLPO
(7) 035204 012746 013147 MOV #HLPO,-(SP)
(6) 035210 012746 000001 MOV #1,-(SP)
(3) 035214 010600 MOV SP,RO
(4) 035216 104417 TRAP C$PNTF
(4) 035220 062706 000004 ADD #4,SP
7549 035224 013737 007470 010650 GTRAS: MOV MODTYP,DEV1
7550 035232 013737 007472 010652 MOV MLTYP,DEV2
7551 035240 013737 007500 010654 MOV RPASS,DEV3
7552 035246 013737 007476 010656 MOV PARAM,DEV4
7553 035254 004737 032340 JSR PC,SHWOP ;PRINT TO OPERATOR THE CURRENT MODE.....
7554
7555 035260 MANUAL ;SEE IF MANUAL INTERVENTION ALLOWED
(3) 035260 104450 TRAP C$MANI
7556 035262 BCOMPLETE GETCL ; BR IF YES (UAM=0 AND NOT CHAINED)
(2) 035262 103412 BCS GETCL
7557 035264 005737 007500 TST RPASS ;SEE IF THIS IS FIRST 'DCLT PASS'
7558 035270 001002 BNE 1$ ; BR IF NOT COMPLETED 1 PASS
7559 035272 EXIT TST ; IF DONE 1 PASS IN UNATTENDED MODE - EXIT
(3) 035272 104432 TRAP C$EXIT
(3) 035274 010560 .WORD L10020-
7560 035276 012737 000001 007472 1$: MOV #TTL,MLTYP ;SET UP DEFAULT FOR UNATTENDED MODE
7561 035304 000137 040334 JMP GTR9 ; 'R M=ACT/LO=I/PAS=1/NOST/CH' AND RUN
7562
7563 .SBTTL COMMAND LINE FETCH & INTERPRETATION SECTION
7564
7565 035310 105037 003561 GETCL: CLRB P$GDBD ;CLEAR CMD LINE PARSING ERROR FLAGS
7566 035314 105037 003560 CLRB P$NNUF
7567 035320 GMANID CLISPM,CMDBUF,A,0,1,72.,NO ;GET A COMMAND LINE FROM OPR.
(3) 035320 104443 TRAP C$GMAN
(3) 035322 000406 BR 10000$
(4) 035324 003060 .WORD CMDBUF
(5) 035326 000142 .WORD T$CODE
(5) 035330 012476 .WORD CLISPM
(5) 035332 000000 .WORD 0
(5) 035334 000001 .WORD T$LOLIM
(5) 035336 000110 .WORD T$HILIM
(3) 035340 10000$:
7568 035340 012737 003060 003544 MOV #CMDBUF,P$BUFA
7569 035346 012737 010660 003546 MOV #CLITRE,P$TREE
7570 035354 012737 036302 003550 MOV #CLIACT,P$ACT

```

```

7571 035362 005037 003204 CLR QUALFG ;CLEAR QUALIFIER FLAG LOCATION
7572 035366 004737 032642 JSR PC,PSTRV ;GO PARSE COMMAND LINE
7573 035372 105737 003561 TSTB PSGDBD ;SEE IF PARSED OK OR AN ERROR
7574 035376 001412 BEQ 1$
7575 035400 PRINTF #CLIERM
(7) 035400 012746 012512 MOV #CLIERM,-(SP)
(6) 035404 012746 000001 MOV #1,-(SP)
(3) 035410 010600 MOV SP,R0
(4) 035412 104417 TRAP C$PNTF
(4) 035414 062706 000004 ADD #4,SP
7576 035420 000137 035310 JMP GETCL
7577 035424 105737 003560 1$: TSTB PSNNUF ;SEE IF INCOMPLETE COMMAND TYPED
7578 035430 001412 BEQ 10$
7579 035432 PRINTF #CLINUF
(7) 035432 012746 012542 MOV #CLINUF,-(SP)
(6) 035436 012746 000001 MOV #1,-(SP)
(3) 035442 010600 MOV SP,R0
(4) 035444 104417 TRAP C$PNTF
(4) 035446 062706 000004 ADD #4,SP
7580 035452 000137 035310 JMP GETCL
7581
7582 :REV B BY EC
7583 035456 023727 003202 000060 10$: CMP KEYWD1,#SETET ;WAS "SET EXPECT = TRANMIT" ENTERED?
7584 035464 001711 BEQ GETCL ;YES,BRANCH
7585
7586 035466 023727 003202 000005 CMP KEYWD1,#HLP ;SEE IF HELP WAS TYPED
7587 035474 001705 BEQ GETCL ;GO GET CMD AGAIN IF YES
7588 035476 023727 003202 000055 CMP KEYWD1,#PRNT ;SEE IF PRINT WAS TYPED
7589 035504 001701 BEQ GETCL ;GO GET CMD AGAIN IF YES
7590 035506 023727 003202 000004 CMP KEYWD1,#RUN ;SEE IF RUN WAS TYPED
7591 035514 001002 BNE 11$ ;BR IF NO
7592 035516 000137 040334 JMP GTR9 ;START EXEC. IF YES
7593 035522 023727 003202 000052 11$: CMP KEYWD1,#DMPS ;SEE IF DUMP WAS TYPED
7594 035530 001004 BNE 12$ ;BR IF NO
7595 035532 004737 031554 JSR PC,DUMPSR ;ELSE, DUMP PART OF MEMORY
7596 035536 000137 035310 JMP GETCL ;THEN RETURN TO GET ANOTHER CMD.
7597 ;;EXIT COMMAND IS A REVISION B CHANGE BY EC
7598 035542 023727 003202 000057 12$: CMP KEYWD1,#EXIT ;EXIT COMMAND ?
7599 035550 001005 BNE 13$ ;NO,BRANCH
7600 035552 012737 000001 007464 MOV #1,DCLFLG ;SET CLEANUP & EXIT FLAG
7601 035560 EXIT TST ;GO BACK TO INIT ROUTINE
(3) 035560 104432 TRAP C$EXIT
(3) 035562 010272 .WORD L10020-
7602 035564 023727 003202 000001 13$: CMP KEYWD1,#CLEAR ;SEE IF CLEAR WAS TYPED
7603 035572 001646 BEQ GETCL ;IF YES, BACK TO GET ANOTHER CMD.
7604 035574 023727 003202 000002 CMP KEYWD1,#SHOW ;SEE IF SHOW WAS TYPED
7605 035602 001642 BEQ GETCL ;IF YES, BACK TO GET ANOTHER CMD.
7606 035604 023727 003202 000010 4$: CMP KEYWD1,#SETEXP ;SEE IF SET EXPECTED
7607 035612 001512 BEQ 2$ ;BR IF YES (A SETEXP WAS TYPED)
7608 035614 013737 007356 007422 5$: MOV TTOTCC,TOTCC
7609 035622 023727 007422 001000 CMP TOTCC,#BUFLIM ;SEE IF BUFFER ALREADY FULL
7610 035630 002414 BLT 15$ ;BR IF NOT FULL (BUFLIM # OF CHARS.)
7611 035632 PRINTF #MSGTRN,#BUFEX ;ELSE TELL OPR. AND DON'T BUILD MSG.
(8) 035632 012746 015223 MOV #BUFEX,-(SP)
(7) 035636 012746 015241 MOV #MSGTRN,-(SP)
(6) 035642 012746 000002 MOV #2,-(SP)

```


(3)	035646	010600							MOV	SP,RO
(4)	035650	104417							TRAP	C\$PNTF
(4)	035652	062706	000006						ADD	#6,SP
7612	035656	000137	035310							
7613	035662	005737	007356	15\$:	JMP	GETCL				; THEN GO GET A NEW COMMAND
7614	035666	001002			TST	TTOTCC				; IF FIRST 'SET' THEN GET RID OF DEFAULT
7615	035670	005037	007354		BNE	6\$				
7616	035674	012737	006562	007334	6\$:	CLR	TXMTOT			; GET POSITION OF END OF TX LIST
7617	035702	013701	007354		MOV	#PTRTAB, TXPTR				
7618	035706	020127	000017		MOV	TXMTOT, R1				; SEE IF MSG COUNT EXCEEDED.
7619	035712	002414			CMP	R1, #MSGLIM				; BR IF NO
7620	035714				BLT	17\$; ELSE TELL OPR. AND DON'T BUILD MSG.
(8)	035714	012746	015163		PRINTF	#MSGTRN, #TABEX				
(7)	035720	012746	015241						MOV	#TABEX, -(SP)
(6)	035724	012746	000002						MOV	#MSGTRN, -(SP)
(3)	035730	010600							MOV	#2, -(SP)
(4)	035732	104417							MOV	SP,RO
(4)	035734	062706	000006						TRAP	C\$PNTF
7621	035740	000137	035310						ADD	#6,SP
7622	035744	006301		17\$:	JMP	GETCL				; THEN GO GET A NEW COMMAND.
7623	035746	006301			ASL	R1				; # OF MSGS *4 = NEXT FREE PTR BLOCK
7624	035750	060137	007334		ASL	R1				
7625	035754	013737	007334	007416	ADD	R1, TXPTR				
7626	035762	013737	007360	007420	MOV	TXPTR, CPTR				; SETUP CHAR. COUNT, CURRENT ADDR, & PTR
7627	035770	004737	031712		MOV	TCURAD, CURADD				
7628	035774	004737	032010		JSR	PC, ADDCC				; ADD IN CHAR. COUNT AND CHECK TOTAL
7629	036000	013737	007416	007334	JSR	PC, BLDBUF				; GO BUILD MESSAGE IN BUFFER AND PTRS.
7630	036006	013737	007422	007356	MOV	CPTR, TXPTR				
7631	036014	013737	007420	007360	MOV	TOTCC, TTOTCC				; UPDATE CHAR. COUNT, CURR ADDR, & PTR
7632	036022	005237	007354		MOV	CURADD, TCURAD				
7633	036026	005337	003206		INC	TXMTOT				
7634	036032	001270			DEC	QUALVL				; DEC THE COPY COUNT
7635	036034	000137	035310		BNE	5\$				
7636					JMP	GETCL				
7637	036040	013737	007342	007422	2\$:	MOV	CTOTCC, TOTCC			; SETUP CHAR. COUNT, CURR. ADDR. & PTR
7638	036046	023727	007422	001000	CMP	TOTCC, #BUFLIM				; SEE IF BUFFER ALREADY FULL
7639	036054	002414			BLT	16\$; BR IF NOT FULL (BUFLIM # OF CHARS.)
7640	036056				PRINTF	#MSGTRN, #BUFEX				; ELSE TELL OPR. AND DON'T BUILD MSG.
(8)	036056	012746	015223						MOV	#BUFEX, -(SP)
(7)	036062	012746	015241						MOV	#MSGTRN, -(SP)
(6)	036066	012746	000002						MOV	#2, -(SP)
(3)	036072	010600							MOV	SP,RO
(4)	036074	104417							TRAP	C\$PNTF
(4)	036076	062706	000006						ADD	#6,SP
7641	036102	000137	035310							
7642	036106	005737	007342	16\$:	JMP	GETCL				; THEN GO GET A NEW COMMAND
7643	036112	001002			TST	CTOTCC				; IF FIRST 'SET' THEN GET RID OF DEFAULT
7644	036114	005037	007340		BNE	7\$				
7645	036120	012701	006562	7\$:	CLR	CMPTOT				
7646	036124	012702	000017		MOV	#PTRTAB, R1				
7647	036130	006302			MOV	#MSGLIM, R2				
7648	036132	006302			ASL	R2				
7649	036134	010137	007336		ASL	R2				
7650	036140	060237	007336		MOV	R1, CMPPTR				
7651	036144	013701	007340		ADD	R2, CMPPTR				; INIT COMPARE MESSAGE POINTER
7652	036150	020127	000017		MOV	CMPTOT, R1				
					CMP	R1, #MSGLIM				; SEE IF MSG COUNT EXCEEDED.

7653 036154 002414
7654 036156
(8) 036156 012746 015163
(7) 036162 012746 015241
(6) 036166 012746 000002
(3) 036172 010600
(4) 036174 104417
(4) 036176 062706 000006
7655 036202 000137 035310
7656 036206 006301
7657 036210 006301
7658 036212 060137 007336
7659 036216 013737 007336 007416
7660 036224 013737 007344 007420
7661 036232 004737 031712
7662 036236 004737 032010
7663 036242 013737 007416 007336
7664 036250 005237 007340
7665 036254 013737 007420 007344
7666 036262 013737 007422 007342
7667 036270 005337 003206
7668 036274 001261
7669 036276 000137 035310
7670
7671
7672
7673
7674

18\$:

BLT 18\$
PRINTF #MSGTRN,#TABEX

JMP GETCL
ASL R1
ASL R1
ADD R1,CMPPTR
MOV CMPPTR,CPTR
MOV CCURAD,CURADD
JSR PC,ADDCC
JSR PC,BLDBUF
MOV CPTR,CMPPTR
INC CMPTOT
MOV CURADD,CCURAD
MOV TOTCC,CTOTCC
DEC QUALVL
BNE 2\$
JMP GETCL

; BR IF NO
; ELSE TELL OPR. AND DON'T BUILD MSG.
MOV #TABEX,-(SP)
MOV #MSGTRN,-(SP)
MOV #2,-(SP)
MOV SP,RO
TRAP C\$PNTF
ADD #6,SP

; THEN GO GET A NEW COMMAND.
;# OF MSGS *4 = NEXT FREE PTR BLOCK

;ADD IN XHAR. COUNT AND CHECK TOTAL

;UPDATE CHAR. COUNT, CURR ADDR. & PTR
;IF COPY WAS GIVEN, PUT MSG IN BUFF
; AGAIN
;GO BACK UNTIL GET A 'RUN'

7676
 7677
 7678
 7679
 7680 036302
 7681 036302 006302
 7682 036304 016202 036320
 7683 036310 062702 036320
 7684 036314 004712
 7685 036316 000207
 7686
 7687
 7688 036320 000150
 7689 036322 000152
 7690 036324 000162
 7691 036326 001604
 7692 036330 000262
 7693 036332 000172
 7694 036334 000306
 7695 036336 000434
 7696 036340 000756
 7697 036342 000766
 7698 036344 001004
 7699 036346 001014
 7700 036350 001024
 7701 036352 001116
 7702 036354 001612
 7703 036356 001136
 7704 036360 001216
 7705 036362 001224
 7706 036364 001234
 7707 036366 001244
 7708 036370 001254
 7709 036372 001264
 7710 036374 001302
 7711 036376 001370
 7712 036400 001400
 7713 036402 001420
 7714 036404 001426
 7715 036406 001436
 7716 036410 001446
 7717 036412 001456
 7718 036414 001504
 7719 036416 001514
 7720 036420 001620
 7721 036422 001634
 7722 036424 001666
 7723 036426 001676
 7724 036430 001706
 7725 036432 001716
 7726 036434 001726
 7727 036436 001736
 7728 036440 000142
 7729 036442 001174
 7730 036444 000712
 7731 036446 000742

.SBTTL ACTION TABLE AND ROUTINES
 : USER MUST CLEAR/SET PSGDBD IF USE 'CLIBIF' IN CONNECTION WITH ACTION
 : R2 WILL HOLD ACTION CODE FROM PARSING (CLI) NODE
 :
 : 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 ACTIONS TAKEN
 10\$: .WORD ACTNUL-10\$;NULL
 .WORD ACTCLR-10\$;CLEAR
 .WORD ACTSHO-10\$;SHOW
 .WORD ACTCHK-10\$;CHECK
 .WORD ACTRUN-10\$;RUN
 .WORD ACTHLP-10\$;HELP
 .WORD ACTCSE-10\$;CLEAR OR SHOW EXPECTED
 .WORD ACTCST-10\$;CLEAR OR SHOW TRANSMIT
 .WORD ACTSTE-10\$;SET EXPECTED
 .WORD ACTSTT-10\$;SET TRANSMIT
 .WORD ACTSZE-10\$;SIZE
 .WORD ACTCOP-10\$;COPY
 .WORD ACTNUM-10\$;NUMERIC VALUE FOR SIZE OR COPY
 .WORD ACTOPM-10\$;QUOTED MESSAGE FROM USER
 .WORD ACTSTS-10\$;STATUS
 .WORD ACTEQO-10\$;END OF QUOTED MESSAGE FROM USER
 .WORD ACTMS0-10\$;ONES
 .WORD ACTMS1-10\$;ZEROS
 .WORD ACTMS2-10\$;1ALT
 .WORD ACTMS3-10\$;0ALT
 .WORD ACTMS4-10\$;ITEP
 .WORD ACTMS5-10\$;CCITT
 .WORD ACTMS6-10\$;ALPHA
 .WORD ACTATV-10\$;ACTIVE MODE
 .WORD ACTPAS-10\$;PASSIVE MODE
 .WORD ACTREC-10\$;RECEIVE MODE
 .WORD ACTLIS-10\$;LISTEN MODE
 .WORD ACTDLL-10\$;DOWNLINE LOAD
 .WORD ACTTRA-10\$;TRANSMIT MODE
 .WORD ACTTAL-10\$;TALK MODE
 .WORD ACTNO-10\$;NO
 .WORD ACTECH-10\$;ECHO
 .WORD ACTCRC-10\$;SET CRC BIT
 .WORD ACTPRO-10\$;SET PROTOCOL BIT
 .WORD ACTRPS-10\$;STATUS
 .WORD ACTMOP-10\$;REMOTE STATION IN MAINTENANCE LOOP MODE
 .WORD ACTTLP-10\$;INTERNAL TTL
 .WORD ACTCLP-10\$;CABLE LOOP
 .WORD ACTLLP-10\$;LOCAL MODEM LOOP
 .WORD ACTRLP-10\$;REMOTE MODEM LOOP
 .WORD ACTNUF-10\$;MORE COMMAND LINE NEEDED
 .WORD ACTBCR-10\$;BAD CHARACTER IN OPERATOR MESSAGE
 .WORD ACTDMS-10\$;DUMP MEMORY START ADDRESS
 .WORD ACTDME-10\$;DUMP MEMORY END ADDRESS

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) 16-JUN-81 14:39 PAGE 13-87
ACTION TABLE AND ROUTINES

SEQ 0123

7732 036450 000734
7733 036452 000246
7734 036454 001626
7735 036456 000236
7736 036460 001326
7737

.WORD ACTDMQ-10\$;DUMP WORD
.WORD ACTPRT-10\$;PRINT
.WORD ACTMOS-10\$;MODEM ACTION REV B BY EC
.WORD ACTEXT-10\$;EXIT ACTION REV B BY EC
.WORD ACTSEX-10\$;SET E=T ACTION REV B BY EC NPI

7739										
7740	036462	112737	177777	003560	ACTNUF:	MOVB	#-1,P\$NNUF			;SET FLAG TO SAY NEED MORE OF COMMAND
7741	036470	000207			ACTNUL:	RTS	PC			;RETURN TO PARSER
7742										
7743	036472	012737	000001	003202	ACTCLR:	MOV	#CLEAR,KEYWD1			;SET LOC TO SAY A CLEAR WAS TYPED
7744	036500	000207				RTS	PC			
7745										
7746	036502	012737	000002	003202	ACTSHO:	MOV	#SHOW,KEYWD1			;SET LOC. TO SAY A SHOW WAS TYPED
7747	036510	000207				RTS	PC			
7748										
7749	036512	012702	003210		ACTHLP:	MOV	#HLPTAB,R2			;SETUP R2 AS A POINTER TO HELP MSG TABLE
7750	036516				1\$:	PRINTF	#HLPF,(R2)+			;PRINT HELP INFORMATION MESSAGES
(8)	036516	012246								MOV (R2)+,-(SP)
(7)	036520	012746	013225							MOV #HLPF,-(SP)
(6)	036524	012746	000002							MOV #2,-(SP)
(3)	036530	010600								MOV SP,R0
(4)	036532	104417								TRAP C\$PNTF
(4)	036534	062706	000006							ADD #6,SP
7751	036540	020227	003230			CMP	R2,#HLPEND			;SEE IF ALL INFO PRINTED YET
7752	036544	001364				BNE	1\$;IF NO KEEP PRINTING
7753	036546	012737	000005	003202		MOV	#HLP,KEYWD1			;SET LOC. TO SAY A HELP WAS TYPED
7754	036554	000207				RTS	PC			
7755	036556	012737	000057	003202	ACTEXT:	MOV	#EXIT,KEYWD1			;EXIT COMMAND
7756	036564	000207				RTS	PC			
7757	036566	012737	000055	003202	ACTPRT:	MOV	#PRNT,KEYWD1			;SET LOC. TO SAY A HELP WAS TYPED
7758	036574	004737	027046			JSR	PC,REPORT			;CALL ROUTINE TO PRINT EVENT LOG AND BASE TABLE
7759	036600	000207				RTS	PC			
7760										
7761	036602	012737	000004	003202	ACTRUN:	MOV	#RUN,KEYWD1			;SET RUN FLAG
7762	036610	112737	177777	003560		MOVB	#-1,P\$NNUF			;SET FLAG TO SAY NEED MORE OF COMMAND
7763	036616	012737	000001	007500		MOV	#1,RPASS			;SET DEFAULT RUN 'PASS' TO 1
7764	036624	000207				RTS	PC			
7765										
7766	036626	012701	006562		ACTCSE:	MOV	#PTRTAB,R1			
7767	036632	012702	000017			MOV	#MSGLIM,R2			
7768	036636	006302				ASL	R2			
7769	036640	006302				ASL	R2			
7770	036642	010137	007336			MOV	R1,CMPPTR			
7771	036646	060237	007336			ADD	R2,CMPPTR			;INIT COMPARE MESSAGE POINTER
7772	036652	013701	007336			MOV	CMPPTR,R1			
7773										
7774	036656	013702	007340			MOV	CMPTOT,R2			
7775	036662	105037	003560			CLRB	P\$NNUF			;FLAG THAT HAVE VALID COMMAND AT THIS PT.
7776	036666	023727	003202	000002		CMP	KEYWD1,#SHOW			;SEE IF A CLEAR OR SHOW WAS TYPED
7777	036674	001500				BEQ	ACTSHW			;BR IF A SHOW WAS TYPED
7778	036676	012737	000001	007340		MOV	#1,CMPTOT			;CLEAR COMPARE MESSAGE COUNT, CHAR. COUNT
7779	036704	005037	007342			CLR	CTOTCC			; AND RESET POINTER
7780										
7781	036710	012701	006562			MOV	#PTRTAB,R1			
7782	036714	012702	000017			MOV	#MSGLIM,R2			
7783	036720	006302				ASL	R2			
7784	036722	006302				ASL	R2			
7785	036724	010137	007336			MOV	R1,CMPPTR			
7786	036730	060237	007336			ADD	R2,CMPPTR			;INIT COMPARE MESSAGE POINTER
7787	036734	013737	007336	007416		MOV	CMPPTR,CPTR			;SET UP TO FILL IN DEFAULT MESSAGE
7788	036742	012701	005562			MOV	#CMPBUF,R1			

```

7789 036746 010137 007344      MOV      R1,CCURAD
7790 036752 000431      BR       ACTCLB
7791
7792 036754 012701 006562      ACTCST: MOV      #PTRTAB,R1
7793 036760 013702 007354      MOV      TXMTOT,R2
7794 036764 105037 003560      CLRB    PSNUF      ;FLAG THAT HAVE VALID COMMAND AT THIS PT.
7795 036770 023727 003202 000002      CMP      KEYWD1,#SHOW ;SEE IF A CLEAR OR SHOW WAS TYPED
7796 036776 001437      BEQ     ACTSHW    ;BR IF A SHOW WAS TYPED
7797 037000 012737 000001 007354      MOV      #1,TXMTOT ;CLEAR TRANSMIT MESSAGE COUNT, CHAR. COUNT
7798 037006 005037 007356      CLR     TTOTCC    ; AND RESET POINTER
7799 037012 012737 006562 007334      MOV      #PTRTAB,TXPTR
7800 037020 013737 007334 007416      MOV      TXPTR,CPTR
7801 037026 012701 003562      MOV      #TXBUF,R1
7802 037032 010137 007360      MOV      R1,TCURAD
7803
7804 037036 012702 001000      ACTCLB: MOV      #BUFLIM,R2
7805 037042 010137 007420      MOV      R1,CURADD ;SET UP TO PUT DEFAULT MSG IN LIST AFTER 033'S
7806 037046 012737 000005 007410      MOV      #5,MSGTYP
7807 037054 013737 002162 007412      MOV      MSG5C,CURCC
7808 037062 105021      1$:     CLRB    (R1)+ ;FILL EXPT OR TRAN BUFFER WITH 0'S IF A CLEAR
7809 037064 005302      DEC     R2        ;DO 'BUFLIM' NUMBER OF BYTE LOCATIONS
7810 037066 001375      BNE    1$
7811 037070 004737 032010      JSR    PC,BLDBUF ;'CLEAR' REALLY MEANS TO PUT DEFAULT MSG IN
7812 037074 000207      RTS     PC        ;WHEN DONE, RETURN TO PARSER
7813
7814
7815 037076 012705 003504      ACTSHW: MOV      #SHTAB,R5
7816 037102 122571 000000      5$:     CMPB   (R5)+,@(R1) ;LOOK AT FIRST BYTE OF MSG TO DECIPHER TYPE
7817 037106 001404      BEQ    6$
7818 037110 020527 003513      CMP     R5,#SHTEND ;SEE IF LOOKED AT ALL OF DEFAULTS YET
7819 037114 001372      BNE    5$
7820 037116 005205      INC     R5        ;MUST BE OPR. SPEC'D THEN
7821 037120 162705 003505      6$:     SUB     #SHTAB+1,R5
7822 037124 006305      ASL    R5
7823 037126 016137 000002 007426      MOV     2(R1),TEMP
7824 037134      PRINTF #SHMSG,SHTYTB(R5),TEMP ;PRINT MSG SIZE & TYPE
(9) 037134 013746 007426      MOV     TEMP,-(SP)
(8) 037140 016546 003464      MOV     SHTYTB(R5),-(SP)
(7) 037144 012746 014430      MOV     #SHMSG,-(SP)
(6) 037150 012746 000003      MOV     #3,-(SP)
(3) 037154 010600      MOV     SP,R0
(4) 037156 104417      TRAP   C$PNTF
(4) 037160 062706 000010      ADD     #10,SP
7825 037164 062701 000004      ADD     #4,R1      ;BUMP R1 TO NEXT SET OF POINTERS
7826 037170 005302      DEC     R2
7827 037172 001341      BNE    ACTSHW
7828 037174 013737 007470 010650      MOV     MODTYP,DEV1
7829 037202 013737 007472 010652      MOV     MLTYP,DEV2
7830 037210 013737 007500 010654      MOV     RPASS,DEV3
7831 037216 013737 007476 010656      MOV     PARAM,DEV4
7832 037224 004737 032340      JSR    PC,SHWOP   ;SHOW THE OPERATOR THE CURRENT MODE..... ALSO
7833 037230 000207      RTS     PC
7834
7835 037232 013737 003554 007402      ACTDMS: MOV     PSNUM,STADD ;SETUP STARTING ADDRESS FOR DUMP
7836 037240 005037 007406      CLR     BYTBIT    ;SET DEFAULT OF WORD DUMP
7837 037244 012737 000052 003202      MOV     #DMPS,KEYWD1 ;FLAG THAT A DUMP WAS TYPED

```



```

7846
7847
7848 037276 012737 000010 003202 ACTSTE: MOV #SETEXP,KEYWD1
7849 037304 000403 BR ACTSTX
7850
7851 037306 012737 000011 003202 ACTSTT: MOV #SETTRN,KEYWD1
7852 037314 012737 000001 003206 ACTSTX: MOV #1,QUALVL ;SET UP DEFAULT COPY TO 1 (/COPY=0)
7853 037322 000207 RTS PC
7854
7855 037324 012737 000012 003204 ACTSIZE: MOV #SIZE,QUALFG
7856 037332 000207 RTS PC
7857
7858 037334 012737 000013 003204 ACTCOP: MOV #QCOPY,QUALFG
7859 037342 000207 RTS PC
7860
7861 037344 023727 003204 000012 ACTNUM: CMP QUALFG,#SIZE ;SEE IF A SIZE OR COPY TYPED
7862 037352 001023 BNE 1$ ;BR IF IT WAS A COPY
7863 037354 005737 003554 TST PSNUM ;CHECK TO BE SURE DIDN'T TRY SIZE=0
7864 037360 001014 BNE 3$ ; BR IF NO
7865 037362 PRINTF #CLISEO
(7) 037362 012746 013001 MOV #CLISEO,-(SP)
(6) 037366 012746 000001 MOV #1,-(SP)
(3) 037372 010600 MOV SP,R0
(4) 037374 104417 TRAP C$PNTF
(4) 037376 062706 000004 ADD #4,SP
7866 037402 112737 177777 003561 MOVB #-1,PSGDBD ;SEE ERROR-IN-CMD FLAG
7867 037410 000411 BR 2$
7868 037412 013737 003554 007412 3$: MOV PSNUM,CURCC ;IF A SIZE LOAD CURCC WITH BYTE COUNT
7869 037420 000405 BR 2$
7870 037422 013737 003554 003206 1$: MOV PSNUM,QUALVL ;IF A COPY, LOAD COPY COUNT
7871 037430 005237 003206 INC QUALVL ;INCREMENT SO FIRST DEC MAKES IT REAL #
7872 037434 000522 2$: BR ACTMEX
7873
7874 037436 012737 000007 007410 ACTOPM: MOV #7,MSGTYP
7875 037444 010437 007426 MOV R4,TEMP ;KEEP TRACK OF START OF QUOTED TEXT
7876 037450 005237 007426 INC TEMP ; SO CAN CALC OPCNT AT END OF QUOTES
7877 037454 000207 RTS PC
7878
7879 037456 010402 ACTEQO: MOV R4,R2
7880 037460 163702 007426 SUB TEMP,R2
7881 037464 010237 007412 MOV R2,CURCC ;CALC BYTE COUNT FOR QUOTED TEXT
7882 037470 010237 002166 MOV R2,OPCNT
7883 037474 013701 007426 MOV TEMP,R1
7884 037500 012705 002524 MOV #OPBUF,R5
7885 037504 112125 1$: MOVB (R1)+,(R5)+ ;COPY QUOTED TEXT TO OPBUF
7886 037506 005302 DEC R2
7887 037510 001375 BNE 1$
7888 037512 000473 BR ACTMEX
7889
7890 037514 ACTBCR: PRINTF #CLIBCR ;BAD CHAR. IN OPR. QUOTED STRING
(7) 037514 012746 012734 MOV #CLIBCR,-(SP)
(6) 037520 012746 000001 MOV #1,-(SP)
(3) 037524 010600 MOV SP,R0
(4) 037526 104417 TRAP C$PNTF
(4) 037530 062706 000004 ADD #4,SP
7891 037534 000207 RTS PC
  
```



```

7892 ;SET THE MESSAGE TYPE AS PER COMMAND LINE
7893 037536 005037 007410 ACTMS0: CLR MSGTYP
7894 037542 000435 BR ACTME1
7895 037544 012737 000001 007410 ACTMS1: MOV #1,MSGTYP ;ALL ONES
7896 037552 000431 BR ACTME1
7897 037554 012737 000002 007410 ACTMS2: MOV #2,MSGTYP ;ONES & ZEROS
7898 037562 000425 BR ACTME1
7899 037564 012737 000003 007410 ACTMS3: MOV #3,MSGTYP ;ZEROS & ONES
7900 037572 000421 BR ACTME1
7901 037574 012737 000004 007410 ACTMS4: MOV #4,MSGTYP ;CCITT
7902 037602 000415 BR ACTME1
7903 037604 012737 000005 007410 ACTMS5: MOV #5,MSGTYP ;QUICK FOX
7904 037612 013737 002162 007412 MOV MSG5C,CURCC ;SETUP DEFAULT SIZE FOR THIS TYPE
7905 037620 000430 BR ACTMEX
7906 037622 012737 000006 007410 ACTMS6: MOV #6,MSGTYP ;ALPHA/NUM
7907 037630 013737 002164 007412 MOV MSG6C,CURCC ;SETUP DEFAULT SIZE FOR THIS TYPE
7908
7909 037636 012737 000100 007412 ACTME1: MOV #64,CURCC ;SETUP DEFAULT SIZE FOR MSG0-4
7910 037644 000416 BR ACTMEX ;GO TO EXIT
7911
7912 ;REV B BY EC
7913 037646 022737 000010 003202 ACTSEX: CMP #SETEXP,KEYWD1 ;DID WE GET HERE FROM "SET E =" COMMAND?
7914 037654 001404 BEQ 10$ ;YES,BRANCH
7915 037656 112737 177777 003561 MOVB #-1,P$GDBD ;SET ERROR FLAG
7916 037664 000406 BR ACTMEX ;GO TO EXIT
7917 037666 004737 032134 10$: JSR PC,FACSIMILE ;GO COPY TRANMIT BUFFER TO EXPECT BUFFER
7918 037672 012737 000060 003202 MOV #SETET,KEYWD1 ;SET FLAG TO BE USED IN T1::
7919 037700 000400 BR ACTMEX ;GO TO EXIT
7920
7921 037702 105037 003560 ACTMEX: CLRB P$NNUF ;CLEAR NOT-ENOUGH FLAG
7922 037706 000207 RTS PC
7923
  
```


7953	040024	012737	000036	003204	ACTNO:	MOV	#NO,QUALFG		
7954	040032	000207				RTS	PC		
7955									
7956	040034	022737	000036	003204	ACTECH:	CMP	#NO,QUALFG		
7957	040042	001422				BEQ	1\$		
7958	040044	052737	000004	007476		BIS	#ECHOB,PARAM		
7959	040052	022737	000002	007470		CMP	#PAS,MODTYP		;BE SURE IN PASSIVE MODE IF
7960	040060	001416				BEQ	2\$;IF TRYING TO SET /ECHO
7961	040062					PRINTF	#CLINPS		
(7)	040062	012746	012671						MOV #CLINPS,-(SP)
(6)	040066	012746	000001						MOV #1,-(SP)
(3)	040072	010600							MOV SP,RO
(4)	040074	104417							TRAP C\$PNTF
(4)	040076	062706	000004						ADD #4,SP
7962	040102	112737	177777	003561		MOVB	#-1,\$GDBD		
7963	040110	042737	000004	007476	1\$:	BIC	#ECHOB,PARAM		
7964	040116	005037	003204		2\$:	CLR	QUALFG		;CLEAR 'NO' OUT OF QUALIFIER FLAG
7965	040122	000501				BR	ACTLXX		
7966									
7967	040124	012701	000002		ACTCHK:	MOV	#DATCKB,R1		;SET DATA CHECK BIT
7968	040130	000413				BR	ACTQFG		
7969									
7970	040132	012701	000001		ACTSTS:	MOV	#STATB,R1		;SET THE STATUS BIT
7971	040136	000410				BR	ACTQFG		
7972									
7973	040140	012701	000020		ACTCRC:	MOV	#CRCB,R1		;SET THE CRC BIT
7974	040144	000405				BR	ACTQFG		
7975									
7976	040146	012701	000010		ACTMOS:	MOV	#MOCHK,R1		;MODEM BIT ADDED BY EC
7977	040152	000402				BR	ACTQFG		
7978									
7979	040154	012701	000040		ACTPRO:	MOV	#PROTOB,R1		;SET THE PROTOCOL BIT
7980									
7981	040160	050137	007476		ACTQFG:	BIS	R1,PARAM		
7982	040164	022737	000036	003204		CMP	#NO,QUALFG		
7983	040172	001002				BNE	1\$		
7984	040174	040137	007476			BIC	R1,PARAM		
7985	040200	005037	003204		1\$:	CLR	QUALFG		;CLEAR 'NO' OUT OF QUALIFIER FLAG
7986	040204	000450				BR	ACTLXX		
7987									
7988	040206	013737	003554	007500	ACTRPS:	MOV	\$NUM,RPASS		;GET NUMBER OF 'RUN PASSES'
7989	040214	000444				BR	ACTLXX		
7990									
7991	040216	012737	000005	007472	ACTMOP:	MOV	#5,MLTYP		
7992	040224	000417				BR	ACTLPX		
7993	040226	012737	000001	007472	ACTTLP:	MOV	#1,MLTYP		
7994	040234	000413				BR	ACTLPX		
7995	040236	012737	000002	007472	ACTCLP:	MOV	#2,MLTYP		
7996	040244	000407				BR	ACTLPX		
7997	040246	012737	000003	007472	ACTLLP:	MOV	#3,MLTYP		
7998	040254	000403				BR	ACTLPX		
7999	040256	012737	000004	007472	ACTRLP:	MOV	#4,MLTYP		
8000									
8001	040264	022737	000003	007470	ACTLPX:	CMP	#ACT,MODTYP		;BE SURE IN ACTIVE IF TRYING TO SET LOOP
8002	040272	001415				BEQ	ACTLXX		; BR IF IN ACTIVE
8003	040274	112737	177777	003561		MOVB	#-1,\$GDBD		

8004 040302 005037 007472
8005 040306
(7) 040306 012746 012627
(6) 040312 012746 000001
(3) 040316 010600
(4) 040320 104417
(4) 040322 062706 000004
8006 040326 105037 003560
8007 040332 000207
8008

CLR MLTYP
PRINTF #CLIBDL

;CLEAR ANY LOOP TYPE THAT MAY HAVE GOT SET

MOV #CLIBDL,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTF
ADD #4,SP

ACTLXX: CLR B PSNNUF
RTS PC

;CLEAR NOT-ENOUGH FLAG

8058
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8080

```
.SBTTL          RECEIVE MODE SECTION
:++
: FUNCTIONAL DESCRIPTION:
: RECEIVE-ONLY (OR ONE-WAY-IN) ROUTINE
: IN THIS MODE OF TESTING THE DEVICE'S RECEIVER IS ENABLED IN EXPECTATION
: OF RECEIVING A MESSAGE. AFTER RECEIVING AN 'EXPECTED' NUMBER OF
: MESSAGES, THE DATA RECEIVED CAN BE COMPARED AGAINST A LIST OF 'EXPECT
: TO RECEIVE' MESSAGES IF DATA-CHECKING IS ENABLED.
:
: SUBORDINATE ROUTINES USED:
:   'ALLTR'
:
: CALLING SEQUENCE:
:   JMP      @MODE(R2)      ;DISPATCH TO MODE BASED ON MODE TYPE IN R2
:--
RXONLY:
RXON2:  MOV     RXPTR,CPTRR
        MOV     RXMTOT,DVRCT ;SET UP MESSAGE COUNT
        BIS     #QRX+#ERX,FLAG ;SET UP RX QUE
        CLR     CPTR      ;CLEAR THE TX POINTER
        JMP     ALLTR     ;GO RX.
```

```
040572
040572 013737 007332 007414
040600 013737 007370 007366
040606 052737 000104 007502
040614 005037 007416
040620 000137 040762
```

```
8082 .SBTTL TRANSMIT MODE SECTION
8083
8084 :++
8085 : FUNCTIONAL DESCRIPTION:
8086 : TRANSMIT-ONLY (OR ONE-WAY-OUT) ROUTINE
8087 : IN THIS MODE OF TESTING A LIST OF MESSAGES IS TRANSMITTED WITHOUT
8088 : EXPECTING ANY DATA TO BE RECEIVED. A REPETITION COUNT CAN BE
8089 : SPECIFIED TO REPETITIVELY TRANSMIT THE LIST.
8090
8091 : SUBORDINATE ROUTINES USED:
8092 : 'ALLTR'
8093
8094 : CALLING SEQUENCE:
8095 : JMP @MODE(R2) ;DISPATCH TO MODE BASED ON MODE TYPE IN R2
8096 :--
8097
8098 040624 042737 000002 007476 TXONLY: BIC #DATCKB,PARAM ;SET NOCHECK
8099 040632 013737 007334 007416 TXON2: MOV TXPTR,CPTR
8100 040640 013737 007354 007352 MOV TXMTOT,DVTCT ;COPY COUNTER FOR THIS PASS
8101 040646 052737 000210 007502 BIS #QTX+#ETX,FLAG ;SET THE QUE TX FLAG
8102 040654 005037 007414 CLR CPTRR ;CLEAR RX POINTER
8103 040660 000137 040762 JMP ALLTR ;GO TX.
```

8105
8106
8107
8108
8109
8110
8111
8112
8113
8114
8115
8116
8117
8118
8119
8120
8121
8122
8123
8124
8125
8126
8127
8128
8129

.SBTTL PASSIVE MODE SECTION

..++
: FUNCTIONAL DESCRIPTION:
: PASSIVE MODE SECTION
: IN THIS MODE OF TESTING, THE DEVICE'S RECEIVER IS ENABLED IN
: EXPECTATION OF RECEIVING A MESSAGE. THEN EVERY TIME A MESSAGE IS
: RECEIVED, A MESSAGE IS TRANSMITTED. DATA CHECKING CAN BE DONE ON THE
: RECEIVED DATA.

..SUBORDINATE ROUTINES USED:

.. 'ALLTR'

.. CALLING SEQUENCE:

.. JMP @MODE(R2) ;DISPATCH TO MODE BASED ON MODE TYPE IN R2

..--
PLCK:

PLCK2: MOV TXMTOT,DVTCT ;SET UP THE TRANSMIT COUNT
MOV TXPTR,CPTR ;SET UP CPTR TO TRANSMIT POINTER
PLCK3: MOV RXPTR,CPTRR ;SET UP CPTRR TO REC POINTER
BIS #QRX+#ERX,FLAG ;SET UP Q AND EXPECT RX
JMP ALLTR ;AND GO RX FIRST MSG.

040664
040664 013737 007354 007352
040672 013737 007334 007416
040700 013737 007332 007414
040706 052737 000104 007502
040714 000137 040762

8131
8132
8133
8134
8135
8136
8137
8138
8139
8140
8141
8142
8143
8144
8145
8146
8147
8148
8149
8150
8151
8152
8153
8154
8155
8156
8157
8158

.SBTTL ACTIVE MODE SECTION

;++
: FUNCTIONAL DESCRIPTION:
: ACTIVE MODE SECTION
: IN THIS MODE OF TESTING A LIST OF MESSAGES IS TRANSMITTED AND
: MESSAGES ARE EXPECTED TO BE RECEIVED. RECEIVED DATA CAN BE COMPARED
: AGAINST 'EXPECTED' DATA IF DATA-CHECKING IS ENABLED.
: NOTE: IF BOTH ENDS OF THE LINK ARE IN ACTIVE MODE, THEN THE
: LINK MUST BE A FULL DUPLEX LINK!

: SUBORDINATE ROUTINES USED:

: 'ALLTR'

: CALLING SEQUENCE:

: JMP @MODE(R2) ;DISPATCH TO MODE BASED ON MODE TYPE IN R2

:--

8150	040720	013737	007354	007352	ALCK:	MOV	TXMTOT,DVTCT	
8151	040726	013737	007334	007416		MOV	TXPTR,CPTR	;SET UP TX COUNTS
8152	040734	013737	007370	007366		MOV	RXMTOT,DVRCT	;SET UP COUNTS
8153	040742	013737	007332	007414		MOV	RXPTR,CPTRR	
8154	040750	052737	000314	007502		BIS	#QRX+#QTX+#ETX+#ERX,FLAG	
8155	040756	000137	040762			JMP	ALLTR	

8160
8161
8162
8163
8164
8165
8166
8167
8168
8169
8170
8171
8172
8173
8174
8175
8176
8177
8178
8179
8180
8181
8182
8183
8184
8185
8186
8187
8188
8189
8190
8191
8192
8193
8194
8195
8196
8197
8198
8199
8200
8201
8202
8203
8204
8205
8206
8207
8208
8209
8210
8211
8212
8213
8214
8215

.SBTTL TRANSMIT - RECEIVE FOR ALL STANDARD MODES

..**
: FUNCTIONAL DESCRIPTION:
: THIS CODE PERFORMS THE FOLLOWING FUNCTIONS
: 1.) IF RX BUFFERS ARE TO BE QUED, TELL DEVICE
: CODE TO QUE THEM, LOG RECEIVE QUED.
: 2.) IF TX BUFFERS ARE TO BE QUED, TELL DEVICE
: CODE TO QUE THEM, LOG TRANSMIT QUED.
: 3.) WAIT FOR EITHER RECEIVE BUFFER OR TRANSMIT BUFFER OR
: BOTH TO COMPLETE
: 4.) IF RECEIVE COMPLETE, LOG IT UPDATE RX TABLE IF DATA
: CHECKING.
: 5.) IF TRANSMIT COMPLETE LOG IT.
: 6.) WHEN BOTH TRANSMIT AND RECEIVE LISTS ARE DONE
: GO TO THE COMPARE BUFFER CODE

: SUBORDINATE ROUTINES USED:
: 'DVRXQ' -QUE RECEIVE BUFFER SPACE TO DEVICE
: 'LOGRXQ' -LOG RECEIVE BUFFER SPACE TO EVENT LOG
: 'LOGTXQ' -LOG TRANSMIT BUFFER QUED TO EVENT LOG
: 'DVTXRX' -QUE TRANSMIT BUFFER AND WAIT FOR RX
: OR TX TO COMPLETE
: 'LOGRXC' -LOG RECEIVE BUFFER COMPLETED TO EVENT LOG
: 'LOGTXC' -LOG TRANSMIT BUFFER COMPLETED TO EVENT LOG

: USE OF FLAG BITS:
: QRX - SET ON INPUT TO ALLTR IF REC IS TO BE QUED TO
: DEVICE. CLEARED BY DVRXQ AND THEN SET BY DVTXRX
: WHEN RX BUFFER IS COMPLETED.
: QTX - SET ON INPUT TO ALLTR IF TRANSMIT IS TO BE QUED TO
: DEVICE. CLEARED ON ENTRY TO DVTXRX AND SET BY DVTXRX
: WHEN TX BUFFER IS COMPLETED.
: ETX - USED BY DVTXRX TO DETERMINE IF TX BUFFER COMPLETED IS
: EXPECTED.
: ERX - USED BY DVTXRX TO DETERMINE IF RX BUFFER COMPLETED IS
: EXPECTED.

: CALLING SEQUENCE:
: JMP ALLTR ;GO TO TRANSMIT-RECEIVE FOR ALL STANDARD MODES
:--

8204	040762				ALLTR:				
8205	040762	032737	000004	007502	ALCK5:	BIT	#QRX,FLAG		
8206	040770	001420				BEQ	ALCK1		;IF NOT RX GO TO TX'S
8207	040772	013702	007414			MOV	CPTRR,R2		
8208	040776	011237	007432			MOV	(R2),TEMP2		
8209	041002	012237	007362			MOV	(R2)+,DVRXA		
8210	041006	011237	007434			MOV	(R2),TEMP3		
8211	041012	011237	007364			MOV	(R2),DVRCC		
8212	041016	010237	007414			MOV	R2,CPTRR		
8213	041022	004737	044040			JSR	PC,DVRXQ		;GO QUE DEVICE
8214	041026	004737	026406			JSR	PC,LOGRXQ		;LOG REC QUED
8215	041032	032737	000010	007502	ALCK1:	BIT	#QTX,FLAG		

8216	041040	001416			BEQ	ALCK2		;IF NO TX'S GO TO 2
8217	041042	013702	007416		MOV	CPTR,R2		
8218	041046	011237	007432		MOV	(R2),TEMP2		
8219	041052	012237	007346		MOV	(R2)+,DVTXA		
8220	041056	011237	007434		MOV	(R2),TEMP3		
8221	041062	012237	007350		MOV	(R2)+,DVTCC		
8222	041066	010237	007416		MOV	R2,CPTR		
8223	041072	004737	026352		JSR	PC,LOGTXQ		
8224								
8225	041076	004737	044120		ALCK2:	JSR	PC,DVTXRX	;GO TO TX AND RX SUB ROUT.
8226								
8227	041102	032737	000004	007502	BIT	#QRX,FLAG		;CHECK FOR REC. MSG.
8228	041110	001514			BEQ	ALCK3		
8229	041112	013737	007362	007432	MOV	DVRXA,TEMP2		
8230	041120	013737	007364	007434	MOV	DVRCC,TEMP3		
8231	041126	004737	026424		JSR	PC,LOGRXC		;LOG REC COMPLETE
8232	041132	032737	000004	007476	UPTABL:	BIT	#ECHOB,PARAM	;IS THIS ECHO MODE(PASSIVE)
8233	041140	001406			BEQ	UPTA4		;IF NOT GO TO 4
8234	041142	013702	007416		MOV	CPTR,R2		;ELSE SET R2 TO PRESENT TX TABL
8235	041146	013722	007432		MOV	TEMP2,(R2)+		;STORE OFF RX ADD
8236	041152	013712	007434		MOV	TEMP3,(R2)		;AND CC
8237	041156	032737	000002	007476	UPTA4:	BIT	#DATCKB,PARAM	;IS DATA CHECKING ASKED FOR
8238	041164	001015			BNE	UPTA1		;IF SO GO TO 1
8239	041166	012737	000001	007366	MOV	#01,DVRCT		;ELSE SET DVRCT TO A 1
8240	041174	013737	007332	007414	MOV	RXPTR,CPTRR		;RESET POINTER
8241	041202	022737	000003	007470	CMP	#ACT,MODTYP		;IS THIS ACTIVE
8242	041210	001002			BNE	UPTA3		
8243	041212	005237	007366		INC	DVRCT		;IF YES BUMP COUNT
8244	041216	000424			UPTA3:	BR	UPTEX	
8245	041220	013702	007414		UPTA1:	MOV	CPTRR,R2	
8246	041224	011237	007426		MOV	(R2),TEMP		;LOAD TEMP WITH PREV. COUNT
8247	041230	163737	007434	007426	SUB	TEMP3,TEMP		;LOAD TEMP WITH PREV.COUNT-CURRENT
8248	041236	013722	007434		MOV	TEMP3,(R2)+		
8249	041242	063737	007434	007432	ADD	TEMP3,TEMP2		
8250	041250	013722	007432		MOV	TEMP2,(R2)+		;STORE OF NEW ADD
8251	041254	013712	007426		MOV	TEMP,(R2)		;AND NEW CC
8252	041260	162702	000002		SUB	#2,R2		;PUT POINTER BACK TO ADDR.
8253	041264	010237	007414		MOV	R2,CPTRR		;AND RESTORE IT.
8254	041270				UPTEX:			
8255	041270	022737	000002	007470	CMP	#PAS,MODTYP		
8256	041276	001007			BNE	ALCK2A		;IF NOT PASSIVE LOOP THEN GO TO 2A
8257	041300	042737	000104	007502	BIC	#QRX+#ERX,FLAG		;CLEAR BOTH EXPECTED AND COMPLETED FLAGS
8258	041306	052737	000210	007502	BIS	#QTX+#ETX,FLAG		;SET THE TX FLAGS
8259	041314	000646			BR	ALCK1		
8260								
8261	041316	005337	007366		ALCK2A:	DEC	DVRCT	;DEC REC COUNT
8262	041322	005737	007366		TST	DVRCT		;IS IT ALL DONE
8263	041326	001005			BNE	ALCK3		;NO. GO CHECK TX
8264	041330	042737	000004	007502	BIC	#QRX,FLAG		;CLEAR THE RX FLAG
8265	041336	005037	007414		CLR	CPTRR		;YES. CLEAR POINTER
8266	041342	032737	000010	007502	ALCK3:	BIT	#QTX,FLAG	;IS IT TX
8267	041350	001447			BEQ	ALCK4		;IF NOT TX THEN GO BACK
8268	041352	013737	007346	007432	MOV	DVTXA,TEMP2		
8269	041360	013737	007350	007434	MOV	DVTCC,TEMP3		;LOG TX COMPLETED
8270	041366	004737	026370		JSR	PC,LOGTXC		
8271	041372	005337	007352		DEC	DVTCT		;DEC TX COUNT

8272	041376	022737	000002	007470		CMP	#PAS,MODTYP	
8273	041404	001013				BNE	ALCK3A	;IF NOT PASSIVE MODE GO TO 3A
8274	041406	042737	000210	007502		BIC	#QTX+ETX,FLAG	;CLEAR THE TX FLAGS
8275	041414	052737	000104	007502		BIS	#QRX+ERX,FLAG	;AND SET THE RX FLAGS
8276	041422	005737	007352			TST	DVTCT	
8277	041426	001005				BNE	ALCK3C	;IF MORE RX'S DO IT
8278	041430	000137	041510			JMP	CMPSR	; ELSE COMPARE
8279	041434	005737	007352		ALCK3A:	TST	DVTCT	;IS IT ALL DONE
8280	041440	001402				BEQ	ALCK3B	;IF NOT GO BACK TO 5
8281	041442	000137	040762		ALCK3C:	JMP	ALCK5	
8282	041446	005037	007416		ALCK3B:	CLR	CPTR	;IF SO CLEAR POINTER
8283	041452	042737	000010	007502		BIC	#QTX,FLAG	;CLEAR TX FLAG
8284	041460	032737	000002	007476		BIT	#DATCKB,PARAM	;IS IT DAT CK
8285	041466	001403				BEQ	ALCK4A	;IF NOT THEN END WO CKING RX.
8286	041470	005737	007414		ALCK4:	TST	CPTRR	
8287								
8288	041474	001362				BNE	ALCK3C	;IF SOME RX'S LEFT GO BACK
8289	041476	005737	007416		ALCK4A:	TST	CPTR	
8290	041502	001402				BEQ	ALCK4B	;BRANCH IF ANY TX'S LEFT
8291	041504	000137	041076			JMP	ALCK2	
8292	041510				ALCK4B:			
8293								
8294								
8295								

8297
 8298
 8299
 8300
 8301
 8302
 8303
 8304
 8305
 8306
 8307
 8308
 8309
 8310
 8311
 8312
 8313
 8314
 8315
 8316
 8317
 8318
 8319
 8320
 8321
 8322
 8323
 8324
 8325
 8326
 8327
 8328
 8329 041510 032737 000002 007476
 8330 041516 001522
 8331 041520 013737 007332 007416
 8332 041526 013737 007336 007414
 8333 041534 013737 007370 007366
 8334
 8335 041542
 8336 041542 013702 007416
 8337 041546 011237 007432
 8338 041552 012201
 8339 041554 012237 007434
 8340 041560 010237 007416
 8341
 8342 041564 013702 007414
 8343 041570 012203
 8344 041572 012204
 8345 041574 010237 007414
 8346 041600 010437 007436
 8347 041604 004737 026520
 8348
 8349 041610 020437 007434
 8350 041614 001410
 8351 041616 005237 007400
 8352 041622

.SBTTL DATA COMPARISON CODE

++
 : FUNCTIONAL DESCRIPTION:

CMPSR - COMPARE CODE
 THIS CODE COMPARES THE RECEIVED DATA AGAINST THE
 EXPECTED AND FILLS THE EVENT LOG WITH 1 OF 3 MSGS.

NOTE: IF NO DATA CHECKING SKIP THIS CODE

- 1.) A DATA COMPARISON ENTRY WHICH REPORTS THE NUMBER OF COMPARISON ERRORS FOUND.
 - 2.) A DATA COMPARISON ENTRY WHICH REPORTS DIFFERENCES IN REC LENGTH TO COMPARE LENGTH.
 - 3.) A DATA COMPARISON STARTED ENTRY WHICH REPORTS ADDRESS OF RECEIVE BUFFER AND BYTE COUNT.
- THIS CODE ALSO REPORTS SOFT ERRORS FOR DATA COMPARISON (THE FIRST 5 ONLY),LENGTH ERROR,AND TOTAL NUMBER OF ERRORS

SUBORDINATE ROUTINES USED:

'LOGCMP' - SEE ITEM 3 ABOVE
 'LOGCML' - SEE ITEM 2 ABOVE
 'LOGCMD' - SEE ITEM 1 ABOVE

CALLING SEQUENCE:

JMP CMPSR ;JUMP TO DATA COMPARISON CODE

--

CMPSR: BIT #DATCKB,PARAM ;IS DATA CHECKING TO BE DONE
 BEQ CMPSEX ;IF NOT THEN EXIT
 MOV RXPTR,CPTR ;PUT START OF RX POINTERS TO CPTR
 MOV CMPPTR,CPTRR ; AND START OF COMPARE POINTS TO CPTRR
 MOV RXMTOT,DVRC

CMPS3:
 MOV CPTR,R2 ;MOVE CURRET RX PT.TO R2
 MOV (R2),TEMP2 ;MOVE RX ADD TO EVENT LOG
 MOV (R2)+,R1 ;SET R1 TO START ADD OF RX
 MOV (R2)+,TEMP3 ;SET CHAR COUNT TO EVENT LOG
 MOV R2,CPTR ;RESTORE RX POINT

MOV CPTRR,R2 ;PUT R2 AT COMPARE TABLE
 MOV (R2)+,R3 ;SET R3 TO COMPARE ADD
 MOV (R2)+,R4 ;SET R4 TO COMP CC
 MOV R2,CPTRR ;RESTORE POINTER
 MOV R4,TEMP4
 JSR PC,LOGCMP ;LOG COMPARE START.

CMP R4,TEMP3 ;IS COMPARE COUNT = TO RX COUNT
 BEQ CMPS7 ;IF SO GO TO 7
 INC ERRCNT
 ERRSOFT 1,EDDLE,ERR10 ;PRINT ERROR

```

(4) 041622 104457
(5) 041624 000001
(5) 041626 016046
(5) 041630 025720
8353 041632 004737 026536 JSR PC,LOGCML ;LOG LENGTH ERROR
8354
8355 041636 005037 007436 CMPS7: CLR TEMP4 ;CLEAR BAD BYTE COUNTER
8356 041642 012737 000001 007424 MOV #1,OFSET ;SET OFSET BYTE COUNT TO 1
8357 041650 122123 CMPS1: CMPB (R1)+,(R3)+ ;COMPARE RX WITH EXPETED
8358 041652 001422 BEQ CMPS6 ;IF EQUAL THEN GO TO 6
8359
8360 041654 005237 007436 CMPS2: INC TEMP4 ;INC BAD COUNT
8361 041660 023727 007436 000005 CMP TEMP4,#5 ;IS IT MORE THEN 5
8362 041666 101014 BHI CMPS6 ;IF SO GO FOR MORE
8363 041670 114337 007446 MOVB -(R3),GOOD ;STORE GOOD BYTE FOR ERROR
8364 041674 114137 007447 MOVB -(R1),BAD ;STORE BAD BYTE FOR ERROR
8365 041700 005237 007400 INC ERRCNT
8366 041704 ERRSOFT 2,EDDDE,ERR1 ;REPORT COMPARISON FAILURE TO OPR.
(4) 041704 104457 TRAP CSERSOFT
(5) 041706 000002 .WORD 2
(5) 041710 016103 .WORD EDDDE
(5) 041712 025630 .WORD ERR1
8367 041714 005201 INC R1
8368 041716 005203 INC R3
8369 041720 005237 007424 CMPS6: INC OFSET ;INC OFFSET
8370 041724 005304 DEC R4 ;ELSE DEC CHAR COUNT AND SEE IF 0
8371 041726 001350 BNE CMPS1 ;IF NOT GO BACK
8372 041730 005737 007436 TST TEMP4 ;SEE IF ANY CMP ERRS FOR THIS MSG
8373 041734 001410 BEQ CMPS5A ;BR IF NONE
8374 041736 005237 007400 INC ERRCNT
8375 041742 ERRSOFT 3,EDDDE,ERR2 ;REPORT # OF MISMATCHES FOR MESSAGE
(4) 041742 104457 TRAP CSERSOFT
(5) 041744 000003 .WORD 3
(5) 041746 016103 .WORD EDDDE
(5) 041750 025672 .WORD ERR2
8376 041752 004737 026554 CMPS5: JSR PC,LOGCMD ;LOG DATA ERROR IN COMPARE
8377 041756 CMPS5A:
8378 041756 005337 007366 DEC DVRCT
8379 041762 001267 BNE CMPS3 ;IF NOT ALL DONE GO BACK
8380
  
```

8382
8383
8384
8385
8386
8387
8388
8389
8390
8391
8392
8393
8394
8395
8396
8397
8398
8399
8400
8401
8402
8403
8404
8405
8406
8407
8408
8409
8410
8411
8412

.SBTTL INTERNAL END OF PASS CODE

++
: FUNCTIONAL DESCRIPTION:
: THIS CODE INCREMENTS THE PASS COUNT FOR THE
: EVENT LOG. LOGS THE END OF PASS EVENT
: IF 'RPASS' IS A MINUS ONE RETURN TO MODE
: DISPATCHER. IF NOT -1 THEN DECREMENT RPASS
: AND IF 'RPASS' IS THEN = TO 0 GO TO DCLT PROMT
: IN NOT = TO 0 THEN GO BACK TO MODE DISPATCHER

: SUBORDINATE ROUTINES USED:
:-----
: 'LOGEOP' - LOG END OF PASS TO EVENT LOG

8399	041764	005237	007376		CMPSEX: INC	PSCNT	:BUMP PASS COUNT
8401	041770	013737	007374	007436	MOV	NOBUF,TEMP4	
8402	041776	013737	007376	007432	MOV	PSCNT,TEMP2	
8403	042004	013737	007400	007434	MOV	ERRCNT,TEMP3	
8404	042012	004737	026572		JSR	PC,LOGEOP	:LOG END OF PASS
8405	042016	022737	177777	007500	5\$: CMP	#-1,RPASS	:SEE IF RPASS=-1
8406	042024	001403			BEQ	1\$:IF IT IS DON'T DECRMNT, LOOP FOREVER
8407	042026	005337	007500		DEC	RPASS	:DEC PASS COUNT
8408	042032	001402			BEQ	2\$:IF DONE EXIT TEST
8409	042034	000137	040524	1\$:	JMP	GTRX2	:ELSE GO BACK AND DISPATCH
8410	042040	004737	045714	2\$:	JSR	PC,DVBTUP	:GO UPDATE BASE TABLE
8411	042044	000137	035224		JMP	GTRAS	:WHEN RPASS=0 GO BACK TO 'DCLT>'

8414
8415
8416
8417
8418
8419
8420
8421
8422
8423
8424
8425
8426
8427
8428
8429
8430
8431
8432
8433
8434
8435
8436
8437
8438
8439
8440
8441
(3)
(3)
(4)
(5)
(5)
(5)
(5)
(5)
(5)
(3)
8442
8443
8444
8445
8446
8447
8448
8449
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8460

.SBTTL DOWN-LINE-LOAD SECTION

FUNCTIONAL DESCRIPTION:
DOWN-LINE-LOAD SECTION
IN THIS MODE OF TESTING THE 'HOST' OR ORIGINATING STATION
REQUESTS THE 'SATELLITE' OR BOOT STATION TO ENTER MOP MODE.
THE BOOT STATION THEN SENDS A 'REQUEST PROGRAM MESSAGE'.
THE 'HOST' THEN SENDS A 'MEMORY LOAD WITH TRANSFER ADDRESS'
THAT CONTAINS IMAGE DATA TO BE LOADED BY THE BOOT STATION'S
M9312 STARTING AT LOC. 0. THIS IMAGE DATA WILL CONTAIN A
PROGRAM THAT WILL PRINT A MSG THAT DOWN-LINE-LOAD WAS SUCCESSFUL.

SUBORDINATE ROUTINES USED:
'DLTXRX' - SPECIAL TX RX ROUTINE FOR DLL
'DVRXQ' - QUE RX BUFFER SPACE TO DEVICE
'LOGRXQ' - LOG RX SPACE QUED TO EVENT LOG
'LOGTXQ' - LOG TX BUFFER QUED TO EVENT LOG
'DVTXRX' - QUE TX BUFFER AND WAIT FOR RX OR TX TO COMPLETE
'LOGTXC' - LOG TX COMPLETED TO EVENT LOG
'LOGRXC' - LOG RX COMPLETED TO EVENT LOG

CALLING SEQUENCE:
JMP @MODE(R2) ;DISPATCH TO MODE BASED ON MODE TYPE IN R2

DLL: GMANID DLLQ1,TEMP3,0,377,0,377,NO ;GET PASSWORD
TRAP CSGMAN
BR 10001\$
.WORD TEMP3
.WORD TSCODE
.WORD DLLQ1
.WORD 377
.WORD TSLOLIM
.WORD TSHILIM

10001\$:
MOV TEMP3,PASS1 ;PUT PASSWORD IN MESSAGE
MOV TEMP3,PASS2 ;PASSWORD IS DUPLICATE
MOV TEMP3,PASS3 ;:HERE
MOV TEMP3,PASS4 ;:AND HERE.
BIS #ERX,FLAG ;SET EXPECTED TO RX
BIC #DATCKB,PARAM ;CLEAR NOCHECK
MOV #DLLM1,CURADD ;SET THE DOWN LINE LOAD MSG TO #1
MOV DLLM1C,CURCC ;SET THE CC
JSR PC,DLTXRX ;GO TO THE DOWN LINE TX RX ROUTINE

;RETURN WHEN TX AND RX ARE COMPLETED
MOV #DLLM2,CURADD ;SET THE DOWN LINE LOAD MSG TO #2
MOV DLLM2C,CURCC ;SET CC
BIC #DLLGA,FLAG ;CLEAR THE GO AHEAD FLAG
JSR PC,DLTXRX ;GO TO THE DOWN LINE TX RX ROUTINE

; RETURN WHEN TX AND RX ARE COMPLETED
DLLPRI:

8461	042202				PRINTF #DLLCM		
(7)	042202	012746	015030			MOV #DLLCM,-(SP)	
(6)	042206	012746	000001			MOV #1,-(SP)	
(3)	042212	010600				MOV SP,R0	
(4)	042214	104417				TRAP C\$PNTF	
(4)	042216	062706	000004			ADD #4,SP	
8462	042222	000137	035224		JMP GTRAS		
8463							
8464	042226				DLLEA: ERRHRD 20,DLLAB,ERR14		
8465	042226						
(4)	042226	104456				TRAP C\$ERHRD	
(5)	042230	000024				.WORD 20	
(5)	042232	025214				.WORD DLLAB	
(5)	042234	026142				.WORD ERR14	
8466							
8467	042236	000137	035224		JMP GTRAS ;PRINT ABORT AND EXIT		
8468							
8469							
8470							
8471	042242				DLTXRX:		
8472	042242	052737	000004	007502	BIS #QRX,FLAG ;SET THE QUE RX FLAG		
8473	042250	012737	004562	007362	MOV #RXBUF,DVRXA ;SET THE DEVICE RX BUFFER TO RXBUF		
8474	042256	012737	004562	007432	MOV #RXBUF,TEMP2 ;SET UP FOR LOG		
8475	042264	012737	000400	007364	MOV #256.,DVRCC ;SET UP FOR CC OF 256		
8476	042272	012737	000400	007434	MOV #256.,TEMP3 ;SET UP FOR LOG		
8477	042300	004737	044040		JSR PC,DVRXQ ; GO QUE RX		
8478	042304	004737	026406		JSR PC,LOGRXQ ;AND LOG IT...		
8479							
8480	042310	013737	007420	007346	MOV CURADD,DVTXA ;SET UP FOR TX		
8481	042316	013737	007420	007432	MOV CURADD,TEMP2 ;AND LOG		
8482	042324	013737	007412	007350	MOV CURCC,DVTCC ;SE UP FOR TX COUNT		
8483	042332	013737	007412	007434	MOV CURCC,TEMP3 ;AND LOG IT		
8484	042340	004737	026352		JSR PC,LOGTXQ ;LOG THE TX QUEUED		
8485	042344	052737	000210	007502	BIS #QTX+#ETX,FLAG ;SET UP TO QUE AND EXPECTED		
8486	042352	004737	044120		DLLE2: JSR PC,DVTXRX ;GO TO DEVICE ROUTINE		
8487	042356	032737	000400	007502	BIT #DLLGA,FLAG ;TEST FOR GO AHEAD BIT		
8488	042364	001047			BNE DLLE1 ;IF SET GO TO ONE		
8489	042366	032737	000010	007502	BIT #QTX,FLAG ;ELSE CHECK FOR TX DONE		
8490	042374	001020			BNE DLLE6 ;IF DONE THEN BRANCH		
8491							
8492	042376	012737	025521	007442	MOV #TXNC,CONOTM ;ELSE ERROR		
8493	042404	013737	004562	007434	DLLE7: MOV RXBUF,TEMP3		
8494	042412	013737	003562	007436	MOV TXBUF,TEMP4		
8495	042420	012737	025214	007432	MOV #DLLAB,TEMP2		
8496	042426	004737	026434		JSR PC,LGDVE ;LOG ERROR		
8497	042432	000137	042226		JMP DLLEA ;ABORT TEST		
8498							
8499	042436	013737	007346	007432	DLLE6: MOV DVTXA,TEMP2		
8500	042444	013737	007350	007434	MOV DVTCC,TEMP3		
8501	042452	004737	026370		JSR PC,LOGTXC ;LOG TX DONE		
8502	042456	042737	000210	007502	BIC #QTX+#ETX,FLAG ;CLEAR QUE AND EXPECTED		
8503	042464	052737	000400	007502	BIS #DLLGA,FLAG ;SET THE GO AHEAD BIT		
8504	042472	023737	002174	007350	CMP DLLM2C,DVTCC		
8505	042500	001472			BEQ DLLE5 ;EXIT IF SECOND MSG.		
8506	042502	000723			BR DLLE2 ;AND GO BACK TO 2		
8507	042504	032737	000004	007502	DLLE1: BIT #QRX,FLAG ;IS THE A RX COMPLETED		

8540
8541
8542
8543
8544
8545
8546
8547
8548
8549
8550
8551
8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562
8563
8564
8565
(3)
(3)
(4)
(5)
(5)
(5)
(5)
(5)
(5)
(3)
8566
8567
8568
8569
8570
8571
8572
8573
8574
8575
8576
8577
8578
8579
8580
8581
8582
8583
8584
8585
8586

```

.SBTTL          TALK MODE SECTION

:++
: FUNCTIONAL DESCRIPTION:
: TALK MODE SECTION
: IN THIS MODE, THE "TALK" END OF THE LINK TRANSMITS OPERATOR
: SPECIFIED MESSAGES UNTIL A "EXIT" MESSAGE IS TYPE. AT THAT POINT,
: THIS END OF THE LINK GOES INTO "LISTEN" MODE.

: SUBORDINATE ROUTINES USED:

:         'LOGTXQ' - LOG TX BUFFER QUED TO EVENT LOG
:         'DVTXRX' - QUE TX BUFFER TO DEVICE AND WAIT FOR COMPLETE
:         'LOGTXC' - LOG TX COMPLETE TO EVENT LOG

: CALLING SEQUENCE:
:         JMP      @MODE(R2)          ;DISPATCH TO MODE BASED ON MODE TYPE IN R2
:--

TALCK:
1$:  BIC      #DATCKB,PARAM      ;SET NOCHECK
      MOV     #OPBUF,R2
      MOV     #-1,(R2)+         ;CLEAR OUT OPBUFFER FIRST
      CMP     #OPEND,R2
      BNE    1$
      GMANID  OPRMM,OPBUF,A,0,1,72.,NO      ;GET TALK MESSAGE

TRAP    CS$GMAN
BR      10002$
.WORD   OPBUF
.WORD   T$CODE
.WORD   OPRMM
.WORD   0
.WORD   T$LOLIM
.WORD   T$HILIM

10002$:
2$:  CLR     R2                  ;NOW GET CHAR COUNT
      CMPB   #377,OPBUF(R2)
      BEQ   3$
      INC   R2
      BR    2$
3$:  MOV     R2,OPCNT

      MOV     #OPBUF,DVTXA      ;SET UP TX ADDR.
      MOV     #OPBUF,TEMP2
      MOV     OPCNT,TEMP3
      MOV     OPCNT,DVTCC      ;SET UP TX CC
      JSR    PC,LOGTXQ
      BIS     #QTX+#ETX,FLAG   ;SET UP FLAGS
      CLR    CPTRR             ;CLEAR RX POINTER

      JSR    PC,DVTXRX

      MOV     DVTXA,TEMP2
      MOV     DVTCC,TEMP3
      JSR    PC,LOGTXC
      CMP     #'EX,OPBUF      ;CHECK FOR EXIT
  
```

042670
042670 042737 000002 007476
042676 012702 002524
042702 012722 177777
042706 022702 002646
042712 001373
042714 104443
042716 000406
042720 002524
042722 000142
042724 014764
042726 000000
042730 000001
042732 000110
042734 005002
042736 122762 000377 002524
042744 001402
042746 005202
042750 000772
042752 010237 002166
042756 012737 002524 007346
042764 012737 002524 007432
042772 013737 002166 007434
043000 013737 002166 007350
043006 004737 026352
043012 052737 000210 007502
043020 005037 007414
043024 004737 044120
043030 013737 007346 007432
043036 013737 007350 007434
043044 004737 026370
043050 022737 054105 002524

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

E 12
MACY11 30A(1052) 16-JUN-81 14:39 PAGE 14-20
TALK MODE SECTION

SEQ 0147

8587	043056	001304		
8588	043060	022737	052111	002526
8589	043066	001300		
8590	043070	042737	000210	007502
8591	043076	012737	000006	007470
8592	043104	000137	040524	

BNE	TALCK	
CMP	#'IT,OPBUF+2	
BNE	TALCK	
BIC	#QTX+#ETX,FLAG	:CLEAR THE TX BITS
MOV	#LIS,MODTYP	:CHANGE TO LISTEN MODE
JMP	GTRX2	:AND GO BACK TO DISPATCH

8594
 8595
 8596
 8597
 8598
 8599
 8600
 8601
 8602
 8603
 8604
 8605
 8606
 8607
 8608
 8609
 8610
 8611
 8612
 8613
 8614
 8615
 (7)
 (6)
 (3)
 (4)
 (4)
 8616
 8617
 8618
 8619
 8620
 8621
 8622
 8623
 8624
 8625
 8626
 8627
 8628
 8629
 8630
 8631
 8632
 8633
 (7)
 (6)
 (3)
 (4)
 (4)
 8634
 8635
 8636
 8637
 8638
 8639

```

.SBTTL      LISTEN MODE SECTION

:++
: FUNCTIONAL DESCRIPTION:
: LISTEN MODE SECTION
: IN THIS MODE, THE 'LISTEN' END OF THE LINK PRINTS ALL OF THE MESSAGES
: RECEIVED BY THE DEVICE ON THE OPERATOR'S CONSOLE. IF THE MESSAGE
: RECEIVED IS AN 'EXIT' MESSAGE, THEN THE NODE ENTERS 'TALK' MODE.

: SUBORDINATE ROUTINES USED:
:
:   'DVRXQ' - QUE RECEIVE BUFFER SPACE TO DEVICE
:   'LOGRXQ' - LOG RECEIVE BUFFER QUED TO EVENT LOG
:   'DVTXRX' - WAIT FOR RX TO COMPLETE
:   'LOGRXC' - LOG RX COMPLETE TO EVENT LOG

: CALLING SEQUENCE:
:   JMP      @MODE(R2)      ;DISPATCH TO MODE BASED ON MODE TYPE IN R2
:--

8614 043110 042737 000002 007476 LISCK: BIC      #DATCKB,PARAM ;CLEAR CHECK BIT
8615 043116 012746 014753          PRINTF   #LISP      ;PRINT PROMPT FOR OPR.
      (7) 043116 012746 014753          MOV      #LISP,-(SP)
      (6) 043122 012746 000001          MOV      #1,-(SP)
      (3) 043126 010600          MOV      SP,R0
      (4) 043130 104417          TRAP    C$PNTF
      (4) 043132 062706 000004          ADD     #4,SP

8616 043136 012737 002524 007362 LISCKA: MOV     #OPBUF,DVRXA ;SET DEVICE UP TO REC AT OPBUF
8617 043144 012737 002524 007432      MOV     #OPBUF,TEMP2
8618 043152 012737 000122 007364      MOV     #82.,DVRCC ;SET UP CHAR COUNT TO 82.
8619 043160 012737 000122 007434      MOV     #82.,TEMP3
8620 043166 052737 000104 007502      BIS     #QRX+#ERX,FLAG ;SET UP FLAG
8621 043174 005037 007416      CLR     CPTR      ;CLEAR THE TX.

8623 043200 004737 044040          JSR     PC,DVRXQ ;QUE RX
8624 043204 004737 026406          JSR     PC,LOGRXQ

8626 043210 004737 044120          JSR     PC,DVTXRX ;GO TO DEVICE RX. SUBROUTINE

8628 043214 013737 007362 007432      MOV     DVRXA,TEMP2
8629 043222 013737 007364 007434      MOV     DVRCC,TEMP3 ;SET UP ADDR.AND CC.
8630 043230 004737 026424          JSR     PC,LOGRXC ;LOG COMPLETED
8631 043234 063737 007362 007364      ADD     DVRXA,DVRCC
8632 043242 105077 144116      CLRB   @DVRCC
8633 043246          PRINTF #OPBFPT
      (7) 043246 012746 002520          MOV     #OPBFPT,-(SP)
      (6) 043252 012746 000001          MOV     #1,-(SP)
      (3) 043256 010600          MOV     SP,R0
      (4) 043260 104417          TRAP    C$PNTF
      (4) 043262 062706 000004          ADD     #4,SP

8634 043266 022737 054105 002524      CMP     #'EX,OPBUF ;COMPARE FOR EX OF 'EXIT'
8635 043274 001320          BNE    LISCKA ;IF NOT EXIT THEN GO BACK
8636 043276 022737 052111 002526      CMP     #'IT,OPBUF+2 ;IF FIRST HALF OK CHECK NEXT PART
8637 043304 001314          BNE    LISCKA ;IF NOT EXIT THE GO BACK
8638 043306 012737 000005 007470      MOV     #TAL,MODTYP ;CHANGE MODE TO TALK
8639 043314 000137 040524          JMP     GTRX2 ;RETURN TO DISPATCHER
  
```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) ^{G 12} 16-JUN-81 14:39 PAGE 14-22
LISTEN MODE SECTION

SEQ 0149

8640
8641

```

8643      .SBTTL      DEVICE FUNCTION SUBROUTINES
8644
8659
8660
8661
8662      .SBTTL      DEVICE INIT SUBROUTINE
8663
8681
8682      :++
8683      : FUNCTIONAL DESCRIPTION:
8684      : DVINIT- DEVICE INIT ROUTINE
8685      : THIS ROUTINE IS DEVICE DEPENDENT CODE THAT INITIS
8686      : THE DEVICE BEING TESTED. (I.E. FULL/HALF DUPLEX BAUD RATE, MAINT MODE.)
8687
8688      : INPUTS:      'FHDPLX' INDICATES IF MODE IS FULL OR HALF DUPLEX. (1=FULL)
8689      :                ADDRESS POINTERS (SELO,...) ALREADY POINT TO DEVICE'S REG.S
8690
8691      : SUBORDINATE ROUTINES USED:
8692
8693      :                'LGDVE' - LOG DEVICE ERROR TO EVENT LOG
8694      :                'TOORIO' - TIME OUT OR INPUT INTERRUPT OR OUTPUT INTERRUPT
8695      :                'CLR'   - CLEAR RQI AND WAIT FOR RDI TO GO AWAY
8696
8697
8698      : CALLING SEQUENCE:
8699      :                JSR      PC,DVINIT
8700      :--
8701
8702      043320      DVINIT:
8703      :MASTER CLEAR DEVICE
8704
8705      043320      012737      000100      007542      MOV      #100,TIMER1      ;SET UP TIMER 1 FOR 100(OCTAL) TICKS
8706      043326      005077      147042      CLR      @SEL6
8707      043332      005077      147032      CLR      @SEL4
8708      043336      012777      040000      147014      MOV      #MCLR,@SELO      ;DO A MASTER CLEAR
8709
8710      043344      022737      000004      012406      CMP      #DMRC6,OPTYP      ;IS THIS A 8206
8711      043352      001003      BNE      DVIN6              ;IF NOT GO TO 6
8712      043354      112777      000200      147000      MOVVB   #200,@BSEL1        ;SET RUN FOR 8206
8713      043362      022737      000006      012406      DVIN6:  CMP      #DMR6,OPTYP      ;IS THIS AN 8206 DMR
8714      043370      001003      BNE      DVIN2              ;IF NOT GO TO 2
8715      043372      112777      000200      146762      MOVVB   #200,@BSEL1        ;SET RUN BIT FOR 8206
8716
8717      043400      005777      146754      DVIN2:  TST      @SELO          ;IS RUN BIT SET
8718      043404      100426      BMI      DVIN1              ;IF YES GO TO 1 ELSE...
8719      043406      BREAK
8720      (3) 043406      104422
8721      043410      005737      007542      TST      TIMER1            ;SEE IF TIME HAS EXPIRED
8722      043414      001371      BNE      DVIN2              ;IF NOT GO BACK AND CHECK
8723      043416      012737      024424      007432      ;AGAIN ELSE...PRINT ERROR
8724      043424      017737      146730      007434      MOV      #DVEM3,TEMP2
8725      043432      017737      146726      007436      MOV      @SELO,TEMP3
8726      043440      004737      026434      MOV      @SEL2,TEMP4      ;LOAD UP ERRM. AND REG OUTPUTS
8727      043444      005237      007400      JSR      PC,LGDVE          ;LOG TIME OUT WAITING FOR RUN
8728      043450      INC      ERRCNT
      ERRSOFT 11,DVEM3,ERR13

```

(4)	043450	104457								TRAP	C\$ERSOFT
(5)	043452	000013								.WORD	11
(5)	043454	024424								.WORD	DVEM3
(5)	043456	026110								.WORD	ERR13
8729											
8730	043460	000717									
8731											
8732	043462										
8733											
8734											
8735											
8736	043462	042737	000003	007502							
8737	043470	112777	000143	146662							
8738	043476	004737	044746								
8739	043502	012777	017370	146660							
8740											
8741	043510	012777	000000	146656							
8742	043516	023727	012406	000006							
8743	043524	002403									
8744	043526	012777	000522	146640							
8745	043534	052777	000100	146622							
8746	043542	042777	004000	146610							
8747	043550	022737	000001	007472							
8748	043556	001003									
8749	043560	052777	004000	146572							
8750	043566	004737	044640								
8751											
8752											
8753											
8754	043572	023727	012406	000006							
8755	043600	002437									
8756	043602	112777	000145	146550							
8757	043610	004737	044746								
8758	043614	042777	000014	146552							
8759	043622	022737	000004	007472							
8760	043630	001003									
8761	043632	052777	000004	146534							
8762	043640	022737	000003	007472							
8763	043646	001003									
8764	043650	052777	000010	146516							
8765	043656	004737	044640								
8766											
8767											
8768											
8769											
8770	043662	112777	000146	146470							
8771	043670	004737	044746								
8772	043674	004737	044640								
8773											
8774											
8775											
8776	043700	112777	000141	146452							
8777	043706	004737	044746								
8778	043712	005077	146456								
8779	043716	022737	000004	007470							
8780	043724	001004									

BR DVINIT ;GO BACK AND TRY MSTR CLR AGAIN IF ERROR

DVIN1:

; DO BASE IN COMMAND

BIC #3,FLAG ;CLEAR INPUT AND OUTPUT INT FLAGS
MOVB #143,@BSEL0 ;SET UP BASE IN INT EN
JSR PC,TOORIO ;GO WAIT FOR INTERRUPT OR TIME OUT
MOV #BASE,@SEL4

DVIN7:

MOV #0,@SEL6 ;SET UP SEL 6
CMP OPTYP,#6 ;IS THIS DMR MODE
BLT DVIN7 ;IF NOT GO TO 7
MOV #522,@SEL6 ;SET DMR MODE
BIS #IEO,@SEL2 ;SET IEO
BIC #LULoop,@SEL0 ;CLEAR LU LOOP
CMP #TTL,MLTYP ;IS TTL SELECTED
BNE DVIN3 ; IF NOT GO TO 3
BIS #LULoop,@SEL0 ;ELSE SET LU LOOP
JSR PC,CLRAW

DVIN3:

; DO WRITE MODEM IF DMR MODE

CMP OPTYP,#6 ;IS THIS DMR MODE
BLT DVIN8 ;IF NOT GO TO 8
MOVB #145,@BSEL0 ;SET UP WRITE MODEM
JSR PC,TOORIO ;GO TO WAIT FOR INT
BIC #BIT2+#BIT3,@SEL6 ;CLEAR BSEL6 AND 7
CMP #MODREM,MLTYP ;IS THIS REMOTE LOOP
BNE DVIN9 ;IF NOT GO TO 9
BIS #BIT2,@SEL6 ;SET THE BIT
CMP #MODLOC,MLTYP ;IS IT MODEM LOCAL
BNE DVIN10 ;IF NOT EXIT
BIS #BIT3,@SEL6 ;SET MODEM LOCAL
JSR PC,CLRAW ;CLEAR RDI AND WAIT

DVIN9:

DVIN10:

; ENABLE EXTENDED ERROR IF DMR MODE

MOVB #146,@BSEL0 ;SET UP FOR ENABLE
JSR PC,TOORIO
JSR PC,CLRAW ;CLEAR RDI AND WAIT

; DO CONTROL IN COMMAND

DVIN8:

MOVB #141,@BSEL0 ;SET UP CONTROL IN
JSR PC,TOORIO ;WAIT FOR INT OR TIME OUT
CLR @SEL6 ;CLEAR HALF/DUP
CMP #DOW,MODTYP ;IS THIS DOWN LINE LOAD?
BNE DVIN5 ; BR IF NOT

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) 16-JUN-81 14:39 PAGE 14-25
DEVICE INIT SUBROUTINE

SEQ 0152

8781	043726	052777	002400	146440		BIS	#MAINTB+HALFDB,@SEL6	;IF SO SET MAINT MODE BIT
8782	043734	000406				BR	DVIN4	; AND FORCE HALF DUPLEX
8783								
8784	043736	005737	007474		DVIN5:	TST	FHDPLX	;IS THIS A HALF/DUP
8785	043742	001003				BNE	DVIN4	;IF NOT GO TO 4
8786	043744	052777	002000	146422		BIS	#HALFDB,@SEL6	;ELSE SET HALF/DUP
8787								
8788	043752	017737	146416	007444	DVIN4:	MOV	@SEL6,CONTIN	;SET UP CONTROL IN FOR MODS
8789	043760	004737	044640			JSR	PC,CLRAW	;GO CLEAR RQI AND WAIT
8790								;FOR RDI TO GO AWAY.
8791	043764	023727	012406	000006		CMP	OPTYP,#6	;IS THIS DMR ?
8792	043772	002403				BLT	DVINEX	;NO,EXIT
8793	043774	052737	001000	007502		BIS	#DMRRUN,FLAG	;SET DMRRUN OUTPUT EXPECTED BIT
8794	044002	000207			DVINEX:	RTS	PC	;RETURN TO CALLER
8795								
8796								
8797								
8798								
8799								

8801
8802
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838
8839
8840
8841
8842

.SBTTL

DEVICE GET MODEM STATUS SUBROUTINE

..++
: FUNCTIONAL DESCRIPTION:
: 'DVMODS' GET MODEM STATUS
:
: IMPLICIT INPUTS:
: THE BIT POSITION AND AVAILABILITY OF THE MODEM SIGNALS CTS,DSR,...RI,,
: IN THE DEPENDENT PORTION OF THE GLOBAL EQUATES SECTION.
:
: OUTPUTS:
: CURRENT MODEM SIGNAL VALUES IN 'MODS'
:
: SUBORDINATE ROUTINES USED:
: 'TOORIO' - TIME OUT OR INPUT INTERRUPT OR OUTPUT INTERRUPT
: 'CLRAW' - CLEAR RQI AND WAIT FOR RDI TO CLEAR
:
: CALLING SEQUENCE:
: JSR PC,DVMODS
:--

8835	044004	112777	000141	146346	DVMODS: MOVB	#141,@BSEL0	:SET UP CONTORL IN
8836	044012	004737	044746		JSR	PC,TOORIO	:GO TIME OUT CHECK
8837	044016	017737	146346	010456	MOV	@SEL4,MODS	:SET UP MODEM STATUS
8838	044024	013777	007444	14 42	MOV	CONTIN,@SEL6	:SET UP OLD CONTORL IN
8839	044032	004737	044640		JSR	PC,CLRAW	
8840	044036	000207			RTS	PC	:RETURN TO CALLER

8844
 8859
 8860
 8861
 8862
 8863
 8864
 8865
 8866
 8867
 8868
 8869
 8870
 8871
 8872
 8873
 8874
 8875
 8876
 8877
 8878
 8879
 8880
 8881
 8882
 8883
 8884
 8885
 8886
 8887
 8888
 8889
 8890
 8891
 8892
 8893
 8894
 8895
 8896

```
.SBTTL          DEVICE QUEUE RECEIVE SPACE SUBROUTINE

:++
: FUNCTIONAL DESCRIPTION:
:   DVRXQ - THIS SUB ROUTINE QUES THE REC BUFFER SPACE TO THE
:           DEVICE, THEN CLEARS THE QRX BIT OF THE FLAG WORD.

: INPUTS:
:   DVRXA = ADDRESS OF RX BUFFER SPACE
:   DVRCC = BYTE CHAR COUNT OF RX BUFFER
:   QRX FLAG BIT = SET BY CALLING ROUTINE

: OUTPUTS:
:   QRX FLAG BIT = CLEARED BY ROUTINE

: SUBORDINATE ROUTINES USED:
:   'TOORIO' - TIME OUT OR OUTPUT INTERRUPT OR INPUT INTERRUPT
:   'CLRAW'  - CLEAR RQI AND WAIT FOR RDI TO CLEAR

: CALLING SEQUENCE:
:   JSR      PC,DVRXQ
:--
```

```
8884 044040          DVRXQ: BIT      #QRX,FLAG
8885 044040 032737 000004 007502 BEQ      DVREX          ;IF NOT RX THEN EXIT
8886 044046 001423          ;ELSE QUE RX
8887          ;CLEAR FLAG FOR RX
8888 044050 042737 000004 007502 BIC      #QRX,FLAG
8889 044056 112777 000144 146274 MOVB    #144,@BSEL0
8890 044064 004737 044746          JSR      PC,TOORIO      ;GO CHECK FJR IN OR OUT
8891 044070 017737 146274 010456 MOV      @SEL4,MODS      ;SET UP NEW MOD STATUS
8892 044076 013777 007362 146264 MOV      DVRXA,@SEL4
8893 044104 013777 007364 146262 MOV      DVRCC,@SEL6
8894 044112 004737 044640          JSR      PC,CLRAW
8895 044116 000207          DVREX: RTS      PC          ;LOAD CC AND ADDR
8896          ;CLEAR AND WAIT
          ;RETURN TO CALLER
```

8898
 8899
 8928
 8929
 8930
 8931
 8932
 8933
 8934
 8935
 8936
 8937
 8938
 8939
 8940
 8941
 8942
 8943
 8944
 8945
 8946
 8947
 8948
 8949
 8950
 8951
 8952
 8953
 8954
 8955
 8956
 8957
 8958
 8959
 8960
 8961
 8962
 8963
 8964
 8965
 8966
 8967
 8968
 8969

.SBTTL DEVICE TRANSMIT AND RECEIVE SUBROUTINE

```

  :++
  : FUNCTIONAL DESCRIPTION:
  : DVTXRX-DEVICE TRANSMIT AND RECEIVE ROUTINE
  : THIS CODE QUES THE TRANSMIT BUFFER TO THE DEVICE
  : IF NEEDED. THE CODE THEN WAITS FOR A TX COMPLE,
  : RX COMPLETE OR BOTH. THE CODE REPORTS A TIME OUT
  : ERROR IF NO BACC OUTPUT INTERRUPT IS RECIEVED BEFORE
  : 60 SECONDS. AFTER REPORTING ERROR TIMER IS RE STARTED
  : AND DEVICE WILL CONTINUE TO WAIT FOR INTERRUPT. CODE
  : ALSO REPORTS ERROR IF INPUT INTERRUPT OCCURS WHEN
  : EXPECTING OUTPUT INTERRUPT;WHEN RX BACC OCCURS WHEN
  : EXPECTING TX,AND WHEN TX INT. OCCURS WHEN EXPECTING
  : RECIEVE.
  
```

```

  : INPUTS:
  : 'DVTXA' = ADDRESS OF TRANSMIT MSG.
  : 'DVTCC' = BYTE COUNT OF TRANSMIT MSG.
  : 'QTX' BIT = SET IF TRANSMIT REQUESTED
  : 'ETX' BIT = SET IF TRNASMIT EXPECTED
  : 'ERX' BIT = SET IF RECIEVE EXPECTED
  
```

```

  : OUTPUTS:
  : 'DVTXA' = ADDRESS OF TX MSG. COMPLETED
  : 'DVTCC' = BYTE COUNT OF TX MSG. COMPLETED
  : 'QTX' = SET IF TX COMPLETED
  : 'DVRXA' = ADDRESS OF RX MSG. COMPLETED
  : 'DVRCC' = BYTE COUNT OF RX MSG. COMPLETED
  : 'QRX' = SET IF RX COMPLETED
  
```

```

  : SUBORDINATE ROUTINES USED:
  :
  : 'TOORIO' - TIME OUT OR OUTPUT INTERRUPT OR INTPUT INTERRUPT
  : 'CLRAW' - CLEAR RQI AND WAIT FOR RDI TO CLEAR
  : 'LGDVE' - LOG DEVICE ERROR TO EVENT LOG
  : 'OUTHDL' - OUTPUT INTERRUPT HANDLER CODE
  
```

```

  : CALLING SEQUENCE:
  : JSR PC,DVTXRX
  :--
  
```

```

8970 044120 032737 000010 007502 DVTXRX: BIT #QTX,FLAG ;ANY TX TO QUE
8971 044126 001423 BEQ DVTR3 ;IF NOT GO WAIT FOR OUPUT
8972 044130 042737 000010 007502 BIC #QTX,FLAG ;CLEAR FLAG
8973 044136 112777 000140 146214 MOVB #140,@BSELO
8974 044144 004737 044746 JSR PC,TOORIO ;GO CHECK FOR IN OR OUT
8975 044150 017737 146214 010456 MOV @SEL4,MODS ;PUT IN NEW MOD STAT
8976 044156 013777 007346 146204 MOV DVTXA,@SEL4
8977 044164 013777 007350 146202 MOV DVTCC,@SEL6
8978 044172 004737 044640 JSR PC,CLRAW ;CLEAR RQUI ANDWAIT
8979 044176 DVTR3:
8980 044176 012737 000074 007546 MOV #60.,TIMERS ;SET TIMER FOR 60 SECS
8981 044204 032737 000060 007502 TOINOT: BIT #CRX+#CTX,FLAG ;IS IT TX OR RX COMP ALREADY?
  
```

```
8982 044212 001071      BNE      DVTR4          ;IS SO EXIT
8983
8984 044214 005737 007546  TST      TIMERS          ;IS TIMER EXPIRED
8985 044220 001022
8986 044222 012737 024510 007432  MOV      #DVEM4,TEMP2
8987 044230 017737 146124 007434  MOV      @SELO,TEMP3
8988 044236 017737 146122 007436  MOV      @SEL2,TEMP4
8989 044244 004737 026434      JSR      PC,LGDVE
8990 044250 005237 007400      INC      ERRCNT
8991 044254      ERRSOFT 12,DVEM4,ERR13
      (4) 044254 104457      TRAP    CSERSOFT
      (5) 044256 000014      .WORD  12
      (5) 044260 024510      .WORD  DVEM4
      (5) 044262 026110      .WORD  ERR13
8992 044264 000744      BR       DVTR3          ;RETURN TO CHECK TIMER
8993
8994
8995 044266      TOIN1:  BREAK
      (3) 044266 104422      TRAP    CSBRK
8996 044270 032737 000001 007502  BIT      #ININT,FLAG     ;IS IT INPUT INTERRUPT
8997 044276 001425      BEQ     TOIN2           ;IF SO LOG ERROR
8998
8999 044300 012737 024602 007432  MOV      #DVEM5,TEMP2
9000 044306 017737 146046 007434  MOV      @SELO,TEMP3
9001 044314 017737 146044 007436  MOV      @SEL2,TEMP4
9002 044322 004737 026434      JSR      PC,LGDVE
9003 044326 042737 000001 007502  BIC      #ININT,FLAG     ;CLEAR BIT
9004 044334 005237 007400      INC      ERRCNT
9005 044340      ERRSOFT 13,DVEM5,ERR13
      (4) 044340 104457      TRAP    CSERSOFT
      (5) 044342 000015      .WORD  13
      (5) 044344 024602      .WORD  DVEM5
      (5) 044346 026110      .WORD  ERR13
9006 044350 000715      BR       TOINOT
9007
9008 044352 032737 000002 007502  TOIN2:  BIT      #OTINT,FLAG
9009 044360 001711      BEQ     TOINOT          ;IF NOT OUTPUT GO BACK AND
9010                                ;CHECK TIMER AGAIN
9011 044362 004737 045070      JSR      PC,OUTHDL      ;ELSE HANDLE OUTPUT AND RETURN
9012 044366 032737 000060 007502  BIT      #CTX+#CRX,FLAG  ;IS IT TX OR RX
9013 044374 001703      BEQ     TOINOT          ;IF NOT GO BACK AND TRY AGAIN
9014 044376 032737 000020 007502  DVTR4:  BIT      #CTX,FLAG    ;IS IT TX
9015 044404 001440      BEQ     DVTR5           ;IF NOT TRY RX
9016 044406 032737 000200 007502  BIT      #ETX,FLAG      ;IF SO SHOULD IT BE
9017 044414 001020      BNE     DVTR4A         ;IF IT SHOULD GO TO 4A
9018 044416 012737 025125 007432  MOV      #DVEM9,TEMP2   ;ELSE LOG ERROR
9019 044424 013737 045702 007434  MOV      TSEL4,TEMP3
9020 044432 013737 045704 007436  MOV      TSEL6,TEMP4
9021 044440 004737 026434      JSR      PC,LGDVE
9022 044444      ERRSOFT 14,DVEM9,ERR13 ;REPORT ERROR
      (4) 044444 104457      TRAP    CSERSOFT
      (5) 044446 000016      .WORD  14
      (5) 044450 025125      .WORD  DVEM9
      (5) 044452 026110      .WORD  ERR13
9023
9024 044454 000411      BR       DVTR4B        ;THEN CLEAR COMPL.FLAG
```

```

9025 044456 013737 045702 007346 DVTR4A: MOV TSEL4,DVTXA
9026 044464 013737 045704 007350 MOV TSEL6,DVTCC
9027 044472 052737 000010 007502 BIS #QTX,FLAG ;AND SET TX COMPL FLAG
9028 044500 042737 000020 007502 DVTR4B: BIC #CTX,FLAG ;ELSE CLEAR FLAG
9029 044506 032737 000040 007502 DVTR5: BIT #CRX,FLAG ;IS IT RX TOO?
9030 044514 001440 BEQ DVTR5X ;IF NOT THEN EXIT.
9031 044516 032737 000100 007502 BIT #ERX,FLAG ;TEST IS THIS SUPPOSED TO BE RX
9032 044524 001020 BNE DVTR5A ;IF YES PROCESS AS SUCH
9033 044526 012737 025036 007432 MOV #DVEM8,TEMP2
9034 044534 013737 045706 007434 MOV RSEL4,TEMP3
9035 044542 013737 045710 007436 MOV RSEL6,TEMP4 ;ELSE
9036 044550 004737 026434 JSR PC,LGDVE ;LOG ERROR
9037 044554 ERRSOF T 15,DVEM8,ERR13
(4) 044554 104457 TRAP CSERSOF T
(5) 044556 000017 .WORD 15
(5) 044560 025036 .WORD DVEM8
(5) 044562 026110 .WORD ERR13
9038
9039 044564 000411 BR DVTRX1 ;AND EXIT
9040 044566 013737 045706 007362 DVTR5A: MOV RSEL4,DVRXA
9041 044574 013737 045710 007364 MOV RSEL6,DVRCC
9042 044602 052737 000004 007502 BIS #QRX,FLAG
9043 044610 042737 000040 007502 DVTRX1: BIC #CRX,FLAG ;CLEAR FLAG FOR RX DONE
9044 044616 000207 DVTR5X: RTS PC ;AND EXIT
9045
  
```

9047
 9048
 9057
 9058
 9059
 9070
 9071
 (3)
 9072
 9073
 (3)
 (2)
 9074
 9085
 (3)
 9086
 9087
 (3)
 (2)
 9088
 9089
 9090
 9091
 9092
 9093
 9094
 9095
 9096
 9097
 9098
 9099
 9100
 9101
 9102
 9103
 9104
 9105
 9106
 9107
 9108
 9109
 9110
 9111
 9112
 9113
 9114
 (3)
 9115
 9116
 9117
 9118
 9119
 9120
 9121
 9122
 9123

; DEVICE DEPENDENT SUBROUTINES

.SBTTL DEVICE INTERRUPT SERVICE ROUTINES

```

BGNSRV DVINS
BIS #ININT,FLAG
ENDSRV
BGNSRV DVOUTS
BIS #OTINT,FLAG
ENDSRV
  
```

DVINS::
 L10021: RTI
 DVOUTS::
 L10022: RTI

```

:++
: FUNCTIONAL DESCRIPTION:
: CLRAW - CLEAR RQI AND WAIT FOR RDI TO GO AWAY
: THIS CODE CLEARS THE INPUT REQUEST BIT(RQI) SETS A
: TIMER UP TO TIME 50(OCTAL) TICKS AND MAKES SURE
: RDI CLEARS BEFORE TIMER EXPIRES. IF TIMER EXPIRES
: CODE REPORTS ERROR AND SETS UP TIMER AND WAITS AGAIN.
  
```

```

: SUBORDINATE ROUTINES USED:
: 'LGDVE' - LOG DEVICE ERROR (TIME OUT)
  
```

```

: CALLING SEQUENCE:
: JSR PC,CLRAW
:--
  
```

```

CLRAW: MOV (SP),PCADD ;SAVE PC OF CALLING ROUTINE
      BIC #RQI,@SELO
CLRA3: MOV #50,TIMER1 ;SET UP TIMER FOR 50(OCTAL) TICKS
CLRA1: TST TIMER1
      BEQ CLRA2 ;IF TIMER EXPIRED ERROR
      BREAK
      TRAP CSBRK
      BIT #RDI,@SELO ;IS RDI CLEAR
      BNE CLRA1 ;IF NOT GO CHECK TIMER
      ; ELSE
      ;RETURN TO CALLER
CLRA2: RTS PC
      MOV #DVEMO,TEMP2
      MOV @SELO,TEMP3
      MOV @SEL2,TEMP4
      JSR PC,LGDVE ;LOG DEVEICE EVENT 0
      INC ERRCNT
  
```

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

D 13
MACY11 30A(1052) 16-JUN-81 14:39 PAGE 14-32
DEVICE INTERRUPT SERVICE ROUTINES

SEQ 0159

9124 044734
(4) 044734 104457
(5) 044736 000020
(5) 044740 024252
(5) 044742 026032
9125 044744 000742

ERRSOFT 16,DVEMO,ERR9 ;WHILE WAITING FOR RDI

BR CLRA3 ;RESET TIMER AND CONTINUE

TRAP CSERSOFT
.WORD 16
.WORD DVEMO
.WORD ERR9

9127 .SBTTL TIME OUT OR INPUT INT. OR OUTPUT INT.
9128
9129
9130
9131

++
FUNCTIONAL DESCRIPTION:
TOORIO - TIME OUT OR INPUT INTERRUPT OR OUTPUT INTERRUPT
THIS ROUTINE SETS UP A TIMER FOR 100 (OCTAL) TICKS
THEN CHECKS FOR TIME OUT, OR INPUT INTERRUPT, OR OUTPUT
INTERRUPT. IF TIME OUT OCCURS IT REPORTS ERROR AND
RESTARTS TIMER. IF INPUT INTERRUPT OCCURS RETURN TO CALLER
IF OUTPUT INTERRUPT OCCURS LOG IT AND CONTINUE WAITING FOR
INPUT INTERRUPT.

USE OF FLAGS:
"OTINT" - SET BY OUTPUT INT ROUTINE
"ININT" - SET BY INPUT INT. ROUTINE
CLEARED BY THIS ROUTINE.

SUBORDINATE ROUTINES USED:
"OUTHDL" - OUTPUT INTERRUPT HANDLER

CALLING SEQUENCE:
JSR PC,TOORIO
--

```
9153 044746 011637 007462 TOORIO: MOV (SP),PCADD ;SAVE ADDR. OF CALLING ROUTINE
9154 044752 012737 000100 007542 MOV #100,TIMER1 ;SET UP TIMER
9155 044760 005737 007542 TOOR3: TST TIMER1 ;IS TIME EXPIRED
9156 044764 001022 BNE TOOR1 ;IF NOT CONTINUE
9157 ;IF YES ERROR
9158 044766 012737 024340 007432 MOV #DVEM1,TEMP2
9159 044774 017737 145364 007436 MOV @SEL2,TEMP4
9160 045002 017737 145352 007434 MOV @SELO,TEMP3
9161 045010 004737 026434 JSR PC,LGDVE
9162 045014 005237 007400 INC ERRCNT
9163 045020 ERRSOFT 17,DVEM1,ERR9
```

TRAP CSERSOFT
.WORD 17
.WORD DVEM1
.WORD ERR9

```
(4) 045020 104457
(5) 045022 000021
(5) 045024 024340
(5) 045026 026032
9164 045030 000746 BR TOORIO
```

```
9166 045032 TOOR1: BREAK
(3) 045032 104422
9167 045034 032737 000002 007502 BIT #OTINT,FLAG ;IS THERE AN OUTPUT
9168 ;PENDING
9169 045042 001402 BEQ TOOR2 ;IF NOT GO TO 2
9170 ;ELSE GO HANDL IT
9171 045044 004737 045070 JSR PC,OUTHDL
9172 045050 032737 000001 007502 TOOR2: BIT #ININT,FLAG ;IS THERE AN INPUT PENDING
9173 045056 001740 BEQ TOOR3 ;IF NOT GO BACK TO TIMER CK.
9174 045060 042737 000001 007502 BIC #ININT,FLAG ;ELSE CLEAR THE INPUT PEND FLAG
9175 045066 000207 RTS PC ;AND RETURN TO CALLER
9176
```

TRAP CSBRK

9178
 9179
 9180
 9181
 9182
 9183
 9184
 9185
 9186
 9187
 9188
 9189
 9190
 9191
 9192
 9193
 9194
 9195
 9196
 9197
 9198
 9199
 9200
 9201
 9202
 9203
 9204
 9205
 9206
 9207
 9208
 9209

.SBTTL OUTPUT INTERRUPT HANDLER

```

:++
: FUNCTIONAL DESCRIPTION:
:   OUTHDL - OUTPUT INTERRUPT HANDLER
:   THIS ROUTINE IS CALLED WHEN AN OUTPUT INTERRUPT HAS SET
:   THE 'OTINT' BIT IN THE 'FLAG' WORD. IT CHECKS FOR
:   AN RDO SIGNAL IF NO RDO THEN REPORT ILLEGAL INTERRUPT.
:   THEN IT CHECKS FOR BACC OUT IF NOT BACC OUT REPORT THE
:   TYPE OF OUTPUT ERROR. IF BACC OUT FIND IF RX OR TX
:   IF RX SET CRX BIT AND MOVE ADDR AND BYTE COUNT TO RSEL4
:   AND RSEL6. IF TX SET CTX BIT AND MOVE ADDR AND BYTE COUNT
:   TO TSEL4 AND TSEL6. CLEAR OTINT FLAG AND RETURN TO CALLER.

```

```

: USE OF FLAGS:
:   'OTINT' - SET BY OUPUT ROUTINE
:             CLEARED BY THIS ROUTINE
:   'DMRRUN' - SET BY DVINIT ROUTINE IF THIS IS DMR
:             CHECKED AND CLEARED BY THIS ROUTINE.
:   'CTX'    - SET IF TRANSMIT COMPLETED
:   'CRX'    - SET IF RECIEVE COMPLETED

```

```

: SUBORDINATE ROUTINES USED:
:   'LGDVE' -LOG DEVICE ERRORS TO EVENT LOG

```

```

: CALLING SEQUENCE
:   JSR    PC,OUTHDL

```

--

```

9210 045070 011637 007462      OUTHDL: MOV    (SP),PCADD      ;SAVE ADDR. OF CALLING ROUTINE
9211 045074 042737 000002 007502  BIC    #OTINT,FLAG
9212 045102 032777 000200 145254  BIT    #RDO,@SEL2      ;CLEAR PEND FLAG AND CHK FOR RDO
9213 045110 001023                BNE    OUTH1           ;IF RDO OK ...ELSE LOG ERROR
9214 045112 012737 024674 007432  MOV    #DVEM6,TEMP2
9215 045120 017737 145240 007434  MOV    @SEL2,TEMP3
9216 045126 017737 145242 007436  MOV    @SEL6,TEMP4
9217 045134 004737 026434      JSR    PC,LGDVE        ;GO LOG ERROR
9218 045140 005237 007400      INC    ERRCNT
9219 045144                ERRSOFT 18,DVEM6,ERR9
(4) 045144 104457                TRAP   C$ERSOFT
(5) 045146 000022                .WORD 18
(5) 045150 024674                .WORD DVEM6
(5) 045152 026032                .WORD ERR9
9220
9221                ;EXIT TEST IF ERROR
9222
9223 045154                ESCAPE TST
(3) 045154 104410                TRAP   C$ESCAPE
(3) 045156 000676                .WORD L10020-.
9224
9225 045160 032777 000001 145176  OUTH1: BIT    #BACC,@SEL2      ;IS THE OUTPUT BACC
9226 045166 001002                BNE    1$             ; BR IF NO
9227 045170 000137 045614                JMP    OUTH2          ;IF SO GO TO 2

```

```

9228                                     ;ELSE LOG ERROR AND PRINT IT
9229 045174 017737 145174 007436 1$:  MOV @SEL6,TEMP4
9230                                     ; IF NO BUFFER OUTPUT JUST COUNT THEM
9231
9232 045202 032737 000004 007436      BIT #BIT2,TEMP4
9233 045210 001404                                     BEQ OUTH6 ;IF NO BUFF INC COUNT AND EXIT
9234                                     ;ELSE GO TO 6
9235 045212 005237 007374      INC NOBUF
9236 045216 000137 045672      JMP OUTH6
9237
9238 045222 023727 012406 000006  OUTH6:  CMP OPTYP,#6 ;DMR ?
9239 045230 002426      BLT 51$ ;IF NOT DMR MODE SKIP TO 51
9240 045232 032737 002000 007502      BIT #BITUP,FLAG ;HERE BECAUSE OF BASE TABLE UPDATE REQ ? REV B BY EC
9241 045240 001402      BEQ 50$ ;NO,BRANCH REV B BY EC
9242 045242 000137 045672      JMP OUTH6 ;EXIT
9243 045246 032737 000040 007436 50$:  BIT #BIT5,TEMP4 ;IS IT RUN STATE
9244 045254 001414      BEQ 51$ ;IF NOT BRANCH
9245 045256 032737 001000 007502      BIT #DMRRUN,FLAG ;IS RUN EXPECTED
9246 045264 001405      BEQ 52$ ;IF NOT BRANCH
9247 045266 042737 001000 007502      BIC #DMRRUN,FLAG ;IF SO THEN CLEAR EXPECTED
9248 045274 000137 045672      JMP OUTH6 ;AND EXIT
9249 045300 012737 025437 007442 52$:  MOV #RUNSBM,CONOTM
9250 045306 012737 024751 007432 51$:  MOV #DVEM7,TEMP2
9251 045314 017737 145044 007434      MOV @SEL2,TEMP3
9252
9253 045322 004737 026434      JSR PC,LGDVE
9254 045326 012737 014635 007442      MOV #LPO,CONOTM ;LOAD 'NULL STRING' TO INIT CONOTM
9255 045334 032737 000001 007436      BIT #BIT0,TEMP4 ;IS THIS DATA CHECK
9256 045342 001403      BEQ 1$
9257 045344 012737 025424 007442      MOV #DATCKM,CONOTM
9258 045352 032737 000002 007436 1$:  BIT #BIT1,TEMP4 ;IS THIS TIMEOUT
9259 045360 001403      BEQ 2$
9260 045362 012737 025413 007442      MOV #TIMOM,CONOTM
9261 045370 032737 000010 007436 2$:  BIT #BIT3,TEMP4 ;IS THIS DDCMP MAINT RECVD
9262 045376 001403      BEQ 4$
9263 045400 012737 025373 007442      MOV #DDCMRM,CONOTM
9264 045406 032737 000020 007436 4$:  BIT #BIT4,TEMP4 ;IS THIS LOST DATA
9265 045414 001403      BEQ 5$
9266 045416 012737 025361 007442      MOV #LOSDAM,CONOTM
9267 045424 032737 000100 007436 5$:  BIT #BIT6,TEMP4 ;IS THIS DISCONNECT
9268 045432 001403      BEQ 6$
9269 045434 012737 025346 007442      MOV #DISCOM,CONOTM
9270 045442 032737 000200 007436 6$:  BIT #BIT7,TEMP4 ;IS THIS DDCMP START RECVD
9271 045450 001403      BEQ 7$
9272 045452 012737 025326 007442      MOV #DDCSR,CONOTM
9273 045460 032737 000400 007436 7$:  BIT #BIT8,TEMP4 ;IS THIS NON-EXSISTENT MEMORY
9274 045466 001403      BEQ 8$
9275 045470 012737 025310 007442      MOV #NXMM,CONOTM
9276 045476 032737 001000 007436 8$:  BIT #BIT9,TEMP4 ;IS THIS PROCEDURE ERROR
9277 045504 001403      BEQ 9$
9278 045506 012737 025270 007442      MOV #PROEM,CONOTM
9279 045514 023727 012406 000006 9$:  CMP OPTYP,#6 ;IS THIS DMR MODE
9280 045522 002416      BLT 11$ ;IF NOT BRANCH
9281 045524 032737 002000 007436      BIT #BIT10,TEMP4 ;IS THIS A RX IDLE
9282 045532 001403      BEQ 10$ ;IF NOT BRANCH
9283 045534 012737 025461 007442      MOV #RXIDM,CONOTM ;IF SO SET UP MESSAGE

```

```

9284 045542 032737 004000 007436 10$: BIT #BIT11,TEMP4 ;IS THIS CTS FAILED
9285 045550 001403 BEQ 11$ ;IF NOT BRANCH
9286 045552 012737 025505 007442 MOV #CTSFM,CONOTM ;IF SO SET UP MESSAGE
9287 045560 032737 010000 007436 11$: BIT #BIT12,TEMP4 ;IS THIS CD GLITCHED
9288 045566 001403 BEQ 12$ ;BR IF NO
9289 045570 012737 025471 007442 MOV #CDGLM,CONOTM ;IF SO SET UP MESSAGE
9290
9291 045576 005237 007400 12$: INC ERRCNT
9292 045602 ERRSOFT 19,DVEM7,ERR8
(4) 045602 104457 TRAP CSERSOFT
(5) 045604 000023 .WORD 19
(5) 045606 024751 .WORD DVEM7
(5) 045610 025750 .WORD ERR8
9293 045612 000427 BR OUTHEX ;CLEAR RDO AND RETURN TO CALLER
9294
9295 045614 OUTH2:
9296 045614 032777 000004 144542 BIT #RXBIT,@SEL2 ;IS THIS RX BACC OUT
9297 045622 001012 BNE OUTH3 ;IF NOT THEN IT MUST BE TX.
9298 045624 052737 000020 007502 BIS #CTX,FLAG
9299 045632 017737 144532 045702 MOV @SEL4,TSEL4
9300 045640 017737 144530 045704 MOV @SEL6,TSEL6
9301 045646 000411 BR OUTHEX
9302
9303 045650 052737 000040 007502 OUTH3: BIS #CRX,FLAG ;SET RX COMPL
9304 045656 017737 144506 045706 OUTH4: MOV @SEL4,RSEL4 ;THEN MOVE TO TEMP
9305 045664 017737 144504 045710 MOV @SEL6,RSEL6 ;AND SEL6 TO TEMP
9306 045672 042777 000200 144464 OUTHEX: BIC #RDO,@SEL2 ;CLEAR RDO
9307 045700 000207 RTS PC ;RETURN TO CALLER
9308 045702 000000 TSEL4: .WORD 0
9309 045704 000000 TSEL6: .WORD 0
9310 045706 000000 RSEL4: .WORD 0
9311 045710 000000 RSEL6: .WORD 0
9312
9313 045712 000207 RTS PC
9314
9315

```

9328
 9329
 9330
 9331
 9332
 9338
 9339
 9340
 9341
 9342
 9343
 9344
 9345
 9346
 9347
 9348
 9349
 9350
 9351
 9352
 9353
 9354
 9355
 9356
 9357
 9358
 9359
 9360
 9361
 9362
 9363
 9364
 9365
 9366
 9367
 9368
 9369
 9370
 9371
 9372
 (3)
 9373
 9374
 9375
 9376
 9377
 9378
 9379
 9380
 9381
 9382
 9383
 9384
 9385
 9386
 9387

.EVEN

.SBTTL REQUEST BASE TABLE UPDATE
 THIS ROUTINE ADDED FOR REV B BY EC

++
 :FUNCTIONAL DESCRIPTION:
 DVBTUP - THIS ROUTINE IS CALLED AT END OF PASS TO UPDATE THE DMR
 BASE TABLE IN LOCAL MEMORY BY ISSUING AN UPDATE BASE TABLE
 REQUEST TO THE DMR.

:USE OF FLAGS:
 FLAG - BITS IN THIS WORD ARE SET BY THE DEVICE INTERRUPT ROUTINE.
 WHEN SET TO A 1, THE FOLLOWING BITS MEAN
 ININT = BIT 1 = DEVICE INPUT READY
 OTINT = BIT 2 = DEVICE OUT READY

:INPUTS: NONE REQUIRED.

:SUBORDINATE ROUTINES USED:

CLRAW - CLEAR RQI AND WAIT FOR DEVICE TO CLEAR RDI.
 OUTHDL - THE DEVICE OUTPUT SERVICE ROUTINE.
 LGDVE - LOG A DEVICE ERROR TO EVENT LOG.

:CALLING SEQUENCE:
 JSR PC,DVBTUP

--
 :DEVICE BASE TABLE UPDATE ROUTINE

```

DVBTUP:
  CMP      OPTYP,#6      ;DMR IN DMR MODE ?
  BLT     40$           ;NO,BRANCH
  MOVB    #151,@BSELO   ;REQUEST BASE TABLE UPDATE
  BIS     #IEO,@SEL2    ;ENABLE INTERRUPT ON OUTPUT READY
  JSR     PC,TOORIO     ;WAIT FOR RDI
  JSR     PC,CLRAW      ;CLEAR RQI AND WAIT FOR RDI TO CLEAR
  MOV     #200,TIMER1   ;WAIT FOR INTERRUPT
10$:
  BREAK
                                     TRAP   C$BRK
  BIT     #OTINT,FLAG   ;OUTPUT READY ?
  BNE     30$           ;YES,BRANCH
  TST     TIMER1        ;TIME OUT ?
  BNE     10$           ;NO,BRANCH
  MOV     #DVEM3,TEMP2  ;DEVICE ERROR
  MOV     @SELO,TEMP3   ;SAVE REGISTER
  MOV     @SEL2,TEMP4   ;SAVE REGISTER
  JSR     PC,LGDVE      ;GO LOG DEVICE ERROR
  INC     ERRCNT        ;BUMP ERROR COUNT
  BR      40$           ;RETURN
30$:
  BIS     #BTUP,FLAG    ;SET BASE TABLE UPDATE BIT
  JSR     PC,OUTHDL     ;OUTPUT SERVICE ROUTINE
  BIC     #BTUP,FLAG    ;CLEAR BASE TABLE UPDATE BIT
40$:
  RTS     PC            ;RETURN
  
```

J 13

CZCLKBO DMR,DMC-11 DATA COMM. LINK TEST
CZCLKB.P11 16-JUN-81 14:38

MACY11 30A(1052) 16-JUN-81 14:39 PAGE 14-38
REQUEST BASE TABLE UPDATE

SEQ 0165

9388 046054
(3) 046054
(3) 046054 104401
9389
9390
9391

ENDTST

L10020: TRAP C\$ETST

9393
9394
9395
9396
9397
9398
9399
9400
9401
9402
9403
9404
9405
9406
9416
9417
9418
9419
9420
9421
9432
9433
9434
9435
9436
9437
9438
9439
9440
9441
9442
9443
9444

.SBTTL HARDWARE PARAMETER CODING SECTION

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

BGNHRD

046056
(3) 046056 000025
(3) 046060

.WORD L10023-L\$HARD/2
L\$HARD::

.SBTTL DEVICE INDEPENDENT SECTION

GPRML DPLX,0,1,YES

046060
(4) 046060 000130
(4) 046062 046132
(4) 046064 000001

.WORD T\$CODE
.WORD DPLX
.WORD 1

.SBTTL DEVICE DEPENDENT SECTION

GPRMA CSRADR,2,0,160000,177776,YES

046066
(4) 046066 001031
(4) 046070 046163
(4) 046072 160000
(4) 046074 177776

.WORD T\$CODE
.WORD CSRADR
.WORD T\$LOLIM
.WORD T\$HILIM

GPRMA VECTOR,4,0,300,776,YES

046076
(4) 046076 002031
(4) 046100 046211
(4) 046102 000300
(4) 046104 000776

.WORD T\$CODE
.WORD VECTOR
.WORD T\$LOLIM
.WORD T\$HILIM

GPRMD PRIOR,6,0,340,4,7,YES

046106
(4) 046106 003032
(4) 046110 046244
(4) 046112 000340
(4) 046114 000004
(4) 046116 000007

.WORD T\$CODE
.WORD PRIOR
.WORD 340
.WORD T\$LOLIM
.WORD T\$HILIM

: GPRMD DEVPRM,10,D,17,0,15.,YES

046120
(4) 046120 005032
(4) 046122 046272
(4) 046124 000007
(4) 046126 000000
(4) 046130 000007

.WORD T\$CODE
.WORD OPTN
.WORD 7
.WORD T\$LOLIM
.WORD T\$HILIM

: GPRMD BAUD,14,0,7,0,7,YES
: GPRMD LININ,16,0,7,0,7,YES

9445 046132

(2)

(3) 046132

9446

9447

9448

9449

9450

9451 046132 052506 046114 042040

9452

9460

9461

9462

9463 046163

104

053105

041511

CSRADR: .ASCIZ /DEVICE CSR ADDRESS : /

9464 046211

111

052116

051105

VECTOR: .ASCIZ /INTERRUPT VECTOR ADDRESS: /

9465 046244

047111

042524

051122

PRIOR: .ASCIZ /INTERRUPT PRIORITY : /

9466 046272

042504

044526

042503

OPTN: .ASCIZ /DEVICE OPTION TYPE : (0=DMC,5=DMR-DMC MODE ,7=DMR)/

9467

9468

9469

9470

046356

9471

9478

ENDHRD

.NLIST BEX

;DEVICE INDEPENDENT QUESTIONS

DPLX: .ASCIZ /FULL DUPLEX OPERATION : /

;DEVICE DEPENDENT QUESTION

CSRADR: .ASCIZ /DEVICE CSR ADDRESS : /

VECTOR: .ASCIZ /INTERRUPT VECTOR ADDRESS: /

PRIOR: .ASCIZ /INTERRUPT PRIORITY : /

OPTN: .ASCIZ /DEVICE OPTION TYPE : (0=DMC,5=DMR-DMC MODE ,7=DMR)/

.LIST BEX

.EVEN

L10023: .EVEN

9481
9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493
9502
9503
9504
9505
9506
9513
9514
9515
9516
9517
9518
9519
9520
9527
9528
(2)
(4)
(4)
(3)
9529
9530
9531

046356
046356 000030

046436

046436 000000
046440 000000
046442
046442

000001

:.SBTTL SOFTWARE PARAMETER CODING SECTION

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

: BGNSFT

: ENDSFT

.....
: TEMPORARY PATCH AREA - FOR DEBUG PURPOSES
:.....

\$PATCH: .BLKW 30

LASTAD

LSLAST:: ENDMOD

.END

.EVEN
.WORD 0
.WORD 0

DLTXRX	042242	8450	8457	8471#		
DLVM	020251	5092	5658#			
DMC =	000000	4524#				
DMCEND	003462	4890#	6317	6337	6368	
DMCIND	003432	4878#	6315	6336	6366	
DMCM	020242	5090	5656#			
DMC002	023604	4880	5736#			
DMC003	023625	4881	5737#			
DMC004	023662	4882	5738#			
DMC005	023723	4883	5739#			
DMC006	023756	4884	5740#			
DMC007	024013	4885	5741#			
DMC010	024050	4886	5742#			
DMC011	024103	4887	5743#			
DMC012	024125	4888	5744#			
DMC013	024147	4889	5745#			
DMC377	024206	4890	5746#			
DMFMT	030547	6414	6423#			
DMPE =	000053	4616#	5216			
DMPM	020255	5093	5659#			
DMPQ =	000054	4617#	5218			
DMP5 =	000052	4615#	5214	7593	7837	
DMRC6 =	000004	4525#	8710			
DMRC7 =	000005	4526#				
DMREND	003430	4875#	6325	6343	6376	
DMRIND	003250	4819#	6323	6342	6374	
DMRLUN=	001000	4555#	8793	9245	9247	
DMR000	020342	4819	5673#			
DMR001	020402	4820	5674#			
DMR002	020432	4821	5675#			
DMR003	020467	4822	5676#			
DMR004	020532	4823	5677#			
DMR005	020566	4824	5678#			
DMR006	020620	4825	5679#			
DMR007	020663	4826	5680#			
DMR010	020717	4827	5681#			
DMR011	020751	4828	5682#			
DMR012	021003	4829	5683#			
DMR013	021035	4830	5684#			
DMR014	021065	4831	5685#			
DMR015	021114	4832	5686#			
DMR016	021146	4833	5687#			
DMR017	021176	4834	5688#			
DMR020	021226	4835	5689#			
DMR021	021255	4836	5690#			
DMR022	021307	4837	5691#			
DMR023	021333	4838	5692#			
DMR024	021365	4839	5693#			
DMR025	021410	4840	5694#			
DMR026	021463	4841	5695#			
DMR027	021513	4842	5696#			
DMR030	021544	4843	5697#			
DMR031	021627	4844	5698#			
DMR032	021664	4845	5699#			
DMR033	021721	4846	5700#			
DMR034	021756	4847	5701#			

DMR035	022042	4848	5702#	
DMR036	022107	4849	5703#	
DMR037	022154	4850	5704#	
DMR040	022212	4851	5705#	
DMR041	022245	4852	5706#	
DMR042	022303	4853	5707#	
DMR043	022351	4854	5708#	
DMR044	022405	4855	5709#	
DMR045	022441	4856	5710#	
DMR046	022472	4857	5711#	
DMR047	022540	4858	5712#	
DMR050	022602	4859	5713#	
DMR051	022630	4860	5714#	
DMR052	022664	4861	5715#	
DMR053	022711	4862	5716#	
DMR054	022744	4863	5717#	
DMR055	022766	4864	5718#	
DMR056	023041	4865	5719#	
DMR057	023114	4866	5720#	
DMR060	023136	4867	5721#	
DMR061	023170	4868	5722#	
DMR062	023222	4869	5723#	
DMR063	023272	4870	5724#	
DMR064	023342	4871	5725#	
DMR065	023376	4872	5726#	
DMR066	023432	4873	5727#	
DMR067	023475	4874	5728#	
DMR177	023540	4875	5729#	
DMR6 =	000006	4527#	8713	
DMR7 =	000007	4528#		
DMSGAD	002176	4702#	6657	
DMSGCT	002150	4687#	6656	
DMUNKN	020322	4878	4879	5672#
DMVM	020316	5101	5667#	
DNM	020246	5091	5657#	
DOW =	000004	4496#	7939	8779
DPLX	046132	9419	9451#	
DPM	020217	5084	5650#	
DQM	020230	5087	5653#	
DSR =	000010	4648#	5049	
DTEM	020261	5094	5660#	
DUM	020222	5085	5651#	
DUMEX	031710	6572	6578#	
DUMPSR	031554	6562#	7595	
DUM1	031646	6566	6570#	
DUM2	031670	6569	6571#	
DUM3	031604	6565#	6576	
DUM4	031560	6563#	6575	
DUPM	020236	5089	5655#	
DVBTUP	045714	7370	8410	9364#
DVEMO	024252	5758#	9119	9124
DVEM1	024340	5760#	9158	9163
DVEM3	024424	5762#	8723	8728 9377
DVEM4	024510	5764#	8986	8991
DVEM5	024602	5766#	8999	9005
DVEM6	024674	5768#	9214	9219

LSCCP	002106	G	4306#	
LSCLEA	034650	G	4306	7360#
LSCO	002032	G	4306#	
LSDEPO	002011	G	4306#	
LSDESC	012424	G	4306	5432#
LSDESP	002076	G	4306#	
LSDEVP	002060	G	4306#	
LSDISP	002124	G	4306	4333#
LSDLY	002116	G	4306#	
LSDTP	002040	G	4306#	
LSDTYP	002034	G	4306#	
LSDU	034674	G	4306	7399#
LSDUT	002072	G	4306#	
LSDVTY	012410	G	4306	5412#
LSEF	002052	G	4306#	
LSEVI	002044	G	4306#	
LSETP	002102	G	4306#	
LSEXP1	002046	G	4306#	
LSEXP4	002064	G	4306#	
LSEXP5	002066	G	4306#	
LSHARD	046060	G	4306	9405#
LSHIME	002120	G	4306#	
LSHPCP	002016	G	4306#	
LSHPTP	002022	G	4306#	
LSHW	002130	G	4306	4351#
LSICP	002104	G	4306#	
LSINIT	033746	G	4306	7169#
LSLADP	002026	G	4306#	
LSLAST	046442	G	4306	9528#
LSLOAD	002100	G	4306#	
LSLUN	002074	G	4306#	
LSMREV	002050	G	4306#	
LSNAME	002000	G	4306#	
LSPRIO	002042	G	4306#	
LSPROT	033740	G	4306	7140#
LSPRT	002112	G	4306#	
LSREPP	002062	G	4306#	
LSREV	002010	G	4306#	
LSRPT	033732	G	4306	7099#
LSSPC	002056	G	4306#	
LSSPCP	002020	G	4306#	
LSSPTP	002024	G	4306#	
LSSTA	002030	G	4306#	
LSTEST	002114	G	4306#	
LSTIML	002014	G	4306#	
LSUNIT	002012	G	4306#	7255
L10000	002150		4351	4400#
L10001	025670		5851#	
L10002	025716		5855#	
L10003	025746		5859#	
L10004	026030		5873#	
L10005	026106		5878#	
L10006	026140		5882#	
L10007	026176		5886#	5888
L10010	026350		6072#	
L10011	033736		7131#	

NEW	034302	7210	7252#	7256		
NO	= 000036	4603#	5224	7953	7956	7982
NOBUF	007374	4955#	8040*	8401	9235*	
NOCLK	015102	5538#	7240	7508		
NODO	010660	5151#				
NOD1	010664	5152#				
NOD10	010742	5159#				
NOD100	011560	5239#				
NOD101	011564	5240#				
NOD102	011600	5241#				
NOD103	011604	5256#				
NOD104	011620	5257#				
NOD105	011624	5258#				
NOD106	011640	5259#				
NOD107	011644	5261#				
NOD11	010744	5160#				
NOD110	011660	5262#				
NOD111	011664	5265#				
NOD112	011670	5268#				
NOD113	011704	5269#				
NOD114	011710	5270#				
NOD115	011726	5271#				
NOD116	011732	5272#				
NOD117	011746	5273#				
NOD12	010756	5161#				
NOD120	011752	5274#				
NOD121	011766	5275#				
NOD122	011772	5276#				
NOD123	012006	5277#				
NOD124	012012	5278#				
NOD125	012026	5279#				
NOD126	012032	5280#				
NOD127	012046	5281#				
NOD13	010762	5162#				
NOD130	012052	5282#				
NOD131	012072	5283#				
NOD132	012076	5286#				
NOD133	012102	5287#				
NOD134	012106	5288#				
NOD135	012112	5289#				
NOD136	012116	5290#				
NOD137	012122	5291#				
NOD14	010776	5163#				
NOD140	012126	5292#				
NOD141	012130	5295#				
NOD142	012134	5296#				
NOD143	012140	5297#				
NOD144	012154	5298#				
NOD145	012160	5299#				
NOD146	012174	5300#				
NOD147	012200	5303#				
NOD15	011002	5164#				
NOD150	012204	5304#				
NOD151	012210	5305#				
NOD152	012214	5308#				
NOD153	012220	5319#				

NOD154	012242	5320#
NOD155	012246	5321#
NOD156	012262	5322#
NOD157	012266	5323#
NOD16	011016	5165#
NOD160	012310	5324#
NOD161	012314	5325#
NOD162	012336	5326#
NOD163	012342	5329#
NOD164	012346	5330#
NOD165	012352	5331#
NOD166	012356	5336#
NOD167	027266	6253#
NOD17	011022	5166#
NOD170	027272	6254#
NOD171	027276	6255#
NOD172	027300	6256#
NOD173	027314	6257#
NOD174	027316	6258#
NOD175	027332	6259#
NOD176	027334	6260#
NOD177	027346	6261#
NOD2	010670	5153#
NOD20	011026	5167#
NOD200	027350	6262#
NOD201	027364	6263#
NOD202	027370	6264#
NOD203	027374	6265#
NOD204	027410	6266#
NOD205	027412	6267#
NOD206	027426	6268#
NOD207	027430	6269#
NOD21	011040	5168#
NOD210	027446	6270#
NOD211	027452	6271#
NOD212	027456	6272#
NOD213	027460	6273#
NOD214	027462	6274#
NOD22	011044	5169#
NOD23	011056	5170#
NOD24	011062	5171#
NOD25	011064	5175#
NOD26	011070	5176#
NOD27	011104	5177#
NOD3	010672	5154#
NOD30	011110	5178#
NOD31	011126	5179#
NOD32	011132	5180#
NOD33	011150	5181#
NOD34	011154	5182#
NOD35	011172	5183#
NOD36	011176	5184#
NOD37	011214	5185#
NOD4	010706	5155#
NOD40	011220	5186#
NOD41	011244	5187#

TEMP	007426	4971#	6113*	6118*	6123*	6127*	6131*	6136*	6145*	6149*	6153*	6157*	6170	6186
		6187*	6188*	6189	6567*	6568	6612*	6613*	6614	6653*	6654*	6660	6794*	6803
		6806*	6809*	6841	7823*	7824	7875*	7876*	7880	7883	8246*	8247*	8251	8526*
		8532*	8534											
TEMP1	007430	4972#	6112*	6117*	6122*	6130*	6135*	6144*	6148*	6152*	6156*	6183	6409*	6414
		6801*	6803	6810*	6813*	6841								
TEMP2	007432	4973#	6137*	6138*	6192	6413*	6414	6656*	6658*	6662	6802*	6803	6814*	6817*
		6841	8208*	8218*	8229*	8235	8249*	8250	8268*	8337*	8402*	8474*	8481*	8495*
		8499*	8511*	8574*	8583*	8617*	8628*	8723*	8986*	8999*	9018*	9033*	9119*	9158*
		9214*	9250*	9377*										
TEMP3	007434	4974#	5858	5871	5876	5881	5885	6139*	6193	6408*	6414	6797*	6800*	6803
		8210*	8220*	8230*	8236	8247	8248	8249	8269*	8339*	8349	8403*	8441	8442
		8443	8444	8445	8476*	8483*	8493*	8500*	8512*	8575*	8584*	8619*	8629*	8724*
		8987*	9000*	9019*	9034*	9120*	9160*	9215*	9251*	9378*				
		4975#	5854	5871	5876	5881	5885	6140*	6167*	6194	8346*	8355*	8360*	8361
TEMP4	007436	8372	8401*	8494*	8725*	8988*	9001*	9020*	9035*	9121*	9159*	9216*	9229*	9232
		9243	9255	9258	9261	9264	9267	9270	9273	9276	9281	9284	9287	9379*
TEMP5	007440	4976#	6818*	6821*	6841									
TIMERS	007546	5034#	6066	6070*	8980*	8984								
TIMER1	007542	5032#	6060	6062*	7503*	7504	8705*	8720	9111*	9112	9154*	9155	9371*	9375
TIMER2	007544	5033#	6063	6065*										
TIMMIN	007534	5028#	6057*	6191	7243*									
TIMOM	025413	5786#	9260											
TIMSEC	007536	5029#	6054*	6055	6058*	6190	7244*							
TIMTCK	007540	5030#	6051*	6053*	6068	6188	7245*							
TM =	001000	4653#	5054											
TOINOT	044204	8981#	9006	9009	9013									
TOIN1	044266	8985	8995#											
TOIN2	044352	8997	9008#											
TOORIO	044746	8738	8757	8771	8777	8836	8890	8974	9153#	9164	9369			
TOOR1	045032	9156	9166#											
TOOR2	045050	9169	9172#											
TOOR3	044760	9155#	9173											
TOTCC	007422	4969#	6604*	6605	6611*	6613	6615*	7515*	7608*	7609	7630	7637*	7638	7666
TRA =	000001	4493#	7942											
TRAMOD=	000034	4601#	5189											
TRVACT	032762	6897	6908#	6924	6929	6934	6937	6957	7018	7036	7057	7081		
TRVALN	033554	6886	7040#											
TRVALP	033510	6885	7026#											
TRVBIF	033066	6882	6937#											
TRVBR	033056	6881	6934#											
TRVBRC	033002	6895	6915#	6935	6940	6959	7023	7038	7059	7085				
TRVDEC	033162	6888	6962#											
TRVERR	033020	6879	6924#											
TRVEXI	033040	6880	6929#											
TRVNMA	033202	6963	6966#											
TRVNOB	033012	6920#	6971	6958	7019	7037	7058							
TRVNUM	033174	6884	6965#											
TRVOCT	033174	6887	6964#											
TRVSPA	033110	6883	6943#											
TRVSTR	033642	6889	7063#											
TSEL4	045702	9019	9025	9299*	9308#									
TSEL6	045704	9020	9026	9300*	9309#									
TTL =	000001	4501#	7560	8747										
TTLLOP=	000044	4609#	5319											
TTOTCC	007356	4946#	6763	7516*	7608	7613	7630*	7798*						

	6241#	6297#	6359#	6405#	6414#	6434#	6452#	6457#	6463#	6470#	6471#	6487#	6495#	6502#	6515#
	6530#	6564#	6568#	6570#	6610#	6803#	6841#	6983#	7021#	7240#	7299#	7311#	7312#	7508#	7548#
	7575#	7579#	7611#	7620#	7640#	7654#	7750#	7824#	7865#	7890#	7961#	8005#	8018#	8461#	8534#
	8615#	8633#													
M\$PUT1	1981#	4228#	5850#	5854#	5858#	5871#	5872#	5876#	5877#	5881#	5885#	6180#	6183#	6222#	6236#
	6241#	6297#	6359#	6405#	6414#	6434#	6452#	6457#	6463#	6470#	6471#	6487#	6495#	6502#	6515#
	6530#	6564#	6568#	6570#	6610#	6803#	6841#	6983#	7021#	7240#	7299#	7311#	7312#	7508#	7548#
	7575#	7579#	7611#	7620#	7640#	7654#	7750#	7824#	7865#	7890#	7961#	8005#	8018#	8461#	8534#
	8615#	8633#													
MSRADI	2077#	4228#	6228#	7237#	7567#	8441#	8565#	9419#	9436#	9437#	9438#	9441#			
MSRBRO	1952#	4228#													
MSRNRO	1962#	4228#	7217#	7223#	7258#										
M\$SETS	2032#	4228#	4249#	4351#	5849#	5853#	5857#	5870#	5875#	5880#	5884#	6048#	7099#	7140#	7169#
	7342#	7360#	7399#	7436#	7489#	9071#	9085#	9405#							
M\$STAR	1733#	4228#													
M\$SVC	1933#	4228#	5850#	5851#	5854#	5855#	5858#	5859#	5871#	5872#	5873#	5876#	5877#	5878#	5881#
	5882#	5885#	5886#	5888#	6180#	6183#	6222#	6228#	6236#	6241#	6297#	6359#	6405#	6414#	6434#
	6452#	6457#	6463#	6470#	6471#	6487#	6495#	6502#	6515#	6530#	6564#	6568#	6570#	6610#	6803#
	6841#	6983#	7021#	7131#	7199#	7202#	7204#	7206#	7209#	7217#	7223#	7232#	7237#	7240#	7241#
	7258#	7299#	7311#	7312#	7314#	7315#	7331#	7351#	7372#	7374#	7390#	7410#	7426#	7447#	7463#
	7508#	7548#	7555#	7559#	7567#	7575#	7579#	7601#	7611#	7620#	7640#	7654#	7750#	7824#	7865#
	7890#	7961#	8005#	8018#	8352	8366	8375	8441#	8461#	8465	8534#	8565#	8615#	8633#	8719#
	8728	8991	8995#	9005	9022	9037	9114#	9124	9163	9166#	9219	9223#	9292	9372#	9388#
M\$TLAB	1929#	4228#	5850#	5851#	5854#	5855#	5858#	5859#	5871#	5872#	5873#	5876#	5877#	5878#	5881#
	5882#	5885#	5886#	6180#	6183#	6222#	6228#	6236#	6241#	6297#	6359#	6405#	6414#	6434#	6452#
	6457#	6463#	6470#	6471#	6487#	6495#	6502#	6515#	6530#	6564#	6568#	6570#	6610#	6803#	6841#
	6983#	7021#	7131#	7199#	7202#	7204#	7206#	7209#	7217#	7223#	7232#	7237#	7240#	7241#	7258#
	7299#	7311#	7312#	7314#	7315#	7331#	7351#	7372#	7374#	7390#	7426#	7463#	7508#	7548#	7555#
	7559#	7567#	7575#	7579#	7601#	7611#	7620#	7640#	7654#	7750#	7824#	7865#	7890#	7961#	8005#
	8018#	8352#	8366#	8375#	8441#	8461#	8465#	8534#	8565#	8615#	8633#	8719#	8728#	8991#	8995#
	9005#	9022#	9037#	9114#	9124#	9163#	9166#	9219#	9223#	9292#	9372#	9388#			
M\$TSTL	1921#	4228#	5850#	5851#	5854#	5855#	5858#	5859#	5871#	5872#	5873#	5876#	5877#	5878#	5881#
	5882#	5885#	5886#	6180#	6183#	6222#	6228#	6236#	6241#	6297#	6359#	6405#	6414#	6434#	6452#
	6457#	6463#	6470#	6471#	6487#	6495#	6502#	6515#	6530#	6564#	6568#	6570#	6610#	6803#	6841#
	6983#	7021#	7131#	7199#	7202#	7204#	7206#	7209#	7217#	7223#	7232#	7237#	7240#	7241#	7258#
	7299#	7311#	7312#	7314#	7315#	7331#	7351#	7372#	7374#	7390#	7426#	7463#	7508#	7548#	7555#
	7559#	7567#	7575#	7579#	7601#	7611#	7620#	7640#	7654#	7750#	7824#	7865#	7890#	7961#	8005#
	8018#	8352#	8366#	8375#	8441#	8461#	8465#	8534#	8565#	8615#	8633#	8719#	8728#	8991#	8995#
	9005#	9022#	9037#	9114#	9124#	9163#	9166#	9219#	9223#	9292#	9372#	9388#			
M\$WORD	1994#	4228#	4306#	4333#	5888#	6228#	7237#	7241#	7315#	7374#	7410#	7447#	7559#	7567#	7601#
	8352#	8366#	8375#	8441#	8465#	8565#	8728#	8991#	9005#	9022#	9037#	9124#	9163#	9219#	9292#
	9419#	9436#	9437#	9438#	9441#	9528									
M\$XFER	1682#	4228#													
NODCL	4261#	5151	5152	5153	5154	5155	5156	5157	5158	5159	5160	5161	5162	5163	5164
	5165	5166	5167	5168	5169	5170	5171	5175	5176	5177	5178	5179	5180	5181	5182
	5183	5184	5185	5186	5187	5188	5189	5190	5191	5192	5196	5197	5198	5199	5200
	5205	5206	5207	5208	5209	5212	5213	5214	5215	5216	5217	5218	5219	5222	5223
	5224	5225	5226	5227	5238	5239	5240	5241	5256	5257	5258	5259	5261	5262	5265
	5268	5269	5270	5271	5272	5273	5274	5275	5276	5277	5278	5279	5280	5281	5282
	5283	5286	5287	5288	5289	5290	5291	5292	5295	5296	5297	5298	5299	5300	5303
	5304	5305	5308	5319	5320	5321	5322	5323	5324	5325	5326	5329	5330	5331	5336
	6253	6254	6255	6256	6257	6258	6259	6260	6261	6262	6263	6264	6265	6266	6267
	6268	6269	6270	6271	6272	6273	6274								
OPEN	1171#	4228#													
POINTE	1176#	4228#	4286												
PRINTB	1239#	4228#	5850	5854	5858	5871	5872	5876	5877	5881	5885				

PRINTF	1279#	4228#	6180	6183	6222	6236	6241	6297	6359	6405	6414	6564	6568	6570	6610
	6983	7021	7240	7508	7548	7575	7579	7611	7620	7640	7654	7750	7824	7865	7890
	7961	8005	8018	8461	8534	8615	8633								
PRINTS	1319#	4228#	6434	6452	6457	6463	6470	6471	6487	6495	6502	6515	6530	6803	6841
PRINTX	1359#	4228#													
READBU	1399#	4228#	7232												
REDEF	1403#	4228#	7202	7204	7206	7209									
RFLAGS	1408#	4228#													
SETPRI	1413#	4228#	7314	7372											
SETVEC	1418#	4228#	7299	7311	7312										
SLASH	1424#	4228#													
STARS	1438#	4228#													
SVC	1452#	4227#	4228												
XFER	1612#	4228#	5888#	7241#	7315#	7374#	7410#	7447#	7559#	7601#					
XFERF	1616#	4228#													
XFERT	1620#	4228#													

. ABS. 046442 000

ERRORS DETECTED: 0

CZCLKB/I,CZCLKB.SEQ/CRF/DOC=SVC34R.MLB,CZCLKB.P11

RUN-TIME: 24 30 3 SECONDS

RUN-TIME RATIO: 68/59=1.1

CORE USED: 21K (41 PAGES)

DOCUMENT PAGES: 199