CYCLE 11 HELPFUL HINTS

THIS PAPER DESCRIBES HELPFUL HINTS ON HOW TO USE CYCLE 11 OF NOS/VE. IT IS INTENDED TO SUPPLEMENT, RATHER THAN TO REPLACE, THE STANDARD NOS/VE DOCUMENTATION. IF YOU HAVE ANY QUESTIONS OR SUGGESTIONS, PLEASE SEE TOM MCGEE OR BONNIE SWIERZBIN. APPENDIX A LISTS BACKGROUND DOCUMENTS AND HOW TO OBTAIN THEM.

THE PROCEDURE TO PRINT COPIES OF THIS DOCUMENT HAS CHANGED FOR CYCLE 11. IT IS NOW POSSIBLE TO PRINT ANY SECTION(S) (OR SUBSECTIONS) AND TO SPECIFY A PREVIOUS LEVEL OF HINTS AGAINST WHICH REVISION BARS ARE TO BE GENERATED. THE COMMAND USED TO OBTAIN FULL COPIES OF THIS DOCUMENT WHILE RUNNING ON S/N 101 AT ARDEN HILLS IS STILL THE SAME

    SES,INT1.LISTHINTS C=<NUMBER OF COPIES>

TO OBTAIN A COPY WITH REVISION BARS AGAINST THE HELPFUL HINTS OF A PREVIOUS BUILD, ONE CAN TYPE

    SES,INT1.LISTHINTS REVB=<BUILD CYCLE NUMBER> C=<NUMBER OF
                        COPIES>

THE C PARAMETER IS OPTIONAL AND DEFAULTS TO ONE. ALLOWABLE VALUES FOR REVB ARE 5, 7, AND 9.

TO OBTAIN A LISTING OF ONLY SELECTED SECTIONS, ONE SHOULD ENTER

    SES,INT1.LISTHINTS SECTION=<LIST OF SECTION NUMBERS ENTERED
                        AS STRINGS>

FOR EXAMPLE: SES,INT1.LISTHINTS SECTION=('1.0','4.6.1.3')

CYCLE 11, JANUARY 1983

UPDATE_HISTORY

DATE	CHANGES
12/22/80	REVISIONS FOR NOS/VE PHASE C
2/12/81	ADDITIONAL REVISIONS FOR NOS/VE PHASE C
6/09/81	REVISIONS FOR NOS/VE BUILD N
6/19/81	ADDITIONAL REVISIONS FOR NOS/VE BUILD N
8/28/81	REVISIONS FOR NOS/VE BUILD O
11/06/81	REVISIONS FOR NOS/VE BUILD P
3/01/82	REVISIONS FOR NOS/VE BUILD Q
4/15/82	REVISIONS FOR NOS/VE CYCLE 2
5/01/82	REVISIONS FOR NOS/VE CYCLE 3
6/30/82	REVISIONS FOR NOS/VE CYCLE 5
7/29/82	REVISIONS FOR NOS/VE CYCLE 7
11/05/82	REVISIONS FOR NOS/VE CYCLE 9
11/29/82	REVISIONS FOR NOS/VE CYCLE 10
12/29/82	REVISIONS FOR NOS/VE CYCLE 11
1/24/83	ADDITIONAL REVISIONS FOR NOS/VE CYCLE 11

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

O TWO SECTIONS HAVE BEEN ADDED TO THIS DOCUMENT: SECTION 1.1.6 DESCRIBES THE DISPLAY_VARIABLE_LIST COMMAND AND SECTION 2.15.15 DESCRIBES THE USE OF MESSAGE TEMPLATES.

O AS OF BUILD 11.13, THE 170 PART OF THE NOS/VE REMOTE HOST (IRHF170) WILL ROLLOUT FOR 15 SECONDS IF THERE ARE NO QUEUE FILES TO TRANSFER. ALSO ALL 170 PARTNER JOBS (GETF/REPF) WILL NOW BE STARTED BY THE NVE SUBSYSTEM JOB RATHER THAN THE IRHF170 JOB. STATISTICS FOR QUEUE FILES AND PARTNER JOBS CAN BE FOUND IN THE APPROPRIATE JOB'S DAYFILE.

O TERMINATION DUMPS -- AS OF CYCLE 9, NOS/VE HAS THE ABILITY TO PRODUCE AN EDD FORMAT DUMP WITHOUT TAKING NOS DOWN. THIS DUMP IS PRODUCED WHENEVER THE SYSTEM FAILS DUE TO A SOFTWARE/HARDWARE ERROR OR OPERATOR DROPVE COMMAND.

   WHEN THE SYSTEM FAILS, THE NVE CONTROL POINT WILL BRING UP A K-DISPLAY INDICATING THAT TERMINATION IS IN PROGRESS. THE APPROPRIATE RESPONSE IS K.*RUN. A SHORT TIME AFTER THE *RUN IS ENTERED, A FLASHING TAPE REQUEST WILL BE SEEN ON THE NVE CONTROL POINT B-DISPLAY. AT THIS TIME A 9-TRACK TAPE SHOULD BE ASSIGNED TO THE NVE JOB. THE NVE JOB WILL PRODUCE AN EDD FORMAT DUMP ON THE TAPE (F=S,PE). FOLLOWING THE EDD DUMP WILL BE THE DUMPS AND DAYFILES OF THE IRHF AND PASSON JOBS. ANY FATAL ERRORS ENCOUNTERED WHILE WRITING THE TAPE WILL CAUSE THE CURRENT TAPE TO BE RETURNED AND ANOTHER REQUESTED.

   THE DUMP MAY BE DISPLAYED BY EITHER DSDI OR THE 180 SIMULATOR (LEVEL 137 OR LATER). THE DUMP IS DIFFERENT FROM A STANDARD EDD DUMP IN THAT ONLY NOS/VE MEMORY AND PP'S ARE PRESENT ON THE DUMP, AND THE JPS VALUE DOES NOT CORRECTLY REFLECT THE FAILED NOS/VE ENVIRONMENT. NOS MEMORY IS NOT PRESENT ON THE DUMP.

   THIS TYPE OF DUMP IS SUFFICIENT FOR MOST NOS/VE FAILURES. IF A NOS PROBLEM IS SUSPECTED THE PROCESSOR SHOULD BE HALTED AND A STANDARD EDD DUMP PERFORMED.

O THE PROCEDURE FOR MIGRATING THE SYSTEM FROM LEVEL N TO LEVEL N+1 BY MEANS OF INSTALLING A NEW SET OF JOB TEMPLATES HAS BEEN IMPLEMENTED. THIS PROCEDURE IS DESCRIBED IN SECTION 5.3.

O THE 'TM' FORMS CODE IS GONE. (SORRY, TOM.) THE DEFAULT FORMS CODE IS NO FORMS CODE, WHICH MEANS 180 OUTPUT WILL PRINT ON ANY PRINTER WITH EC=A9 AND NO FORMS CODE ASSIGNED TO IT.

O INPUT QUEUE FILES FOR BATCH JOBS ONLY AND OUTPUT QUEUE FILES ARE NOW RECOVERED AND RE-QUEUED. EXECUTING JOBS ARE NOT RECOVERED IN R1.

O THE VEDISPLAY CP (CONTROL POINT) HAS CHANGED TO VEDISPLAY AJ (ACTIVE JOBS).

O IN THE PAST, THE DS PROCEDURE, ALONG WITH THE FILES DCFILE, RDCFILE, AND NVEPROL HAVE BEEN MAINTAINED BY THE 170 SCU AND EXPANDED TO (PERMANENT) FILES BY THE EDITDS PROCEDURE. AS OF CYCLE 10 (USE1 AS OF 11/1/82) THESE FILES WILL BE DECKS ON VE170PL AND WILL BE MAINTAINED VIA MADIFY MODSETS. THE DECK NAMES ARE

        DDSPROC DS PROCEDURE - EVERYTHING IN ONE DECK
        DNVEPRL PROLOG FILE
        DCFILE DEADSTART COMMAND FILE

1.1 NOS/VE_USAGE_EXAMPLES

1.1.1 EXECUTING PROGRAMS

PROCESS

CREATE AN OBJECT TEXT FILE BY COMPILING A PROGRAM ON NOS. THEN PERFORM THE FOLLOWING STEPS ON NOS/VE

- ACQUIRE ANY NECESSARY LIBRARIES (WHICH ARE NOT QUOTED IN TEXT EMBEDDED DIRECTIVES) BY EITHER

    O ATTACHING THEM FROM THE SYSTEM CATALOG, EITHER EXPLICITLY OR VIA PROLOG
                                OR
    O CREATING THE LIBRARY FILE VIA THE OBJECT LIBRARY GENERATOR
                                OR
    O STAGING THE LIBRARY FILE FROM NOS TO NOS/VE USING THE GET_OBJECT_LIBRARY COMMAND.

- GET THE FILE FROM NOS AND CONVERT THE OBJECT TEXT FILE FROM THE CI DATA MAPPING TO II DATA MAPPING BY EXECUTING THE CONVERT_OBJECT_FILE COMMAND.

- LOAD AND EXECUTE THE PROGRAM VIA THE EXECUTE_TASK COMMAND, SPECIFYING THE NECESSARY LIBRARIES WITH THE LIBRARY PARAMETER; ALTERNATIVELY SET_PROGRAM_ATTRIBUTES MAY BE USED TO INCLUDE THE LIBRARIES IN ALL SUBSEQUENT EXECUTE_TASK COMMANDS.

- STAGE THE LOADMAP FROM NOS/VE TO NOS FOR PRINTING BY USING EITHER

    O THE REPLACE_FILE COMMAND WITH A6 CONVERSION MODE SPECIFIED IF RUNNING ON THE SIMULATOR.
                                OR
    O THE PRINT_FILE COMMAND IF RUNNING ON THE HARDWARE.

CYCLE 11, JANUARY 1983

EXAMPLES

THE FOLLOWING IS AN EXAMPLE COMMAND SEQUENCE FOR EXECUTING A PROGRAM NOT REQUIRING ANY LIBRARIES FOR LOADING

ASSUMPTIONS: ALL MODULES TO BE LOADED ARE CONTAINED ON THE NOS PERMANENT FILE 'CITXTRS'.

```
CONVERT_OBJECT_FILE CITXTRS
EXECUTE_TASK CITXTRS PARAMETER='PROGRAM PARAMETERS'
PRINT_FILE LOADMAP
```

THE FOLLOWING IS AN EXAMPLE COMMAND SEQUENCE FOR EXECUTING A PROGRAM REQUIRING LIBRARIES FOR LOADING

ASSUMPTIONS: THE NOS PERMANENT FILE 'CITXTRS' CONTAINS OBJECT TEXT GENERATED BY THE CYBIL CI COMPILER. THE COMPILER MODULES REFERENCE PROCEDURES CONTAINED ON THE LIBRARY 'MYLIB' AND THE CYBIL RUN-TIME LIBRARY. THESE LIBRARIES HAVE BEEN GENERATED ON NOS/VE AND SAVED ON NOS.

```
GET_OBJECT_LIBRARY MYLIB
SET_PROGRAM_ATTRIBUTES LOAD_MAP_OPTIONS=(BLOCK,ENTRY_POINT,SEGMENT)
CONVERT_OBJECT_FILE CITXTRS
EXECUTE_TASK CITXTRS 'PROGRAM PARAMETERS' LIBRARY=MYLIB
PRINT_FILE LOADMAP
```

1.1.2 CREATE OBJECT LIBRARY ON NOS/VE AND SAVE IT ON NOS

NOTES

O CLG0170 IS NOS PERMANENT FILE NAME FOR FILE CONTAINING CI OBJECT TEXT FOR MODULES TO BE INCLUDED IN THE LIBRARY.

O IITEXT180 IS NOS/VE LOCAL FILE NAME FOR FILE CONTAINING II OBJECT TEXT FOR MODULES TO BE INCLUDED IN THE LIBRARY.

O LIBRARY180 IS NOS/VE LOCAL FILE NAME FOR THE LIBRARY BEING CREATED.

O ILIB170 IS NOS PERMANENT FILE NAME FOR FILE CONTAINING THE LIBRARY.

NOS/VE JOB COMMANDS

```
CONVERT_OBJECT_FILE IITEXT180 CLG0170
CREATE_OBJECT_LIBRARY
ADD_MODULE LIBRARY=IITEXT180
GENERATE_LIBRARY LIBRARY=LIBRARY180
QUIT
```

REPLACE_FILE LIBRARY180 ILIB170 DC=B56

1.1.3 MODIFY A PREVIOUSLY SAVED OBJECT LIBRARY

NOTES_

O ILIB170 IS NOS PERMANENT FILE NAME FOR FILE CONTAINING THE OLD
   LIBRARY

O LIBRARY180 IS NOS/VE LOCAL FILE NAME FOR FILE CONTAINING THE
   OLD LIBRARY

O CMOD170 IS NOS PERMANENT FILE NAME FOR FILE CONTAINING CI
   OBJECT TEXT FOR THE NEW MODULE

O NEWIIMODULE IS NOS/VE LOCAL FILE NAME FOR FILE CONTAINING II
   OBJECT TEXT FOR THE NEW MODULE

O NEWLIBRARY IS NOS/VE LOCAL FILE NAME FOR THE LIBRARY BEING
   CREATED

O NLIB170 IS NOS LOCAL FILE NAME FOR NEW LIBRARY

   NOS/VE JOB COMMANDS

GET_OBJECT_LIBRARY LIBRARY180 ILIB170
CONVERT_OBJECT_FILE NEWIIMODULE CMOD170
CREATE_OBJECT_LIBRARY
ADD_MODULE LIBRARY=LIBRARY180
REPLACE_MODULE LIBRARY=NEWIIMODULE
GENERATE_LIBRARY LIBRARY=NEWLIBRARY
QUIT
REPLACE_FILE NEWLIBRARY NLIB170 DC=B56

1.1.4 ROUTE AN INPUT FILE FROM NOS TO NOS/VE

RUNNING FROM AN INTERACTIVE TERMINAL, ENTER

GET,FILENAME.
ROUTE,FILENAME,DC=LP,FC=RH.

THE INPUT FILE WHICH IS SENT TO NOS/VE MUST BE IN 6/12 ASCII
(OR DISPLAY CODE SUBSET). THE JOB FILE MUST BE A SINGLE
PARTITION NOS RECORD CONTAINING NOS/VE COMMANDS. THE FIRST
STATEMENT MUST BE A VALID LOGIN COMMAND WITH USER, PASSWORD AND
FAMILY NAME SPECIFIED. MULTI PARTITION INPUT FILES ARE NOT
SUPPORTED BY NOS/VE SO NOS DATA FILES USED BY THE PROGRAM MUST BE
OBTAINED THROUGH THE GET_FILE COMMAND.

1.1.5 PRINT A NOS/VE FILE

AT NOS/VE JOB TERMINATION THE JOB LOG WILL BE AUTOMATICALLY RETURNED TO NOS. THE JOB LOG WILL BE APPENDED TO THE NOS/VE OUTPUT FILE OUTPUT. NOS/VE PRINT FILES MUST BE WRITTEN BY BAM AS 8/8 ASCII RT=V. PRINT FILES WILL BE CONVERTED FROM 8/8 ASCII RT=V TO NOS 8/12 ASCII WHEN THEY ARE SENT TO NOS AND WILL BE PRINTED IN UPPER/LOWER CASE.

ALL NOS/VE OUTPUT FILES WILL APPEAR IN THE NOS OUTPUT QUEUE (NOS Q,PR DISPLAY) WITH THE NAME NVEXXXX AS A BANNER. IN ORDER TO PRINT A NOS/VE FILE, THE FOLLOWING COMMAND MUST BE ISSUED WITHIN YOUR JOB OR BE ENTERED FROM THE SYSTEM CONSOLE VIA THE OPERATOR FACILITY

PRINT_FILE FILENAME

1.1.6 MESSAGE TEMPLATES ON NOS/VE

TO MAKE FULL USE OF THE FACILITIES OF THE NOS/VE SYSTEM MESSAGE GENERATOR, A PRODUCT OR USER PROGRAM MUST MAKE AVAILABLE A MESSAGE TEMPLATE MODULE. MESSAGE TEMPLATE MODULES ARE PRODUCED VIA THE UTILITY GENMT (AN SES PROCEDURE IN NOS/VE INTEGRATION'S PROCEDURE LIBRARY), AND THEN COMPILED WITH CYBIL.

THE MESSAGE TEMPLATE MODULE FOR A PRODUCT OR USER PROGRAM WILL TYPICALLY EXIST IN TWO PLACES

- ON ONE OF THE OBJECT FILES OR LIBRARIES THAT COMPRISE THE PRODUCT OR USER PROGRAM (THIS MAKES MESSAGE TEMPLATES BELONGING TO THE PRODUCT OR USER PROGRAM AVAILABLE WHILE IT IS "ACTIVE")

- ON AN OBJECT LIBRARY IN THE COMMAND LIST (THIS MAKES MESSAGE TEMPLATES BELONGING TO THE PRODUCT OR USER PROGRAM AVAILABLE WHEN IT IS NOT "ACTIVE").

WHEN THE MESSAGE FORMATTER NEEDS A TEMPLATE IT SEARCHES THE FOLLOWING

- SYSTEM: THESE ARE THE TEMPLATES FOR THE OPERATING SYSTEM AND CERTAIN PRODUCTS. THEY ARE "BUILT-N" AND ALWAYS AVAILABLE.

- PROGRAM: THESE ARE THE TEMPLATES FOR THE CURRENTLY ACTIVE (RUNNING) PRODUCT OR USER PROGRAM. THEY ARE ACCESSED VIA THE DYNAMIC LOAD INTERFACE (PMP$LOAD).

- COMMAND LIST: THESE ARE THE TEMPLATES FOR ALL PRODUCTS. EACH OBJECT LIBRARY IN THE COMMAND LIST IS SEARCHED AND IF

THE TEMPLATE IS FOUND IT IS ACCESSED VIA AN INTERNAL FORM OF THE DYNAMIC LOAD INTERFACE (PMP$LOAD_FROM_LIBRARY). PRODUCT SET MESSAGE TEMPLATES RESIDE ON THE LIBRARY THAT IS PART OF THE $SYSTEM COMMAND LIST ENTRY. TEMPLATES FOR USER PROGRAMS MAY RESIDE ON ANY OTHER LIBRARY IN THE COMMAND LIST.

SINCE THE LOADER IS USED TO ACCESS NON-SYSTEM TEMPLATES IT IS NOT POSSIBLE TO CHANGE A TEMPLATE (ACTUALLY A RANGE OF TEMPLATES) IN A TASK ONCE IT HAS (THEY HAVE) BEEN LOADED. THIS SHOULD ONLY IMPACT DEVELOPERS OF PRODUCTS OR PROGRAMS TRYING TO CHECK OUT NEW MESSAGES.

2.0 COMMAND_INTERFACE_STATUS

2.1 ACCESS_TO_NOS/VE_IN_DUAL_STATE

2.1.1 LOGIN TO NOS/VE

IT IS NO LONGER NECESSARY TO FORCE THE FIRST LOGIN ATTEMPT TO FAIL. TO ACCESS VEIAF RESPOND AS FOLLOWS TO THE "FAMILY:" PROMPT

,USER,PASSWORD,VEIAF

YOU CAN ACCESS VEIAF FROM IAF BY DOING "HELLO,VEIAF" OR BY ANSWERING VEIAF TO THE SYSTEM PROMPT "APPLICATION:".

2.1.2 TERMINAL USAGE

1) THE SLANT (/) IS THE PROMPT TO ENTER A NOS/VE COMMAND. ANY NORMAL NOS/VE COMMAND CAN NOW BE ENTERED (CONTINUATION LINES ARE PROMPTED WITH ../). THE FULL ASCII CHARACTER SET, LOWER OR UPPER CASE AND ALL SPECIAL CHARACTERS, CAN BE USED.

2) A LOGOUT COMMAND WILL CAUSE THE NOS/VE INTERACTIVE JOB TO TERMINATE. A NEW NOS/VE INTERACTIVE JOB CAN THEN BE STARTED BY RESPONDING TO THE 'APPLICATION:' PROMPT WITH VEIAF.

3) TERMINAL BREAKS (CONTROL-T AND CONTROL-P) CAN BE USED TO TERMINATE A TASK OR COMMAND AND SUSPEND A TASK AND ENTER A NEW TASK TO PROCESS SCL COMMANDS. CONTROL-T CAUSES A TERMINATE BREAK AND CONTROL-P CAUSES A PAUSE BREAK. TERMINATE BREAK WILL TERMINATE A COMMAND OR THE MOST RECENTLY EXECUTED TASK. A PAUSE BREAK WILL SUSPEND EXECUTION AND ALLOW COMMANDS TO BE ENTERED. WHEN A TERMINAL IS IN PAUSE BREAK STATE, TWO ADDITIONAL COMMANDS ARE AVAILABLE

RESUME_COMMAND - RESUME EXECUTION AT THE POINT OF
        INTERRUPTION.

TERMINATE_COMMAND - CAUSE A TERMINATE BREAK CONDITION AS IF
        A TERMINATE BREAK HAD BEEN ENTERED.

BOTH TERMINATE BREAK AND PAUSE BREAK ARE AVAILABLE TO PROGRAMS AS CONDITIONS VIA THE PROGRAM MANAGEMENT CONDITION MECHANISM.

2.1.3 NOS/VE PROGRAM ACCESS TO THE TERMINAL

1) INTERACTIVE NOS/VE JOBS ARE ABLE TO OBTAIN TERMINAL INPUT THROUGH THE AMP$GET_NEXT OR AMP$GET_PARTIAL PROGRAM INTERFACE WHICH CAN BE USED BY BOTH TASK SERVICES AND USER RING PROGRAMS. INTERACTIVE PROGRAMS WHICH USE THIS INTERFACE SHOULD BE ABLE TO HANDLE BOTH UPPER AND LOWER CASE INPUT IN ORDER TO MAKE THEM MORE CONVENIENT TO USE IN BOTH 64 AND 96 CHARACTER SET MODES.

2.2 COMMAND_AND_PARAMETER_NAMES

DURING THE NEXT FEW MONTHS A COMMAND SUPPORTED BY THE SYSTEM MAY NOT BE IN SYNC WITH YOUR COMMAND INTERFACE DOCUMENT. THE PARAMETER DESCRIPTOR TABLE GIVES AN ACCURATE, CONCISE DESCRIPTION OF THE COMMAND INTERFACE AS CURRENTLY SUPPORTED.

PDT_READER'S_GUIDE

THE DEFINITION OF A COMMAND'S PARAMETER LIST IS ENCLOSED IN PARENTHESIS WITH A PARAMETER DESCRIPTION PER LINE. EACH DESCRIPTION HAS THE GENERAL FORM

PARAMETER NAME: ALLOWED PARAMETER VALUES = PARAMETER DEFAULT VALUE

PARAMETER NAMES - DESCRIBES THE PARAMETER NAME AND ANY ABBREVIATIONS.

ALLOWED PARAMETER VALUES - DESCRIBES THE KIND OF VALUE ALLOWED AND WHETHER A LIST OF VALUES IS POSSIBLE. THE VALUE KIND CAN BE FURTHER QUALIFIED. IN SOME CASES, THE ACTUAL VALUES ALLOWED ARE DESCRIBED USING THE KEY NOTATION. THE VALUE KINDS INCLUDE INTEGER, STRING, NAME, FILE, STATUS.

PARAMETER DEFAULT VALUES - DESCRIBES THE DEFAULTING RULES AND/OR VALUES FOR THE PARAMETER. $REQUIRED AND $OPTIONAL ARE OBVIOUS. OTHER VALUES IN THIS POSITION WILL BE TREATED AS IF THEY WERE ENTERED BY THE USER ON COMMAND INVOCATION.

SEE THE PROC COMMAND IN THE COMMAND INTERFACE ERS FOR MORE DETAILS.

THE PDTS FOR THE COMMANDS CURRENTLY IN THE SYSTEM CAN BE DISPLAYED USING THE DISPLAY_COMMAND_INFORMATION COMMAND. THIS IS DOCUMENTED IN THE NONSTANDARD COMMAND SECTION OF THIS DOCUMENT.

2.3 COMMAND_FUNCTIONS

FUNCTION	STATUS
$MOD	UNCHANGED
$CHAR	UNCHANGED
$CLOCK	UNCHANGED
$DATE	UNCHANGED
$FILE	UNCHANGED
$FNAME	UNCHANGED
$INTEGER	UNCHANGED
$NAME	UNCHANGED
$ORD	UNCHANGED
$REAL	UNCHANGED
$STRING	UNCHANGED
$STRLEN	UNCHANGED
$STRREP	UNCHANGED
$SUBSTR	UNCHANGED
$UNIQUE	UNCHANGED
$TIME	UNCHANGED
$VAR	UNCHANGED
$SPECIFIED	UNCHANGED
$SET_COUNT	UNCHANGED
$VALUE_COUNT	UNCHANGED
$RANGE	UNCHANGED
$PARAMETER_LIST	UNCHANGED
$PARAMETER	UNCHANGED
$STATUS	UNCHANGED
$CONDITION	UNCHANGED
$SEVERITY	UNCHANGED
$PROCESSOR	UNCHANGED
$JOB	UNCHANGED
$PROGRAM	UNCHANGED
$RING	NEW

2.4 SYSTEM_ACCESS_COMMANDS

COMMANDS	STATUS
SET_LINK_ATTRIBUTES	UNCHANGED
LOGIN	UNCHANGED - *1
LOGOUT	UNCHANGED
SET_PASSWORD	UNCHANGED

*1   THE FAMILY NAME OF THE JOB DOING THE SUBMIT WILL BE   USED   AS
     THE DEFAULT FAMILY NAME ON BATCH JOBS. THE DEFAULT FOR JOBS
     SUBMITTED FROM NOS WILL BE FAMILY $SYSTEM. THIS EFFECTIVELY
     MEANS THAT WHENEVER NOS/VE JOBS ARE SUBMITTED FROM NOS THE
     FAMILY PARAMETER IS REQUIRED.

2.5 RESOURCE_MANAGEMENT

COMMAND	STATUS
REQUEST_TERMINAL	UNCHANGED

2.6 FILE_MANAGEMENT

COMMAND	STATUS
SET_FILE_ATTRIBUTES	UNCHANGED
COPY_FILE	UNCHANGED
DISPLAY_FILE	UNCHANGED
COMPARE_FILE	UNCHANGED
DISPLAY_FILE_ATTRIBUTES	UNCHANGED
SKIP_TAPE	UNCHANGED

2.7 PERMANENT_FILE_MANAGEMENT

COMMAND	STATUS
GET_FILE	UNCHANGED
REPLACE_FILE	UNCHANGED
CREATE_FILE	UNCHANGED
ATTACH_FILE	UNCHANGED
DELETE_FILE	UNCHANGED
CHANGE_CATALOG_ENTRY	UNCHANGED
CREATE_FILE_PERMIT	UNCHANGED
DELETE_FILE_PERMIT	UNCHANGED
CREATE_CATALOG	UNCHANGED
DELETE_CATALOG	UNCHANGED
DELETE_CATALOG_PERMIT	UNCHANGED
CREATE_PERMIT_CATALOG	UNCHANGED
DISPLAY_CATALOG	UNCHANGED
DISPLAY_CATALOG_ENTRY	UNCHANGED
SET_WORKING_CATALOG	UNCHANGED

2.8 SCL_STATEMENTS_AND_PROCEDURES

COMMAND	STATUS
PROC/PROCEND	UNCHANGED
SET_COMMAND_LIST	UNCHANGED
DISPLAY_COMMAND_LIST	UNCHANGED
REPEAT/UNTIL	UNCHANGED
WHILE/WHILEND	UNCHANGED
CREATE_VARIABLE	UNCHANGED
DELETE_VARIABLE	UNCHANGED
BLOCK/BLOCKEND	UNCHANGED
LOOP/LOOPEND	UNCHANGED
FOR/FOREND	UNCHANGED

```
        IF/ELSEIF/ELSE/IFEND              UNCHANGED
        CYCLE                             UNCHANGED
        EXIT                              UNCHANGED
        WHEN/WHENEND                      UNCHANGED
        CONTINUE                          UNCHANGED
        CANCEL                            UNCHANGED
        INCLUDE_FILE                      UNCHANGED
        COLLECT_TEXT                      UNCHANGED
        DISPLAY_VALUE                     UNCHANGED
        EXIT_PROC                         UNCHANGED
        ACCEPT_LINE                       UNCHANGED
        INCLUDE_LINE                      UNCHANGED
        CREATE_FILE_CONNECTION            UNCHANGED
        DELETE_FILE_CONNECTION            UNCHANGED
        DISPLAY_FILE_CONNECTION           UNCHANGED
        CHANGE HCS VARIABLE               UNCHANGED
        DISPLAY HCS VARIABLE              UNCHANGED
```

2.9 INTERACTIVE_COMMANDS

COMMAND	STATUS
RESUME_COMMAND	UNCHANGED
TERMINATE_COMMAND	UNCHANGED
SET_TERMINAL_ATTRIBUTES	UNCHANGED
DISPLAY_TERMINAL_ATTRIBUTES	UNCHANGED
ESC-E	NEW - 1*
ESC-L	NEW - 1*
ESC-J	NEW - 1*
ESC-T	NEW - 1*
ESC-X	NEW - 1*

*1   THESE COMMANDS ARE ENTERED WITH THE 3-KEY SEQUENCE
     ESCAPE_KEY, CHARACTER, CARRIAGE_RETURN. THE CHARACTERS HAVE
     THE FOLLOWING MEANINGS

```
        E    PERFORM "DISPLAY_JOB_STATUS" COMMAND
        L    PERFORM "DISPLAY_LOG 10" COMMAND
        J    PERFORM "DISPLAY_JOB_STATUS ALL" COMMAND
        T    DISCARD ALL UNPROCESSED, TYPED-AHEAD INPUT
        X    TERMINATE JOB, BUT DO NOT DISCONNECT
```

2.10 OBJECT_CODE_MAINTENANCE

COMMAND	STATUS
CREATE_OBJECT_LIBRARY	UNCHANGED
DISPLAY_NEW_LIBRARY	UNCHANGED
SELECT_DISPLAY_OPTION	UNCHANGED
ADD_MODULE	UNCHANGED
REPLACE_MODULE	UNCHANGED
COMBINE_MODULE	UNCHANGED

```
CREATE_MODULE                      UNCHANGED
BIND_MODULE                        UNCHANGED
CREATE_PROGRAM_DESCRIPTION         UNCHANGED
DELETE_MODULE                      UNCHANGED
CHANGE_MODULE_ATTRIBUTE            UNCHANGED
SATISFY_EXTERNAL_REFERENCES        UNCHANGED
REORDER_MODULE                     UNCHANGED
GENERATE_LIBRARY                   UNCHANGED
DISPLAY_OBJECT_LIBRARY             UNCHANGED
COMPARE_OBJECT_LIBRARY             UNCHANGED
QUIT                               UNCHANGED
CI TO II CONVERSION                UNCHANGED
```

2.11 USER_SERVICES

COMMAND	STATUS
DISPLAY_LOG	UNCHANGED
DISPLAY_MESSAGE	UNCHANGED

2.12 FILE_ROUTING

COMMAND	STATUS
HCS JMROUTE	REMOVED

2.13 JOB_MANAGEMENT

COMMAND	STATUS
SUBMIT_JOB	UNCHANGED
DISPLAY_JOB_STATUS	UNCHANGED
TERMINATE_JOB	UNCHANGED
PRINT_FILE	UNCHANGED
TERMINATE_PRINT	UNCHANGED
DISPLAY_PRINT_STATUS	UNCHANGED

2.14 JOB_MANAGEMENT

COMMAND	STATUS
SUBMIT_JOB	UNCHANGED
DISPLAY_JOB_STATUS	UNCHANGED
TERMINATE_JOB	UNCHANGED
PRINT_FILE	UNCHANGED
TERMINATE_PRINT	UNCHANGED
DISPLAY_PRINT_STATUS	UNCHANGED

2.15 NON_STANDARD_COMMANDS

THE FOLLOWING COMMANDS PROVIDE A NONSTANDARD MEANS OF PERFORMING VARIOUS FREQUENTLY PERFORMED FUNCTIONS. THEY MAY BE

SUPERCEDED IN SUBSEQUENT BUILDS BY STANDARD COMMANDS AND CAPABILITIES.

2.15.1 DELETE_CATALOG_CONTENTS 1 DELCC

THE PURPOSE OF THIS COMMAND IS TO DELETE ALL ENTRIES FROM THE SPECIFIED CATALOG. THIS INCLUDES SUBCATALOGS AND THE FILES THEY CONTAIN.

DELETE_CATALOG_CONTENTS [CATALOG=<CATALOG>]
                        [STATUS=<STATUS VARIABLE>]

    CATALOG1C: THIS PARAMETER SPECIFIES FROM WHICH CATALOG ALL
                FILES ARE TO BE DELETED. OMISSION WILL CAUSE
                THE CURRENT WORKING CATALOG TO BE USED.

    STATUS: SEE ERROR HANDLING.

2.15.2 DISPLAY_ACTIVE_TASK 1 DISAT

THE PURPOSE OF THIS COMMAND IS TO DISPLAY TASK STATISTICS FOR ALL CURRENTLY ACTIVE TASKS IN A JOB. THE FOLLOWING INFORMATION IS DISPLAYED.

        TASK NAME
        EXECUTION TIME USE
        NUMBER OF PAGE FAULTS

DISPLAY_ACTIVE_TASK [OUTPUT=<FILE>]
                    [STATUS=<STATUS VARIABLE>]

    OUTPUT10:   THIS PARAMETER SPECIFIES THE FILE TO WHICH THE
                TASK STATISTICS IS DISPLAYED. OMISSION WILL
                CAUSE $OUTPUT TO BE USED.

2.15.3 DISPLAY_SYSTEM_DATA 1 DISSD

THE PURPOSE OF THIS COMMAND IS TO DISPLAY SYSTEM PAGE FAULT STATISTICS AND SYSTEM MONITOR REQUEST STATISTICS.

DISPLAY_SYSTEM_DATA [DISPLAY_OPTION=PAGE_FAULTS1PF
                    1MONITOR_REQUESTS1MR1ALL
                    [DISPLAY_FORMAT=INCREMENTAL1I1TOTAL1T]
                    [OUTPUT=<FILE>]
                    [STATUS=<STATUS VARIABLE>]

    DISPLAY_OPTION1DO: THIS PARAMETER SPECIFIES WHICH
                        STATISTICS ARE TO BE DISPLAYED. THE
                        FOLLOWING OPTIONS ARE ALLOWED

PAGE_FAULTS   - DISPLAY THE PAGE
                     FAULT STATISTICS.

MONITOR_REQUESTS   - DISPLAY THE SYSTEM
                     MONITOR   REQUEST
                     STATISTICS.

OMISSION   WILL CAUSE ALL TO BE
                     USED.

DISPLAY_FORMAT1DF: THIS PARAMETER SPECIFIES WHETHER A
                     DISPLAY OF THE ALL STATISTICS RECORDED
                     SO FAR (TOTAL) OR ONLY THOSE STATISTICS
                     RECORDED   SINCE   THE   LAST
                     DISPLAY_SYSTEM_DATA   COMMAND
                     (INCREMENTAL) SHOULD BE DISPLAYED.
                     OMISSION WILL CAUSE INCREMENTAL TO BE
                     USED.

OUTPUT1O: THIS PARAMETER SPECIFIES THE FILE TO
                     WHICH THE SYSTEM DATA WILL BE DISPLAYED.
                     OMISSION WILL CAUSE $OUTPUT TO BE USED.

STATUS: SEE ERROR HANDLING.

2.15.4 DISPLAY_JOB_DATA 1 DISJD

    THE PURPOSE OF THIS COMMAND IS TO DISPLAY THE FOLLOWING JOB
RELATED STATISTICS

          TIME IN JOB MODE
          TIME IN MONITOR MODE
          COUNT OF PAGE IN OPERATIONS
          RECLAIMED PAGES
          NEW PAGES ASSIGNED
          WORKING SET SIZE
          COUNT OF READY TASKS

DISPLAY_JOB_DATA [DISPLAY_OPTION=JOB_DATA]
                 [DISPLAY_FORMAT=INCREMENTAL1I1TOTAL1T]
                 [OUTPUT=<FILE>]
                 [STATUS=<STATUS VARIABLE>]

DISPLAY_OPTION1DO: THIS   PARAMETER   SPECIFIES   WHICH
                     STATISTICS ARE TO BE DISPLAYED. THE
                     FOLLOWING OPTIONS ARE ALLOWED

                     JOB_DATA - DISPLAY JOB RELATED DATA.

                     OMISSION WILL CAUSE JOB_DATA TO BE
                                USED.

DISPLAY_FORMAT1DF: THIS PARAMETER SPECIFIES   WHETHER   A
                      DISPLAY OF THE ALL STATISTICS RECORDED
                      SO FAR (TOTAL) OR ONLY THOSE STATISTICS
                      RECORDED SINCE THE LAST DISPLAY_JOB_DATA
                      COMMAND   (INCREMENTAL)   SHOULD   BE
                      DISPLAYED.   OMISSION   WILL   CAUSE
                      INCREMENTAL TO BE USED.

OUTPUT1O: THIS PARAMETER SPECIFIES THE FILE TO
                      WHICH THE JOB DATA WILL BE DISPLAYED.
                      OMISSION WILL CAUSE $OUTPUT TO BE USED.

STATUS: SEE ERROR HANDLING.

2.15.5 DISPLAY_COMMAND_INFORMATION 1 DISCI

   THE   PURPOSE OF THIS COMMAND IS TO DISPLAY CURRENT INFORMATION
ABOUT A NOS/VE COMMAND.   THE PARAMETER NAMES, ABBREVIATIONS,
ALLOWED VALUES AND KNOWN PROBLEMS FOR A COMMAND, AS SUPPORTED IN
THE CURRENT SYSTEM, CAN BE DETERMINED.   THIS IS A NONSTANDARD
COMMAND AND WILL BE REPLACED BY THE HELP UTILITY SOMETIME IN THE
FUTURE.

DISPLAY_COMMAND_INFORMATION COMMAND_NAME=<NAME>1ALL
        [UTILITY_NAME=CREATE_OBJECT_LIBRARY1
                  COL1SOURCE_CODE_UTILITY1SCU1SYSTEM]
        [DISPLAY_OPTION=PARAMETER_DESCRIPTION_TABLE1
                  PDT1NOTES1NAMES1HELP]
        [OUTPUT=<FILE REFERENCE>]
        [STATUS=<STATUS VARIABLE>]

COMMAND_NAME1CN: THIS PARAMETER SPECIFIES THE NAME OF THE
                   COMMAND ABOUT WHICH INFORMATION IS TO BE
                   DISPLAYED.

UTILITY_NAME1UN: THIS PARAMETER SPECIFIES WHICH UTILITY
                   THE COMMAND BELONGS TO.   OMISSION WILL
                   CAUSE SYSTEM TO BE USED.

DISPLAY_OPTION1DO: THIS PARAMETER SPECIFIES THE TYPE OF
                   DISPLAY BEING REQUESTED.   THE OPTIONS ARE

                   PARAMETER_DESCRIPTION_TABLE1PDT - SELECTS
                       A DISPLAY OF THE PARAMETER DESCRIP-
                       TION TABLE USED BY THE COMMAND WHEN
                       EXECUTED.

                   NOTES - SELECTS A DISPLAY OF ANY KNOWN
                       PROBLEMS WITH THE COMMAND.

                   NAMES - SELECTS A DISPLAY OF THE COMMAND
                       NAMES FOR A UTILITY.

HELP - SELECTS A DISPLAY OF THE COMMAND
INTERFACE DESCRIPTION OF THE COMMAND.

OMISSION WILL CAUSE PDT TO BE USED.

OUTPUT1O: THIS PARAMETER SPECIFIES THE FILE TO
WHICH INFORMATION WILL BE DISPLAYED.
OMISSION WILL CAUSE $OUTPUT TO BE USED.

STATUS: SEE ERROR HANDLING.

2.15.6 CONVERT_OBJECT_FILE 1 CONOF

THE PURPOSE OF THIS COMMAND IS TO GET A NOS/VE OBJECT FILE
PRODUCED ON NOS AND TO CONVERT IT TO AN OBJECT FILE SUITABLE FOR
PROCESSING BY THE NOS/VE LOADER OR OBJECT CODE MAINTENANCE
COMMANDS.

CONVERT_OBJECT_FILE TO=<FILE REFERENCE>
                    [FROM=<NAME>]
                    [USER=<NAME>]
                    [STATUS=<STATUS VARIABLE>]

   TO1T: THIS PARAMETER SPECIFIES THE NOS/VE FILE NAME ON
WHICH THE CONVERTED OBJECT FILE IS TO BE WRITTEN.

   FROM1F: THIS PARAMETER SPECIFIES THE NAME OF THE NOS FILE TO
BE CONVERTED. THIS IS THE PERMANENT FILE NAME AS
DEFINED IN THE NOS FILE SYSTEM AND CAN BE UP TO SEVEN
CHARACTERS IN LENGTH.

OMISSION WILL CAUSE THE PERMANENT FILE NAME OF THE
TO PARAMETER TO BE USED.

   USER1U: THIS PARAMETER SPECIFIES THE NOS USER IDENTIFICATION
OF THE OWNER OF THE FILE. THIS PARAMETER IS ONLY
NECCESSARY IF THE FILE IS IN A CATALOG OTHER THAN THE
USER WHO WAS SPECIFIED BY THE MOST RECENTLY ISSUED
SET_LINK_ATTRIBUTES COMMAND.

   STATUS: SEE ERROR HANDLING.

2.15.7 GET_OBJECT_FILE 1 GETOF

THE PURPOSE OF THIS COMMAND IS TO GET A PREVIOUSLY CONVERTED
NOS/VE OBJECT FILE FROM THE NOS SIDE AND SETS THE APPROPRIATE
FILE ATTRIBUTES THAT WILL ALLOW THE OBJECT FILE TO BE USED BY
NOS/VE.

GET_OBJECT_FILE TO=<FILE REFERENCE>

                    [FROM=<NAME>]
                    [USER=<NAME>]
                    [STATUS=<STATUS VARIABLE>]

    TO1T: THIS PARAMETER SPECIFIES THE NOS/VE FILE NAME OF THE
             OBJECT FILE.

   FROM1F: THIS PARAMETER SPECIFIES   THE NOS FILE NAME OF THE
             OBJECT FILE. THIS IS THE   PERMANENT FILE NAME AS
             DEFINED IN NOS AND CAN BE UP TO SEVEN CHARACTERS IN
             LENGTH.

             OMISSION WILL CAUSE THE PERMANENT FILE NAME OF THE
                    TO PARAMETER TO BE USED.

   USER1U: THIS PARAMETER SPECIFIES THE NOS USER IDENTIFICATION
             OF THE OWNER OF THE FILE.   THIS PARAMETER IS ONLY
             NECESSARY IF THE FILE IS IN A CATALOG OTHER THAN THE
             USER WHO WAS SPECIFIED BY THE MOST RECENTLY ISSUED
             SET_LINK_ATTRIBUTES COMMAND.

   STATUS: SEE ERROR HANDLING.

2.15.8 GET_OBJECT_LIBRARY 1 GETOL

THE   PURPOSE OF THIS   COMMAND IS TO GET A PREVIOUSLY CREATED
NOS/VE OBJECT LIBRARY FROM THE NOS SIDE AND SET THE APPROPRIATE
FILE ATTRIBUTES THAT WILL ALLOW THE OBJECT LIBRARY TO BE USED ON
NOS/VE.

GET_OBJECT_LIBRARY TO=<FILE REFERENCE>
                   [FROM=<NAME>]
                   [USER=<NAME>]
                   [STATUS=<STATUS VARIABLE>]

    TO1T: THIS PARAMETER SPECIFIES THE NOS/VE FILE NAME OF THE
             OBJECT LIBRARY.

   FROM1F: THIS PARAMETER SPECIFIES   THE NOS FILE NAME OF THE
             OBJECT FILE. THIS IS THE   PERMANENT FILE NAME AS
             DEFINED IN NOS AND CAN BE UP TO SEVEN CHARACTERS IN
             LENGTH.

             OMISSION WILL CAUSE THE PERMANENT FILE NAME OF THE
                    TO PARAMETER TO BE USED.

   USER1U: THIS PARAMETER SPECIFIES THE NOS USER IDENTIFICATION
             OF THE OWNER OF THE FILE.   THIS PARAMETER IS ONLY
             NECESSARY IF THE FILE IS IN A CATALOG OTHER THAN THE
             USER WHO WAS SPECIFIED ON THE MOST RECENTLY ISSUED
             SET_LINK_ATTRIBUTES COMMAND.

STATUS: SEE ERROR HANDLING.

2.15.9 DISPLAY_OBJECT_TEXT 1 DISOT

THE PURPOSE OF THIS COMMAND IS TO PRODUCE A FORMATTED DISPLAY
OF THE OBJECT TEXT CONTAINED IN AN OBJECT FILE OR OBJECT LIBRARY
PRODUCED ON NOS/VE.

```
DISPLAY_OBJECT_TEXT FILE=<FILE>
                    [OUTPUT=<FILE REFERENCE>]
                    [STATUS=<STATUS VARIABLE>]
```

FILE1F: THIS PARAMETER SPECIFIES THE OBJECT FILE OR OBJECT
          LIBRARY CONTAINING THE OBJECT TEXT TO BE LISTED.

OUTPUT1O: THIS PARAMETER SPECIFIES THE FILE TO WHICH THE
           DISPLAY IS TO BE WRITTEN.

           OMISSION WILL CAUSE THE FILE $OUTPUT TO BE USED.

STATUS: SEE ERROR HANDLING.

2.15.10 GET_SOURCE_LIBRARY 1 GETSL

THE PURPOSE OF THIS COMMAND IS TO GET A PREVIOUSLY CREATED SCU
SOURCE LIBRARY FROM THE NOS SIDE AND SET THE APPROPRIATE FILE
ATTRIBUTES THAT WILL ALLOW THE SOURCE LIBRARY TO BE USED ON
NOS/VE.

```
GET_SOURCE_LIBRARY TO=<FILE REFERENCE>
                    [FROM=<NAME>]
                    [USER=<NAME>]
                    [STATUS=<STATUS VARIABLE>]
```

TO1T: THIS PARAMETER SPECIFIES THE NOS/VE FILE NAME OF THE
          SOURCE LIBRARY.

FROM1F: THIS PARAMETER SPECIFIES THE NOS FILE NAME OF THE
          SOURCE LIBRARY.   THIS IS THE PERMANENT FILE NAME AS
          DEFINED IN NOS AND CAN BE UP TO SEVEN CHARACTERS IN
          LENGTH.

          OMISSION WILL CAUSE THE PERMANENT FILE NAME OF THE
          TO PARAMETER TO BE USED.

USER1U: THIS PARAMETER SPECIFIES THE NOS USER IDENTIFICATION
          OF THE OWNER OF THE FILE. THIS PARAMETER IS ONLY
          NECESSARY IF THE FILE IS IN A CATALOG OTHER THAN THE
          USER WHO WAS SPECIFIED ON THE MOST RECENTLY ISSUED
          SET_LINK_ATTRIBUTES COMMAND.

STATUS: SEE ERROR HANDLING.

2.15.11 EDIT_FILE 1 EDIF

THE PURPOSE OF EDIT_FILE IS TO INITIATE THE EXECUTION OF THE
SCU EDITOR ON A TEXT FILE. (FOR DETAILS SEE ARH3883.)

EDIT_FILE 1 EDIF - EDIT LINES ON A SOURCE FILE. (PROCEDURE FILE
NOT NECESSARILY IN ITS FINAL FORM)

PARAMETERS	DEFAULTS
FILE=FILE(SOURCE)	$REQUIRED
[RESULT=FILE(SOURCE)]	$VALUE(FILE)
[INPUT=FILE REFERENCE]	$COMMAND
[OUTPUT=FILE REFERENCE]	$OUTPUT
[STATUS]	--

2.15.12 JEDIT

THE PURPOSE OF THIS COMMAND IS TO INITIATE EXECUTION OF THE
JEDIT EDITOR BUILT BY JACK BOHNHOFF. ANYONE WANTING INFORMATION
ABOUT THE EDITOR SHOULD CONTACT JACK.

JEDIT FROM=<FILE>
     [STATUS=<STATUS VARIABLE>]

    FROM1F: THIS PARAMETER SPECIFIES THE FILE TO BE EDITED.
             THIS FILE IS REWRITTEN AFTER THE EDITOR TERMINATES.

    STATUS: %SEE ERROR HANDLING IN THE NOS/VE COMMAND
             INTERFACE.

2.15.13 DEBUG

THE PROTOTYPE R1 NOS/VE DEBUGGER IS NOW AVAILABLE. DETAILS ON
HOW TO USE THE DEBUGGER CAN BE FOUND IN THE "CYBER 180
INTERACTIVE DEBUG EXTERNAL REFERENCE SPECIFICATION AND USER'S
GUIDE", SUNNYVALE DCS NUMBER S4028.

2.15.14 SET_LINK_ATTRIBUTES 1 SETLA

THE SET_LINK_ATTRIBUTES COMMAND IS THE SAME AS DOCUMENTED IN
THE NOS/VE COMMAND INTERFACE WITH THE EXCEPTION THAT THE CHARGE
AND PROJECT PARAMETERS ARE OPTIONAL (AND IN FACT NOT USEFUL IN
THE CURRENT ENVIRONMENT SINCE WE DISABLE THAT FEATURE ON THE NOS
SIDE).

2.15.15 DISPLAY_VARIABLE_LIST 1 DISVL

THE PURPOSE OF THIS COMMAND IS TO DISPLAY ALL LOCAL VARIABLES.

DISPLAY_VARIABLE_LIST [OUTPUT=<FILE REFERENCE>]
                      [STATUS=<STATUS VARIABLE>]

OUTPUT1O: THIS PARAMETER SPECIFIES THE FILE TO WHICH THE
           INFORMATION WILL BE DISPLAYED.

           OMISSION WILL CAUSE $OUTPUT TO BE USED.

STATUS: SEE ERROR HANDLING.

3.0 PROGRAM_INTERFACE_STATUS

THE 'STATUS' COLUMN INDICATES WHETHER THE PROCEDURE IS UNCHANGED FROM THE PREVIOUS BUILD, MODIFIED FROM THE PREVIOUS BUILD OR NOT AVAILABLE IN THIS BUILD. FOOTNOTES ARE NUMBERED WITHIN EACH SECTION.

3.1 COMMAND_PROCESSING

PROCEDURE	STATUS
CLP$SCAN_PARAM_LIST	UNCHANGED
CLP$TEST_PARAMETER	UNCHANGED
CLP$GET_KEYWORD	UNCHANGED
CLP$GET_SET_COUNT	UNCHANGED
CLP$GET_VALUE_COUNT	UNCHANGED
CLP$TEST_RANGE	UNCHANGED
CLP$GET_VALUE	UNCHANGED
CLP$CREATE_VARIABLE	UNCHANGED
CLP$DELETE_VARIABLE	UNCHANGED
CLP$READ_VARIABLE	UNCHANGED
CLP$WRITE_VARIABLE	UNCHANGED
CLP$SCAN_COMMAND_FILE	UNCHANGED
CLP$END_SCAN_COMMAND_FILE	UNCHANGED
CLP$SCAN_COMMAND_LINE	UNCHANGED
CLP$CREATE_FILE_CONNECTION	UNCHANGED
CLP$DELETE_FILE_CONNECTION	UNCHANGED
CLP$PUSH/POP_UTILITY	UNCHANGED
CLP$GET_COMMAND_ORIGIN	UNCHANGED
CLP$GET_DATA_LINE	UNCHANGED
CLP$SCAN_PROC_DECLARATION	UNCHANGED

3.2 MESSAGE_GENERATOR

PROCEDURE	STATUS
OSP$FORMAT_MESSAGE	UNCHANGED
OSP$SET_STATUS_ABNORMAL	UNCHANGED
OSP$APPEND_STATUS_PARAMETER	UNCHANGED
OSP$APPEND_STATUS_INTEGER	UNCHANGED

3.3 RESOURCE_MANAGEMENT

PROCEDURE	STATUS
RMP$REQUEST_MASS_STORAGE	UNCHANGED
RMP$REQUEST_TERMINAL	UNCHANGED

ALL TERMINAL ATTRIBUTES CAN BE SPECIFIED ON THE RMP$REQUEST_TERMINAL CALL BUT ONLY THE FOLLOWING ARE OPERATIONAL

    O AUTO_INPUT
    O TRANSPARENT_MODE
    O PROMPT_FILE
    O PROMPT_STRING

    FILES ASSIGNED TO A TERMINAL DEVICE CAN BE ACCESSED VIA THE
FOLLOWING BAM REQUESTS

    O AMP$OPEN
    O AMP$GET_NEXT
    O AMP$GET_DIRECT
    O AMP$GET_PARTIAL
    O AMP$PUT_NEXT
    O AMP$PUT_DIRECT
    O AMP$PUT_PARTIAL
    O AMP$CLOSE
    O AMP$REWIND
    O AMP$SKIP
    O AMP$SEEK_DIRECT

3.4 PROGRAM_EXECUTION

PROCEDURE	STATUS
PMP$EXIT	UNCHANGED
PMP$EXECUTE	UNCHANGED
PMP$TERMINATE	UNCHANGED
PMP$AWAIT_TASK_TERMINATION	UNCHANGED
PMP$MODULE_TABLE_ADDRESS	UNCHANGED
PMP$ENTRY_POINT_TABLE_ADDRESS	UNCHANGED
PMP$PUSH_TASK_DEBUG_MODE	UNCHANGED
PMP$SET_TASK_DEBUG_MODE	UNCHANGED
PMP$TASK_DEBUG_MODE_ON	UNCHANGED
PMP$SET_DEBUG_RING	UNCHANGED
PMP$DEBUG_RING	UNCHANGED
PMP$CHANGE_DEBUG_LIBRARY_LIST	UNCHANGED
PMP$POP_TASK_DEBUG_MODE	UNCHANGED

3.5 PROGRAM_COMMUNICATION

PROCEDURE	STATUS
OSP$AWAIT_ACTIVITY_COMPLETION	UNCHANGED
PMP$DEFINE_QUEUE	UNCHANGED
PMP$REMOVE_QUEUE	UNCHANGED
PMP$CONNECT_QUEUE	UNCHANGED
PMP$DISCONNECT_QUEUE	UNCHANGED
PMP$SEND_TO_QUEUE	UNCHANGED
PMP$RECEIVE_FROM_QUEUE	UNCHANGED
PMP$STATUS_QUEUE	UNCHANGED
PMP$STATUS_QUEUES_DEFINED	UNCHANGED
PMP$GET_QUEUE_LIMITS	UNCHANGED

3.6 CONDITION_PROCESSING

PROCEDURE	STATUS
PMP$ESTABLISH_CONDITION_HANDLER	ADDED SUPPORT OF DETECTED UNCORRECTED ERROR
PMP$DISESTABLISH_COND_HANDLER	UNCHANGED
PMP$CAUSE_CONDITION	UNCHANGED
PMP$CONTINUE_TO_CAUSE	UNCHANGED
PMP$TEST_CONDITION_HANDLER	UNCHANGED
PMP$VALIDATE_PREVIOUS_SAVE_AREA	UNCHANGED
PMP$ESTABLISH_DEBUG_OFF	UNCHANGED
OSP$SET_STATUS_FROM_CONDITION	UNCHANGED

3.7 PROGRAM_SERVICES

PROCEDURE	STATUS
PMP$GENERATE_UNIQUE_NAME	UNCHANGED
PMP$GET_TIME	UNCHANGED
PMP$GET_MICROSECOND_CLOCK	UNCHANGED
PMP$GET_TASK_CP_TIME	UNCHANGED
PMP$GET_DATE	UNCHANGED
PMP$GET_USER_IDENTIFICATON	UNCHANGED
PMP$GET_ACCOUNT_PROJECT	UNCHANGED
PMP$GET_JOB_NAMES	UNCHANGED
PMP$GET_JOB_ID	UNCHANGED
PMP$GET_JOB_MODE	UNCHANGED
PMP$GET_PROGRAM	UNCHANGED
PMP$GET_TASK_ID	UNCHANGED
PMP$MANAGE_SENSE_SWITCHES	UNCHANGED
PMP$GET_OS_VERSION	UNCHANGED
PMP$GET_PROCESSOR_ATTRIBUTES	UNCHANGED
PMP$DEFINE_DEBUG_ENTRY	UNCHANGED
PMP$GET_DEBUG_ENTRY	UNCHANGED
PMP$MODIFY_DEBUG_ENTRY	UNCHANGED
PMP$REMOVE_DEBUG_ENTRY	UNCHANGED

3.8 LOGGING

PROCEDURE	STATUS
PMP$LOG	UNCHANGED
PMP$LOG_ASCII	UNCHANGED

3.9 FILE_MANAGEMENT

PROCEDURE	STATUS
SEQUENTIAL ACCESS	UNCHANGED
BYTE_ADDRESSABLE ACCESS	UNCHANGED
RECORD ACCESS	UNCHANGED
SEGMENT ACCESS	UNCHANGED - *1

```
       V_SYSTEM SPECIFIED                    UNCHANGED
       V_USER  SPECIFIED                     UNCHANGED
       U_SYSTEM SPECIFIED                    UNCHANGED
       U_USER  SPECIFIED                     UNCHANGED
       F_SYSTEM SPECIFIED                    UNCHANGED
       F_USER  SPECIFIED                     UNCHANGED
       AMP$DESCRIBE_NEW_FILE                 DELETED
       AMP$FILE                              UNCHANGED
       AMP$GET_FILE_ATTRIBUTES               UNCHANGED
       AMP$FETCH                             UNCHANGED
       AMP$STORE                             UNCHANGED
       AMP$COPY_FILE                         UNCHANGED
       AMP$RENAME                            UNCHANGED
       AMP$RETURN_FILE                       NEW NAME
       AMP$OPEN                              UNCHANGED
       AMP$CLOSE                             UNCHANGED
       AMP$FETCH_ACCESS_INFORMATION          UNCHANGED
       AMP$SKIP                              UNCHANGED
       AMP$REWIND                            *2
       AMP$WRITE_END_PARTITION               UNCHANGED
       AMP$GET_NEXT                          UNCHANGED
       AMP$GET_DIRECT                        UNCHANGED
       AMP$GET_PARTIAL                       UNCHANGED
       AMP$PUT_NEXT                          UNCHANGED
       AMP$PUT_DIRECT                        UNCHANGED
       AMP$PUT_PARTIAL                       UNCHANGED - *3
       AMP$SEEK_DIRECT                       UNCHANGED
       AMP$GET_SEGMENT_POINTER               UNCHANGED
       AMP$SET_SEGMENT_EOI                   UNCHANGED
       AMP$SET_SEGMENT_POSITION              UNCHANGED
       AMP$SET_LOCAL_NAME_ABNORMAL           UNCHANGED
       AMP$SET_FILE_INSTANCE_ABNORMAL        UNCHANGED
       AMP$ACCESS_METHOD                     UNCHANGED
       AMP$FETCH_FAP_POINTER                 UNCHANGED
       AMP$STORE_FAP_POINTER                 UNCHANGED
```

*1 SEGMENT ACCESS IF A SEGMENT ACCESS FILE IS WRITTEN AND AN AMP$SET_SEGMENT_EOI IS NOT ISSUED TO RECORD THE EOI, EOI REMAINS ZERO. THE HIGHEST PAGE REFERENCED IS NOT YET USED AS THE DEFAULT EOI. THIS PARTICULARLY AFFECTS THOSE WHO WISH TO MAKE HEAPS PERMANENT BECAUSE EOI IS ALWAYS ZERO FOR A HEAP.

*2 AMP$REWIND THE WAIT PARAMETER ON THE PROCEDURE CALL IS NOT SUPPORTED.

*3 AMP$PUT_PARTIAL PUT_PARTIAL WITH THE TERM_OPTION = AMC$TERMINATE DOES NOT ACT AS A PUT_NEXT IF A PRECEDING START WAS NOT ISSUED.

3.10 PERMANENT_FILE_MANAGEMENT

PROCEDURE	STATUS
PFP$DEFINE	UNCHANGED
PFP$ATTACH	UNCHANGED
PFP$PURGE	UNCHANGED
PFP$CHANGE	UNCHANGED
PFP$PERMIT	UNCHANGED
PFP$DELETE_PERMIT	UNCHANGED
PFP$DEFINE_CATALOG	UNCHANGED
PFP$PURGE_CATALOG	UNCHANGED
PFP$PERMIT_CATALOG	UNCHANGED
PFP$DELETE_CATALOG_PERMIT	UNCHANGED

3.11 MEMORY_MANAGEMENT

MMP$ADVISE_IN	UNCHANGED
MMP$ADVISE_OUT	UNCHANGED
MMP$ADVISE_OUT_IN	UNCHANGED
MMP$WRITE_MODIFIED_PAGES	UNCHANGED
MMP$CREATE_SEGMENT	UNCHANGED
MMP$DELETE_SEGMENT	UNCHANGED
MMP$STORE_SEGMENT_ATTRIBUTES	UNCHANGED
MMP$FETCH_SEGMENT_ATTRIBUTES	UNCHANGED
MMP$VERIFY_ACCESS	UNCHANGED
MMP$FREE	UNCHANGED
MMP$LOCK_PAGES	NUMBER OF LOCKED PAGES PER
MMP$UNLOCK_PAGES	SEGMENT RESTRICTED TO 32
MMP$FETCH_PVA_UNWRITTEN_PAGES	UNCHANGED

3.12 STATISTICS_FACILITY

SFP$ESTABLISH_STATISTIC	UNCHANGED
SFP$ENABLE_STATISTIC	UNCHANGED
SFP$DISABLE_STATISTIC	UNCHANGED
SFP$DISESTABLISH_STATISTIC	UNCHANGED
SFP$EMIT_STATISTIC	UNCHANGED
SFP$EMIT_SYSTEM_STATISTIC	UNCHANGED

3.13 INTERACTIVE_FACILITY

IFP$TERMINAL	UNCHANGED
IFP$FETCH_TERMINAL	UNCHANGED
IFP$STORE_TERMINAL	UNCHANGED
IFP$GET_DEFLT_TERMINAL_ATTRIBUTES	UNCHANGED
IFP$GET_TERMINAL_ATTRIBUTES	UNCHANGED
IFP$ADVANCE	NEW - *1

*1 ONLY THE OPTION IFC$ADVANCE_ALL_QUEUED_OUTPUT IS SUPPORTED.

3.14 NOS/VE_EXCEPTIONS

THE FOLLOWING SUMMARIZES THE EXCEPTION CODE RANGES CURRENTLY ASSIGNED TO NOS/VE. THESE CODE RANGES REPRESENT A FINER BREAKDOWN THAN THE ONE SPECIFIED IN THE SIS FOR INTERNAL NOS/VE DEVELOPMENT PURPOSES. HOWEVER, IT IS IMPORTANT TO REMEMBER THAT ONLY THE PRODUCT IDENTIFIERS DOCUMENTED IN THE SIS MAY APPEAR IN ERROR MESSAGES.

```
COMMON MODULES                     9,000 - 9,999
COMMON CODE GENERATOR              8,000 - 8,999
```

EXCEPTION CODE	PRODUCT IDENTIFIER	PRODUCT NAME
1 - 158,999	RESERVED	
159,000 - 159,999	SY	SYSTEM CORE
160,000 - 169,999	AM	BASIC ACCESS METHODS
160,000 - 163,999	BA	BASIC ACCESS
164,000 - 164,999	LN	LOCAL NAME MGR
165,000 - 165,999	JF	JOB FILE MGR
166,000 - 166,999	SR	CONVERSION SERVICES
170,000 - 179,999	CL	COMMAND LANGUAGE
180,000 - 189,999	JM	JOB MANAGEMENT
190,000 - 199,999	LL	LOADER
200,000 - 209,999	MM	MEMORY MANAGEMENT
200,000 - 204,999	MM	MONITOR LEVEL
205,000 - 205,999	MM	TASK LEVEL
210,000 - 219,999	OS	OPERATING SYSTEM
210,000 - 210,999	OS	OS
211,000 - 211,999	MT	EXEC
212,000 - 212,999	IO	MS I/O
213,000 - 213,999	IO	TAPE I/O
214,000 - 214,999	DM	DEVICE MANAGEMENT
215,000 - 215,999	ML	MEMORY LINK
216,000 - 216,999	IF	INTERACTIVE
217,000 - 217,999	TM	TM MONITOR
218,000 - 218,999	TM	TM TASK
219,000 - 219,999	JS	JOB SWAPPERS
220,000 - 229,999	PF	PERMANENT FILE MANAGEMENT
221,000 - 221,999	ST	SET MANAGEMENT
222,000 - 222,999	PU	PERMANENT FILE UTILITIES
230,000 - 239,999	PM	PROGRAM MANAGEMENT
240,000 - 249,999	RM	RESOURCE MANAGEMENT
250,000 - 259,999	OF	OPERATOR FACILITY
260,000 - 269,999	AV	USER ADMINISTRATOR
270,000 - 279,999	IC	INTERSTATE COMMUNICATION
280,000 - 289,999	RH	REMOTE HOST FACILITY
290,000 - 299,999	OC	OBJECT CODE UTILITIES
300,000 - 309,999	DS	DEADSTART/RECOVERY
310,000 - 319,999	MS	MAINTENANCE SERVICES
320,000 - 329,999	IF	INTERACTIVE FACILITY
330,000 - 339,999	US	USER ERRORS
340,000 - 349,999	SF	STATISTICS FAC.
350,000 - 359,999	CM	CONFIGURATION MANAGEMENT
360,000 - 369,999	HU	HELP UTILITIES
370,000 - 379,999	NA	NETWORK ACCESS METHOD
500,000 - 509,999	AA	ADVANCED ACCESS METHOD
510,000 - 519,999	AG	ALGOL
520,000 - 529,999	AL	ASSEMBLY LANGUAGE
530,000 - 539,999	AP	APL
540,000 - 549,999	BA	BASIC
550,000 - 559,999	CA	CONVERSION AIDS SYSTEM
560,000 - 569,999	CB	COBOL

570,000 - 579,999	CY	CYBIL
580,000 - 589,999	FT	FORTRAN
590,000 - 599,999	PA	PASCAL (WIRTH)
600,000 - 609,999	P1	PL/1
610,000 - 619,999	SM	SORT MERGE
620,000 - 629,999	SC	SOURCE CODE UTILITY
640,000 - 649,999	DB	DEBUG

4.0 DUAL_STATE_DEADSTART_AND_OPERATION

4.1 CURRENT_DUAL_STATE_CONFIGURATION

THE ARDEN HILLS S2 DEVELOPMENT LAB CONTAINS 12 FMD UNITS.

O FMD UNIT 43 (CH. 1, 21, 22)

   THIS UNIT CONTAINS NOS PERMANENT FILES.

O FMD UNIT 41 (CH. 1, 21, 22)

   THIS UNIT CONTAINS THE FOLLOWING

   - FILES REQUIRED TO DEADSTART DUAL STATE CYCLE 11.1; A170 NOS
      (6.1 PLUS CHANGES NECESSARY FOR CYCLE 11.1), CTI, MSL, EI
      BINARIES, AND NOS DEADSTART FILES.
   - IT IS ALSO USED AS A TEMP DEVICE.

O FMD UNIT 42 (CH. 1, 21, 22)

   THIS UNIT CONTAINS NOS PERMANENT FILES.

O FMD UNIT 44 (CH. 1)

   THIS IS ANOTHER NOS PF DEVICE.

O FMD UNIT 45 (CH. 1)

   THIS UNIT CONTAINS FILES REQUIRED TO DEADSTART DUAL STATE
   CYCLE 11.2 AND LATER; NOS 5G41 DEADSTART FILE, CTI, MSL, AND
   EI. IT IS ALSO A NOS TEMP DEVICE.

   THE FOLLOWING DEVICES ARE AVAILABLE FOR HANDS-ON USE: CHANNEL
   1, 21, 22: UNIT 40; CHANNEL 2, 22, 31: UNITS 44 AND 45; CHANNEL
   2: UNITS 40, 41, 42, AND 43.

4.2 USER_NAMES_AND_PERMANENT_FILES

1) THE CONVENTION USED FOR CREATING USER NAMES ON NOS/VE IS AS
   FOLLOWS
   O YOUR USER NAME WILL BE YOUR INITIALS.
   O YOUR PASSWORD WILL BE THESE 3 LETTERS FOLLOWED BY THE
      LETTER 'X'.
   O YOU MUST SEE COMSOURCE (R.K. COOPER — X3092) TO BE
      ASSIGNED A USER INDEX

2) PF DUMPING AND LOADING

                                        CYCLE 11, JANUARY 1983

YOU MAY USE "SES.DUMPPF" ON SN/101 TO DUMP YOUR PERMANENT FILES TO TAPE, AND THEN LOAD THEM ONTO YOUR USER NAME ON A170 NOS USING "SES.LOADPF". DOCUMENTATION ON HOW TO USE THESE SES PROCEDURES AND WHAT THEIR PARAMETERS ARE IS INCLUDED IN THE SES USER'S GUIDE, OR THEY CAN BE OBTAINED BY TYPING

SES,HELP.DUMPPF AND SES,HELP.LOADPF.

4.3 CONVERTING_VERSION_1_VALIDATION_FILES_TO_VERSION_2

THE BASIC REFERENCE FOR THIS SECTION IS THE NOS V2 SYSTEM MAINTENANCE MANUAL (60459300), SECTION 5. THE BEST SPECIFIC REFERENCE IS EXAMPLE 13 ON PAGE 5-37 IN THAT MANUAL.

USING NOS 5.3, THE ANALYST SHOULD DO THE FOLLOWING

```
X.DIS.
SUI,377777.
PURGE(SOURCE/NA)
DEFINE(SOURCE)
MODVAL(OP=S,S=SOURCE,FA)
DROP.
```

NEXT, THE FILE SOURCE SHOULD BE XEDITED WITH THE FOLLOWING DIRECTIVES

```
D/AB...=/*
D/NF...=/*
D/OF...=/*
C/STANDARD/NORMAL/*
C/CMLI/CNVE/*
```

FINALLY, AFTER BRINGING UP A NOS V2 SYSTEM, DO THE FOLLOWING (NOTE: AN IDLE FAMILY SITUATION MUST EXIST WHEN THE ISF COMMANDS ARE GIVEN.)

```
X.DIS.
SUI,377777.
PURGE,NEWDUZ,NEWNDZ.
DEFINE,NEWDUZ,NEWNDZ.
ATTACH,SOURCE.
MODVAL(OP=C,I=SOURCE,SI,N=NEWDUZ,U=NEWNDZ)
RETURN(NEWDUZ,NEWNDZ)
ISF,R=NEWDUZ.    (WILL RELEASE NEWDUZ IF IT IS FAST-ATTACH)
ISF,E=NEWDUZ.
DROP.
```

4.4 TO RELOAD CONTROLWARE FOR THE NOS/VE DISK DRIVER

AT DEADSTART TIME NOS WILL AUTOMATICALLY LOAD 7155-1X DISK CONTROLWARE ON ONE CHANNEL WITH CONTROLLER TYPE=FM (LBC CMRDECK ENTRY), AND WILL AUTOMATICALLY LOAD 7155-4X DISK CONTROLWARE ON ANY CHANNEL WITH CONTROLLER TYPE=HT (LBC CMRDECK ENTRY). NOS/VE SUPPORTS BOTH OF THOSE TYPES OF CONTROLLERS. NOTE: IT IS NOT POSSIBLE TO USE 844 HALF-TRACK CONTROLWARE IN THIS ENVIRONMENT.

4.5 A170 NOS DEADSTART

4.5.1 CTI AND CHECKING CENTRAL MEMORY

DEADSTARTING A170 NOS ASSUMES SOME KNOWLEDGE OF CTI. CTI STANDS FOR COMMON TEST AND INITIALIZATION. IT IS SOFTWARE THAT PLACES AN 800 SERIES MACHINE IN A STATE SUCH THAT IT IS POSSIBLE TO DEADSTART AN OPERATING SYSTEM. CTI IS USED SOMEWHAT AMBIGUOUSLY IN THE SOFTWARE COMMUNITY TO IMPLY CTI AND MSL (MAINTENANCE SOFTWARE LIBRARY). THE MSL IS A COLLECTION OF PROGRAMS AND DATA THAT INCLUDES SUCH THINGS AS A SUBSET OF CMSE (CYBER MAINTENANCE SOFTWARE EXECUTIVE) THAT ENABLES ONE TO LOAD CONTROLWARE TO CONTROLLERS, LOOK AT CYBER 180 MAINTENANCE REGISTERS, LOOK AT MICROCODE, ETC. THE MSL ALSO CONTAINS MICROCODE THAT CAN BE LOADED BY CTI. THE MSL IS ACTUALLY AN OPERATING SYSTEM THAT RUNS INDEPENDENTLY OF NOS. AN IMPORTANT ELEMENT OF CTI/MSL IS HIVS (HARDWARE VERIFICATION SEQUENCE), WHICH IS A PROGRAM THAT LOADS MICROCODE, CLEARS AND CHECKS CENTRAL MEMORY AND TESTS ALL 170 OPCODES. IF YOU ARE NOT SURE WHAT THE MACHINE WAS USED FOR (PARTICULARLY THE FIRST HANDS ON USER EACH MORNING) THEN THE HIVS PROGRAM SHOULD BE RUN. THIS IS ACCOMPLISHED BY

1)   DEADSTART TO NOS/VE (UNIT 40 FOR S3, UNIT 45 FOR S2)

2)   ENTER O (OPERATOR INTERVENTION)

3)   ENTER P (DEADSTART PANEL,MAKE SURE LEVEL 0 DEADSTART)

4)   <BKSP>

5)   ENTER H (ASSURE YOURSELF THAT CS=YES TO RELOAD MICROCODE)

6)   <BKSP>

7)   ENTER V (VERIFICATION SEQUENCE)

8)   HIT <CR> AT 'PARAMETER DISPLAY' TO TEST CM & CP

WHEN YOU SEE TEXT THAT TELLS YOU THAT VERIFICATION IS COMPLETE AND A DEADSTART IS REQUIRED, YOU ARE NOW READY TO DEADSTART NOS.

4.5.2 NOS DEADSTART

O SET THE D/S PANEL TO DEADSTART FROM THE PRIMARY SYSTEM DISK.
   THIS IS UNIT 41 FOR 11.1 AND UNIT 45 FOR 11.2 AND SUBSEQUENT
   SYSTEMS.
O PUSH D/S BUTTON
O ENTER (CR)
O ENTER DATE/TIME

WAIT FOR DEADSTART TO COMPLETE.

NOTE: THE DEADSTART TAPES ARE FOUND IN THE TAPE CABINET IN THE
NORTHEAST CORNER OF THE S2 LAB.

4.6 NOS/VE DEADSTART AND INSTALLATION

O ENTER DOWN,CH2. SO NOS/VE CAN USE THE CHANNEL.

O ENTER DOWN,CH32. SO NOS/VE CAN USE THE CHANNEL.

O THE FOLLOWING FILE MUST BE AVAILABLE IN YOUR CATALOG ON THE
   S2

TPXXXK CONTAINS A NOS/VE DEADSTART IMAGE. THIS MUST BE A COPY OF
THE DUAL STATE DEADSTART IMAGES AVAILABLE FROM THE LINK
PROCEDURES.

O IF YOU'VE NEVER DEADSTARTED NOS/VE FROM THE USER NUMBER FROM
   WHICH YOU WANT TO RUN OR IF YOU WISH TO CHANGE THE CURRENT
   PARAMETER SETTINGS FOR YOUR PARTICULAR USER NUMBER, THEN DO A
   SETVE. SETVE ASSUMES THE FILE TPXXXK IS IN YOUR USER NUMBER;
   YOU DO NOT HAVE TO DO ANOTHER SETVE IF TPXXXK HAS CHANGED
   SINCE THE LAST TIME YOU RAN. THE GENERAL FORM OF SETVE IS

        X.SETVE(PN=FFFF,UN=UN,C=6)

   WHERE FFFF IS AN IDENTIFIER OF UP TO 4 CHARACTERS AND UN IS
   THE USER NUMBER TO SEARCH FIRST FOR FILES. 6 IS THE NUMBER OF
   THE SYSTEM CORE COMMAND DECK FOR THE ARDEN HILLS S2
   CONFIGURATION. WARNING: SPECIFYING C=6 ON THE ARDEN HILLS S3
   WILL DESTROY THE CLOSED SHOP PERMANENT FILE BASE. IN GENERAL
   FFFF AND UN WILL BE THE SAME, E.G. X.SETVE(DAH,UN=DAH,C=6)

   **ONLY ONE SETVE SHOULD BE DONE FOR EACH USER NUMBER AND A
   SETVE SHOULD NOT BE DONE FOR ANY INTEGRATION USER NUMBER
   EXCEPT BY THE INTEGRATION PROJECT.**

   SEE SECTION 5.1 FOR MORE DETAILS.

O BRING UP DUAL STATE

NVEFFFF.

WHERE FFFF IS THE IDENTIFIER SPECIFIED IN SETVE, E.G. NVEINT1.

O BRING UP THE OPERATOR FACILITY

ENTER K,NVE.

NOS/VE IS CURRENTLY GENERATED AND INITIALIZED ON BOTH NOS AND NOS/VE. ALL SOURCE AND OBJECT LIBRARIES THAT MAKE UP THE NOS/VE SYSTEM ARE PRODUCED ON NOS AND THEREFORE MUST BE CONVERTED FROM THEIR CI TO II COUNTERPARTS. OTHER PARTS OF INSTALLING AND INITIALIZING THE SYSTEM (E.G. BUILDING THE $SYSTEM CATALOG) ARE PERFORMED BY COMMAND LANGUAGE PROCEDURES ON NOS/VE. SINCE THE SAME SYSTEM WILL BE DEADSTARTED MANY TIMES IN A CLOSED SHOP ENVIRONMENT, IT IS ADVANTAGEOUS TO ONLY PERFORM THE CONVERSION FROM CI TO II A SINGLE TIME; SAVE THE RESULTS IN THE NOS FILE SYSTEM AND THEN SIMPLY BRING THE FILES BACK DURING DEADSTART.

THE ACTUAL FILES THAT GET INSTALLED AND LOADED ON EACH DEADSTART ARE DETERMINED BY A COMMAND LANGUAGE PROCEDURE (THE SYSTEM PROFILE) INTERPRETED ON NOS/VE. THIS PROCEDURE CAN BE MODIFIED BY EACH SITE TO INITIALIZE THEIR NOS/VE ENVIRONMENT IN THE MOST SUITABLE FASHION. THE PROCESS OF BUILDING THE SYSTEM PROFILE AND OF PERFORMING THE CI TO II CONVERSIONS IS REFERRED TO AS AN INSTALLATION DEADSTART AND THE PROCESS OF EXECUTING THE SYSTEM PROFILE AND OF FETCHING PREVIOUSLY CONVERTED FILES FROM NOS AND MAKING THEM AVAILABLE IN THE NOS/VE FILE SYSTEM IS REFERRED TO AS A DEADSTART. A SINGLE COMMAND IS AVAILABLE TO PERFORM BOTH AN INSTALLATION DEADSTART AND A DEADSTART.

4.6.1 THE DS PROCEDURE AND PRODUCT INSTALLATION

4.6.1.1 THE_DS_PROCEDURE

THE PURPOSE OF THIS COMMAND IS TO PERFORM AN INSTALLATION, NORMAL OR RECOVERY DEADSTART OF NOS/VE. THE DEFAULTS FOR PARAMETERS ARE THOSE MOST CONVENIENT FOR "CLOSED SHOP" DEADSTARTS.

THE PROCEDURE "BRINGS UP" THE JOB LOG DISPLAY ON THE LEFT SCREEN WHERE THE PROGRESS OF THE PROCEDURE MAY BE WATCHED, AND THE CONTROL POINT DISPLAY ON THE RIGHT SCREEN. JUST BEFORE THE PROCEDURE COMPLETES IT CHANGES THE LEFT SCREEN TO DISPLAY THE SYSTEM LOG AND WRITES TO THAT LOG THE MESSAGE

'----- DEADSTART COMPLETED ----'

AT WHICH POINT THE OPERATOR MAY ENTER COMMANDS.

```
DS [KIND=INSTALL 1 NORMAL 1 RECOVER]
   [GET_PRODUCTS=<BOOLEAN>]
   [ECHO=<BOOLEAN>]
   [ALTERNATE_USER=<NOS_USER_NAME>]
   [SAVE_INSTALL_FILES=<BOOLEAN>]
   [VALIDATE_USERS=<BOOLEAN>]
   [QUICK_VALIDATE=LIST OF <NAME>]
   [STATUS=<STATUS VARIABLE>]
```

KIND 1 K: THIS PARAMETER SPECIFIES WHAT KIND OF DEADSTART IS TO BE PERFORMED. VALID SPECIFICATIONS ARE

INSTALL 1 I - INSTALLATION DEADSTART TO BE PERFORMED. THE SYSTEM LIBRARIES ARE BUILT FROM CI OBJECT FILES.

NORMAL 1 N - NORMAL DEADSTART TO BE PERFORMED. THE SYSTEM LIBRARIES ARE OBTAINED FROM THE RESULTS OF A PREVIOUS INSTALLATION DEADSTART.

RECOVER 1 R - RECOVERY DEADSTART. JUST INITIATES SYSTEM TASKS. PERMANENT FILES ARE "RECOVERED" FROM A PREVIOUS RUN OF THE SYSTEM.

OMISSION WILL CAUSE A RECOVERY DEADSTART TO BE PERFORMED.

GET_PRODUCTS 1 GP: THIS PARAMETER SPECIFIES WHETHER THE OBJECT LIBRARIES DEFINING THE CURRENT PRODUCT SET AND SCU LIBRARIES DEFINING THE SOURCE LIBRARIES ARE TO BE INSTALLED. VALID SPECIFICATIONS ARE

TRUE 1 YES 1 ON - THE PRODUCTS ARE TO BE INSTALLED

FALSE 1 NO 1 OFF - THE PRODUCTS ARE NOT TO BE INSTALLED

THE LIST OF PRODUCTS INSTALLED ON THE ARDEN HILLS CLOSED SHOP S3 SYSTEM WHEN GET_PRODUCTS=YES IS SPECIFIED IS GIVEN IN THE INSTALL_PRODUCTS SECTION.

OMISSION WILL CAUSE THE PRODUCT SET TO BE INSTALLED.

ECHO 1 E: THIS PARAMETER SPECIFIES WHETHER THE COMMANDS SHOULD BE ECHOED TO THE CONSOLE DURING EXECUTION. VALID SPECIFICATIONS ARE

TRUE 1 YES 1 ON - ECHO COMMANDS

FALSE 1 NO 1 OFF - DO NOT ECHO COMMANDS

OMISSION WILL CAUSE COMMANDS NOT TO BE ECHOED.

ALTERNATE_USER 1 AU: THIS PARAMETER SPECIFIES WHAT NOS USER TO CHECK IF THE DEFAULT NVE USER DOES NOT HAVE THE NEEDED FILE. ANY

NOS USER NAME IS ALLOWED.

OMISSION WILL CAUSE INT1 TO BE USED.

VALIDATE_USERS 1 VU: THIS PARAMETER SPECIFIES WHETHER TO RUN THE JOB THAT VALIDATES NOS/VE USERS. THIS PARAMETER IS IGNORED FOR A RECOVERY DEADSTART. VALID SPECIFICATIONS ARE

TRUE 1 YES 1 ON - RUN THE VALIDATION JOB

FALSE 1 NO 1 OFF - DO NOT RUN THE VALIDATION JOB

OMISSION WILL CAUSE THE VALIDATION JOB TO BE RUN.

QUICK_VALIDATE 1 QV: THIS PARAMETER DETERMINES WHICH USERS WILL BE VALIDATED BY THE VALIDATION JOB IF IT IS RUN. WHEN SPECIFIED, THIS PARAMETER GIVES A LIST OF USER NAMES TO BE VALIDATED IN ADDITION TO THE USERS: INT1, INT2, DEV1, DEV2, REL1, EVAL AND RKC.

OMISSION WILL CAUSE ALL USERS TO BE VALIDATED.

STATUS: SEE ERROR HANDLING IN THE NOS/VE ERS.
NOTES

THE GET_SOURCE_LIBRARIES PARAMETER HAS BEEN REMOVED. SOURCE LIBRARIES ARE NOW TREATED AS PRODUCTS AND THEIR INSTALLATION IS NOW CONTROLLED BY THE GET_PRODUCTS PARAMETER.

THE DEBUG PARAMETER HAS ALSO BEEN REMOVED. LATER IN THIS DOCUMENT THERE IS AN EXPLANATION OF HOW DS NOW HANDLES ABNORMAL SITUATIONS.

SUMMARY_OF_DS_PROCESSING

KIND_OF_INSTALLATION	PROCESSES_PERFORMED
I, N, R	A CONDITION HANDLER IS ESTABLISHED FOR ANY FAULT. IF SOMETHING GOES WRONG DURING THE DS PROCEDURE, IT WILL GET CONTROL, DISPLAY THE STATUS, AND LET THE OPERATOR INTERACT WITH THE DS PROCEDURE. THE OPERATOR CAN ENTER COMMANDS TO INVESTIGATE AND/OR CORRECT THE PROBLEM, RESUME EXECUTION OF DS BY ENTERING CONTINUE, OR ABORT DS BY ENTERING EXIT_PROC.
I, N, R	THE PROCEDURE "BRINGS UP" THE JOB LOG DISPLAY (ON THE LEFT SCREEN) WHERE THE

PROGRESS OF THE DS PROCEDURE MAY BE WATCHED, AND THE CONTROL POINT DISPLAY (ON THE RIGHT SCREEN).

I ONLY

SYSTEM FILES ARE CREATED FROM 170 CI BINARIES FOR

CYF$RUN_TIME_LIBRARY
OSF$OPERATOR_LIBRARY
OSF$SYSTEM_LIBRARY
OSF$OPERATOR_COMMAND_LIBRARY
OSF$COMMAND_LIBRARY

IF SAVE_INSTALL_FILES=YES, THE II BINARIES OF THE ABOVE FILES ARE REPLACED ON THE 170.

N ONLY

SYSTEM FILES ARE ACCESSED FROM 170, THESE ARE THE REPLACED II BINARIES MENTIONED ABOVE.

I, N, R

REMOTE HOST OUTPUT IS STARTED.

I, N

SYSTEM AND USER PROLOGS AND EPILOGS ARE BUILT.

I, N

THE VALIDATION JOB IS SUBMITTED AND SYSTEM STATISTICS ARE OUTPUT.

I, N

IF GET_PRODUCTS=YES, THE PRODUCT SET AND SOURCE LIBRARIES ARE INSTALLED; FAILED PRODUCT SET OR SOURCE LIBRARY INSTALLATIONS WILL PRINT A JOB LOG.

I, N

WAIT FOR VALIDATION JOB TO COMPLETE.

I, N, R

RHINPUT, IFEXEC, DUMP_BROKEN_JOB TASKS ARE STARTED.

I, N

FEEDBACK CAPABILITY SET UP.

I, N, R

SCL TASK STARTED.
JOB LOG PRINTED.
THE FOLLOWING MESSAGE IS OUTPUT

-- DEADSTART COMPLETED --

THE BIGGEST CHANGE TO THE DS PROCEDURE IS IN THE PRODUCT SET INSTALLATION (GET_PRODUCTS=YES). A PROCEDURE CALLED INSTALL_PRODUCTS IS USED TO INSTALL THE ENTIRE PRODUCT SET (PRODUCT SET AND SOURCE LIBRARIES). IN THIS PROCEDURE, A JOB IS SUBMITTED FOR EACH PRODUCT, IN EACH JOB A GET_PRODUCT IS

EXECUTED. THE PRODUCTS ARE INSTALLED IN THE $SYSTEM.LIBRARY CATALOG UNDER THE PRODUCT NAME. THE SUBMITTED JOBS CAN BE VIEWED ON A RIGHT K DISPLAY, THE LEFT K DISPLAY SHOWS THE PROCESSING OF THE DS PROCEDURE.

UPON JOB COMPLETION, A SUCCESS OR FAILURE MESSAGE WITH THE JOB NAME (REFLECTING THE PRODUCT BEING INSTALLED, I.E. CYI$COMPILER FOR PRODUCT CYF$COMPILER - THE CYBIL II COMPILER) IS WRITTEN TO A STATUS FILE (DSF$PRODUCT_JOB_STATUS). THE CONTENTS OF THIS FILE IS USED TO MAKE SURE ALL THE PRODUCT JOBS HAVE COMPLETED BEFORE CONTINUING WITH THE DS PROCEDURE, AS WELL AS THE CONTENTS BEING DISPLAYED TO THE JOB LOG WHICH CAN BE VIEWED BY THE OPERATOR.

IF A PRODUCT INSTALLATION FAILS, SEVERAL THINGS HAPPEN

- A FAILURE MESSAGE WITH JOB NAME IS WRITTEN TO THE STATUS FILE, WHICH WILL SUBSEQUENTLY BE DISPLAYED TO THE OPERATOR.

- THE JOB LOG FOR THE FAILED PRODUCT INSTALLATION WILL BE PRINTED, INDICATING THE ERROR.

- AT THE END OF THE INSTALL_PRODUCTS PROCEDURE, IF ANY PRODUCTS HAVE FAILED, THE CONDITION HANDLER ESTABLISHED AT THE START OF THE DS PROCEDURE GETS CONTROL. A STATUS MESSAGE OF "N PRODUCT INSTALLATIONS FAILED" IS DISPLAYED AND A PROMPT FOR OPERATOR INTERACTION OCCURS. THE OPERATOR AT THIS POINT CAN ATTEMPT REINSTALLATION OF FAILED PRODUCTS THROUGH THE USE OF INSTALL_PRODUCTS OR JUST CONTINUE WITH THE DS PROCEDURE.

THERE ARE TIMES WHEN IT IS NECESSARY TO RECREATE THE SYSTEM LIBRARIES - NEW PROCEDURES TO BE ADDED TO THE OPERATOR/SYSTEM COMMAND LIBRARIES, NEW VERSIONS OF THE CYBIL RUN TIME LIBRARY/OBJECT CODE UTILITY- BUT STILL MAINTAIN THE PERMANENT FILE BASE. THIS CAN BE ACCOMPLISHED VIA THE FOLLOWING CALL TO THE DS PROCEDURE

DS KIND=I SIF=TRUE VU=NO

THIS WILL REBUILD AND REPLACE THE SYSTEM LIBRARIES, BUT NOT GO THROUGH THE VALIDATION JOB RECREATING THE FAMILIES AND USERS.

4.6.1.2 THE_GET_PRODUCT_PROCEDURE

```
GET_PRODUCT    PRODUCT=<NAME>
               FILE=<170_FILE_NAME>
               USER=<170_USER_NAME>
               [CATALOG=<180_CATALOG_REFERENCE>]
               [FILE_CONTENTS=<NAME>]
               [FILE_STRUCTURE=<NAME>]
               [FILE_PROCESSOR=<NAME>]
               [RING_ATTRIBUTES=LIST 3..3 OF <INTEGER>]
               [STATUS=STATUS_VARIABLE]
```

PRODUCT 1 P: NAME OF THE PRODUCT

FILE 1 F: NAME OF THE 170 FILE THAT CONTAINS THE B56 FORM OF THE PRODUCT.

USER 1 U: NAME OF THE 170 USER THAT OWNS "FILE".

CATALOG 1 C: NAME OF THE CATALOG IN WHICH THE PRODUCT IS TO BE INSTALLED.

OMISSION WILL CAUSE $USER TO BE USED.

FILE_CONTENTS 1 FILE_CONTENT 1 FC: THE VALUE FOR THE PRODUCT FILE'S FILE_CONTENTS ATTRIBUTE.

OMISSION WILL CAUSE OBJECT TO BE USED.

FILE_STRUCTURE 1 FS: THE VALUE FOR THE PRODUCT FILE'S FILE_STRUCTURE ATTRIBUTE.

OMISSION WILL CAUSE LIBRARY TO BE USED.

FILE_PROCESSOR 1 FP: THE VALUE FOR THE PRODUCT FILE'S FILE_PROCESSOR ATTRIBUTE.

OMISSION WILL CAUSE UNKNOWN TO BE USED.

RING_ATTRIBUTES 1 RA: THE VALUES FOR THE PRODUCT FILE'S RING_ATTRIBUTES ATTRIBUTE.

OMISSION WILL CAUSE (11,11,11) TO BE USED.

STATUS: SEE ERROR HANDLING IN THE NOS/VE COMMAND INTERFACE ERS.

4.6.1.3 INSTALL_PRODUCTS_PROCEDURE

THE INSTALL_PRODUCTS PROCEDURE CAN BE USED AS AN INDEPENDENT COMMAND TO DO AN INSTALLATION/REINSTALLATION AT THE OPERATOR'S DISCRETION.

IF GET_PRODUCTS=YES, ALL OF THE FOLLOWING WILL BE INSTALLED.

C170 FILE	PRODUCT NAME (C180_FILE)	FILE CONTENTS
CYFIIC	CYF$COMPILER	CYBIL II COMPILER
ALFOL	ALF$OBJECT_LIBRARY	180 ASSEMBLER
SCFOL	SCF$OBJECT_LIBRARY	SOURCE CODE UTILITY
SCFCL	SCF$COMMAND_LIBRARY	SOURCE CODE UTILITY "STAND-ALONE" COMMAND LIBRARY
IFFEDIT	IFF$EDITOR	JACK BOHNHOFF'S EDITOR
PSFCL	PSF$COMMAND_LIBRARY	PRODUCT SET COMMANDS
AAF44D	AAF$44D_LIBRARY	AAM LIBRARY
AAF4DD	AAF$4DD_LIBRARY	AAM-FORTRAN INTERFACE
CBF7DD	CBF$7DD_LIBRARY	COBOL COMPILER
CBF4DD	CBF$4DD_LIBRARY	COBOL RUN-TIME LIBRARY
CCFLIB	CCF$LIBRARY	COMMON COMPILER MODULES
CGFLIB	CGF$LIBRARY	COMMON CODE GENERATOR
DBFLIB	DBF$LIBRARY	DEBUG LIBRARY
FCFLIB	FCF$LIBRARY	FORTRAN COMPILER
FLFLIB	FLF$LIBRARY	FORTRAN RUN-TIME
FMFLIB	FMF$LIBRARY	FILE MANAGEMENT UTILITY
MLFLIB	MLF$LIBRARY	MATH LIBRARY
SMFLIB	SMF$LIBRARY	SORT-MERGE
TTFLIB	TTF$TEST_TOOL_LIBRARY	TEST TOOL LIBRARY - ACR & EXAMINE
OSFPIL	OSF$PROGRAM_INTERFACE_LIBRARY	OPERATING SYSTEM PROGRAM INTERFACE
OSFSL	OSF$SOURCE_LIBRARY	SUBSET OF OPERATING SYSTEM SOURCE LIBRARY

INSTALL_PRODUCTS_PROCEDURE

```
INSTALL_PRODUCTS   [PRODUCTS=LIST OF <NAME> 1 ALL]
                   [STATUS=STATUS_VARIABLE]
```

PRODUCTS 1 PRODUCT 1 P: THE NAMES OF ONE OR PRODUCTS TO BE INSTALLED.

OMISSION WILL CAUSE ALL PRODUCTS TO BE INSTALLED.

STATUS: SEE ERROR HANDLING IN THE NOS/VE COMMAND INTERFACE

CYCLE 11, JANUARY 1983

ERS.

4.6.2 EXAMPLE OF NOS/VE INSTALLATION DEADSTART

TYPE

```
K,NVE.
K.SETLA (YOUR_UN,NVE) YOUR_PASSWORD
K.GETF DS U=SCAT
K.DS INSTALL  GP=NO AU=SCAT
```

4.6.3 EXAMPLE OF NOS/VE "NORMAL" DEADSTART

THE INTEGRATION SYSTEM HAS HAD THE INSTALLATION DEADSTART RUN ON IT.   ALSO THE FILES PRODUCED BY THE INSTALLATION DEADSTART HAVE BEEN MADE SEMI-PRIVATE AND ARE FOUND ON THE CATALOG USED IN THE NVEXXXX CALL.

TYPE (WHERE DEV1 IS THE SAME AS THE XXXX IN THE NVEXXXX CALL)

```
K,NVE.
K.SETLA (DEV1,NVE) DEV1X
K.GETF DS
K.DS NORMAL
```

4.6.4 EXAMPLE OF NOS/VE RECOVERY DEADSTART

THIS IS THE KIND OF DEADSTART THAT SHOULD MOST FREQUENTLY BE DONE IN A "CLOSED SHOP" ENVIRONMENT AND CONSEQUENTLY IS THE ONE FOR WHICH ALL THE PARAMETER DEFAULTS ARE ORIENTED.   IT PRESUPPOSES THAT PERMANENT FILE RECOVERY HAS BEEN SUCCESSFUL.

TYPE (WHERE DEV1 IS THE SAME AS THE XXXX IN THE NVEXXXX CALL)

```
K,NVE.
K.SETLA (DEV1,NVE) DEV1X
K.GETF DS
K.DS
```

4.6.5 EXAMPLE OF MINIMAL NOS/VE DEADSTART

THE MINIMAL DEADSTART SHOWN BELOW MAY BE USEFUL TO OS DEVELOPERS WHO NEED TO GET THE SYSTEM UP QUICKLY AND DO NOT NEED THE PRODUCT SET OR ALL VALIDATED USERS.

TYPE

K,NVE.
K.SETLA (YOUR_UN,NVE) YOUR_PASSWORD
K.GETF DS U=SCAT
K.DS NORMAL GP=NO QV=YOUR_UN AU=SCAT

4.6.6 USE OF THE QUICK_DEADSTART COMMAND

THIS COMMAND IS INTENDED AS A DEVELOPMENT TOOL TO FACILITATE
'FAST' DEADSTARTS WHERE RECOVERY IS NOT NEEDED; INDEED, IF THIS
COMMAND IS ENTERED RECOVERY_WILL_NOT_BE_PERFORMED WHEN THE SYSTEM
IS BROUGHT DOWN FOR WHATEVER REASON. SPECIFYING THIS COMMAND
WILL CAUSE AN INSTALLATION DEADSTART TO TAKE PLACE.   IF THE
INITDD COMMAND IS NOT SPECIFIED THEN A DEFAULT VALUE OF 'VSN001'
IS USED FOR THE SYSTEM DEADSTART DEVICE.   USE OF INITDD WILL
ALLOW SETTING THE DEADSTART DEVICES IDENTIFIER TO ANY VALUE.
THIS COMMAND WILL NOT BE ACCEPTED FROM A DEADSTART COMMAND FILE,
SO 'D=T' MUST BE SPECIFIED ON THE SETVE PROCEDURE TO PERMIT ENTRY
OF THE QUICKDS COMMAND.

    FORMAT: QUICKDS OR QUICK_DEADSTART

    VALUES: THE DEFAULT IS FALSE. EXECUTING THIS COMMAND CAUSES
THIS INITIAL VALUE TO BE TOGGLED, THUS EXECUTING THIS COMMAND
TWICE WILL CAUSE THE FINAL VALUE TO BE FALSE.

    NOTE: QUICK_TEMPLATE_LOAD DOES NOT EXIST NOW.

4.7 NOS/VE_INTERACTIVE_FACILITY_OPERATION

4.7.1 OPERATOR INITIATION

TO BRING UP THE NOS/VE INTERACTIVE FACILITY DO THE FOLLOWING

    1) BRING UP NOS/VE.

    2) BRING UP NAM

        AT THE SYSTEM CONSOLE ENTER

    NAM.

        WITH THE 5G NETWORKS, A FLASHING REQUEST FOR A DUMP TAPE
    WILL APPEAR, ENABLING THE OPERATOR TO DUMP A PREVIOUS
    NETWORK CRASH AFTER BRINGING NAM & FRIENDS UP AGAIN.   IF
    YOU DON'T WANT A DUMP ENTER ASSIGN,JSN,77. FOR THE
    APPROPRIATE JOB.

3) IF IAF IS NOT UP AT CONTROL POINT 1, ENTER

IAF.

4.7.2 OPERATOR TERMINATION

TO TERMINATE NOS/VE INTERACTIVE DO EITHER OF THE FOLLOWING

- PREFERRED METHOD

```
K,NAM.
K.AP=NVF.
K.DI,AP=VEIAF.
```

TO BRING VEIAF BACK UP, YOU MUST DO

```
K,NAM.
K.AP=NVF.
K.EN,AP=VEIAF.

X.DIS.
IIPPAS.
DROP.
```

- SECOND METHOD

```
IDLE,NAM.
IDLE,IAF.      (DON'T IDLE IAF WHEN IAFEX2 IS EXECUTING
               OR A PP WILL HANG.)
```

THIS TERMINATES THE ENTIRE NETWORK INCLUDING IAF, RBF, ETC. AND SHOULD BE USED ONLY IF YOU HAVE NO_INTENTION OF BRINGING VEIAF BACK UP BECAUSE THE 170 AND 180 SIDES OF VEIAF GET OUT OF SYNCH WHEN THE NETWORK IS BROUGHT DOWN THIS WAY.

4.7.3 OTHER OPERATOR CAPABILITIES

- TO LOGICALLY TURN THE PRINTER ON, UNDER DSD ENTER

ON,33.

- TO SEND A "SHUTDOWN WARNING" TO ALL TERMINALS LOGGED ON TO VEIAF DO

```
K,NAM.
K.AP=NVF.
```

K.ID,AP=VEIAF.

- TO SEND A MESSAGE TO ALL TERMINALS DO

CFO,NAM.MSG,ALL,MESSAGE.   K,NAM.
     K.AP=CS.
     K.NPU=NPUNAME,MESSAGE.
        NPUNAME=SN1147 ON THE S2,
               =SN1322 ON THE S3.

- PASSON HAS THE ABILITY TO RECORD VARIOUS TYPES OF DIAGNOSTIC INFORMATION. THIS CAPABILITY IS CONTROLLED VIA THE SENSE SWITCHES AT THE PASSON CONTROL POINT. TO TURN A SENSE SWITCH ON (OFF) AT JOB JSN DO

ONSW,JSN,X.   (OFFSW,JSN,X.)

WHERE X IS THE DESIRED SENSE SWITCH (1 TO 6). THE PASSON DEFAULT IS ALL SENSE SWITCHES OFF. IT WILL TAKE A SHORT PERIOD OF TIME BEFORE PASSON DETECTS A CHANGE IN A SENSE SWITCH AND REACTS TO IT. THE SENSE SWITCHES CURRENTLY USED BY PASSON ARE

SWITCH #	USE
1	NETWORK TRACE
2	PASSON LOGIC TRACE TO DAYFILE
3	MEMORY LINK TRACE TO DAYFILE

4.8 NOS/VE OPERATOR FACILITY AND OPERATOR COMMANDS

WITH THE RELEASE OF THE OPERATOR FACILITY PHASE 1 SEVERAL CHANGES TO THE NOS/VE OPERATING SYSTEM WILL OCCUR THAT WILL EFFECT THE USERS OF THE NOS/VE OPERATORS CONSOLE. THE OPERATOR FACILITY RUNS AS PART OF THE NVE CONTROL POINT. WHEN THE REQUEST FOR K DISPLAY APPEARS ON THE NOS B DISPLAY, ASSIGN THE K DISPLAY TO THE NVE CONTROL POINT. THE OPERATOR FACILITY IS CAPABLE OF DISPLAYING BOTH A LEFT AND A RIGHT SCREEN AREA AT THE SAME TIME. IF THE OPERATOR WANTS BOTH SCREENS THEN TYPE IN KK. THE CONTENTS OF THESE DISPLAYS ARE DETERMINED BY THE COMMANDS ENTERED BY THE OPERATOR.

THE LEFT SCREEN IS DIVIDED INTO FOUR DIFFERENT AREAS. THE TOP MOST AREA IS THE SYSTEM HEADER WHICH CONTAINS THE CURRENT DATE AND TIME, MEMORY STATISTICS, AND AN OPERATOR ACTION MESSAGE IF ONE IS POSTED. THE OPERATOR ACTION WILL INCLUDE THE JOB SEQUENCE NUMBER OF THE OWNER OF THE MESSAGE AND 'MESSAGE CANCELLED' IF THE MESSAGE IS CANCELLED BECAUSE THE TASK HAS TERMINATED. THE NEXT

LINE CONTAINS THE FIRST 64 CHARACTERS OF THE OPERATOR ACTION
MESSAGE (60 CHARACTERS IN STAND ALONE).

THE NEXT AREA OF THE SCREEN IS THE MAIN OUTPUT AREA. THIS
AREA HAS THE FILE NAME OF OUTPUT. ANY DISPLAY COMMAND CAN HAVE
ITS OUTPUT DIRECTED TO THIS AREA AS WELL AS ANY SYSTEM COMMAND.

THE THIRD AREA IS TOWARDS THE BOTTOM OF THE SCREEN. THIS AREA
IS TWO LINES LONG AND CONTAINS THE RESPONSE AREA. THIS WILL
CONTAIN ERROR MESSAGES FROM SYSTEM COMMANDS. THE AREA IS CLEARED
WHEN THE NEXT OPERATOR TYPEIN IS ENTERED AT THE OPERATOR'S
CONSOLE AND RECEIVED BY NOS/VE.

THE FOURTH AREA IS THE PROMPT AREA. THIS WILL CONTAIN THE
STATUS OF THE KEYBOARD. IF NOS/VE IS PROCESSING A COMMAND, THEN
THE KEYBOARD IS LOCKED AND ALL TYPEINS WILL BE IGNORED. WHEN THE
KEYBOARD IS LOCKED, THE MESSAGE 'DATA RECEIVED BY 180 - KEYBOARD
LOCKED' WILL APPEAR AT THE SCREEN'S BOTTOM. WHEN THE KEYBOARD IS
UNLOCKED THEN ANY DATA IN THE KEYBOARD BUFFER WILL BE SENT TO
NOS/VE. THE BOTTOM LINE IS THE LAST LINE THAT WAS PROCESSED BY
OPERATOR FACILITY.

THE RIGHT K DISPLAY HAS THE FILE NAME OUTPUT_RIGHT. THERE IS
ONLY ONE AREA ON THE RIGHT SCREEN THEREFORE THE MAIN DISPLAY AREA
IS 10 LINES LONGER THAN THE LEFT SCREEN AREA. IF A DAYFILE
DISPLAY OR AJ DISPLAY IS SHOWN ON THE RIGHT SCREEN YOU WILL GET
MORE LINES OF INFORMATION THAN ON THE LEFT SCREEN.

THERE ARE NO DEFAULT DISPLAYS THAT COME UP AUTOMATICALLY ON
EITHER OUTPUT DISPLAY AREA. IT IS UP TO THE OPERATOR TO DECIDE
THE DISPLAY THE OPERATOR WISHES TO SEE. THE ONLY PARTS OF THE
DISPLAY THAT COME UP AUTOMATICALLY IS THE SYSTEM HEADER DISPLAY
AND THE PROMPT AREA FOR KEYBOARD STATUS.

THE PAGE WIDTH OF THE SCREEN IS 60 CHARACTERS FOR STANDALONE
AND 64 CHARACTERS FOR DUAL STATE. THE CHARACTER SET TRANSLATION
CODE IS THE SAME AS THAT FOR THE CURRENT NVE SUBSYSTEM CONTROL
POINT. THE ESCAPE CODE SEQUENCE FOR THE SPECIAL CHARACTERS TO BE
TYPED HAS NOT CHANGED. THERE ARE A FEW DIFFERENCES IN THE
PROCESSING OF DATA BY THE OPERATOR FACILITY AND NVE SUBSYSTEM.

1) DO NOT END COMMANDS WITH A PERIOD. PERIODS ARE SENT TO
NOS/VE.

2) THE NVE SUBSYSTEM COMMANDS THAT BEGIN WITH AN ASTERISK WILL
NOT BE SUPPORTED FROM THE OPERATOR FACILITY CONTROL POINT. IF
THESE COMMANDS ARE ENTERED FROM THE OPERATOR FACILITY THEY
WILL BE PASSED ON TO SCL WHERE AN ILLEGAL COMMAND WILL BE

ISSUED.

3) ROUTING OF CONSOLE JOB DATA TO A SPECIFIC JOB BY THE 'N=COMMAND' PROTOCOL WILL NOT BE SUPPORTED IN DUAL STATE. THIS FEATURE SHOULD WORK IN STANDALONE BUT WILL NOT BE SUPPORTED.

4) NO TYPE AHEAD — COMMANDS CANNOT BE ENTERED UNTIL THE PROMPT AREA SHOWS THAT THEY ARE REQUESTED.

THERE IS ONE NEW COMMAND TO REPLACE THE CURRENT DISPLAY COMMANDS. THE ENTRY POINTS FOR ZDIS, ZDISB, AND SDIS HAVE BEEN DELETED. THE NEW COMMAND IS VEDISPLAY AND HAS TWO PARAMETERS. THE OPTIONS ARE LISTED BELOW. THE VALUES IN PARANTHESIS ARE THE ABBREVIATIONS. NOTE THAT THIS COMMAND DOES NOT BEGIN WITH AN ASTERISK (*). THIS COMMAND WILL BE PROCESSED BY SCL AND CREATE A NEW SYSTEM CONTROL POINT TASK TO DISPLAY IT'S DATA. THE USER CAN HAVE THE SAME DISPLAY TYPE ON EACH OF THE DISPLAY AREAS, IF THE USER SO DESIRES.

COMMAND NAME	DISPLAY TYPE	SCREEN AREA
	PARAMETER NAME	PARAMETER NAME
	DISPLAY_OPTIONS DISPLAY_OPTION (DO)	OUTPUT (O)
	PARAMETER VALUES	PARAMETER VALUES
VEDISPLAY (VED)	DISPLAY_SYSTEM_LOG JOB_LOG (JL) ACTIVE_JOBS (AJ)	OUTPUT OUTPUT_RIGHT (OR)

THE DEFAULT FILE NAME FOR ALL DISPLAYS IS OUTPUT.

THE FOLLOWING IS A BRIEF LIST OF COMMANDS TO BRING UP NOSVE WITH THE OPERATOR FACILITY INSTALLED.

NVEFFFF.   TO BRING UP NOS/VE. (SEE SECTION 5)

K,NVE.

KK.   TO BRING UP THE K DISPLAY ON BOTH SCREENS.

K.VED JL   TO BRING UP THE JOB LOG.
OR
K.VED DISPLAY_OPTIONS=JL OUTPUT=OUTPUT

                    TO BRING UP THE JOB LOG   USING   KEY   WORD
                                   IDENTIFIERS.

K.VED AJ OUTPUT_RIGHT
                    TO BRING UP THE CONTROL POINT DISPLAY ON THE
                                   RIGHT SCREEN.
OR
K.VED DISPLAY_OPTIONS=CONTROL_POINT OUTPUT=OUTPUT_RIGHT
                    TO BRING UP THE CONTROL POINT DISPLAY USING
                                   KEY WORD IDENTIFIERS.

K.XXX   SEND ANY COMMAND TO NOS/VE.
  •
  •
  •
  •

K.TERMINATE_SYSTEM   TO TERMINATE NOS/VE.

    NOTE: AFTER THE DS PROCEDURE HAS COMPLETED EXECUTION THE
COMMAND TO ENTER TO BRING DOWN THE SYSTEM IS TERMINATE_SYSTEM,
NOT TERMINATE_SYSTEM_JOB.   YES, THIS IS THE OPPOSITE OF WHAT IT
USED TO BE.

4.8.1 DELETE_JOB_QUEUE 1 DELETE_JOB_QUEUES 1 DELJQ

    THE PURPOSE OF THIS COMMAND IS TO DELETE ALL FILES FROM THE
JOB INPUT SUBCATALOG, THE PRINT SUBCATALOG OR BOTH. THIS COMMAND
IS ONLY ALLOWED FROM JOBS WITH OPERATOR AND OR SYSTEM PRIVILEGES.
    DELETE_JOB_QUEUE [QUEUE_NAME=INPUT1OUTPUT1ALL]
    [STATUS=<STATUS VARIABLE>]

    QUEUE_NAME   1   QN: THIS PARAMETER SPECIFIES FROM WHICH
SUBCATALOGS FILES ARE TO BE DELETED. SPECIFYING INPUT WILL CAUSE
ALL FILES TO BE DELETED FROM THE JOB SWAP SUBCATALOG AND THE JOB
INPUT SUBCATALOG. SPECIFYING OUTPUT WILL CAUSE ALL FILES TO BE
DELETED FROM THE JOB OUTPUT SUBCATALOG. OMISSION WILL CAUSE ALL
TO BE USED.

    STATUS: SEE ERROR HANDLING.

4.8.2 REBUILD_INPUT_QUEUE 1 REBIQ

    THE PURPOSE OF THIS COMMAND IS TO REBUILD AN ENTRY IN THE
KNOWN JOB LIST (KJL) FROM INFORMATION IN THE SYSTEM LABEL OF THE
FILE REPRESENTING THE JOB BEING PROCESSED. THIS COMMAND IS TO BE
USED DURING THE PROCESS OF RECOVERING THE INPUT QUEUES DURING
RECOVERY DEADSTART.
    REBUILD_INPUT_QUEUE [NAME=<NAME>]
    [STATUS=<STATUS VARIABLE>]

    NAME 1 N: THIS PARAMETER SPECIFIES THE FILE NAME OF THE FILE

REPRESENTING THE JOB. AN ATTEMPT IS MADE TO PROCESS THE
SPECIFIED FILE WITHIN THE CATALOG WHERE JOB INPUT QUEUES ARE
KNOWN TO RESIDE.

STATUS: SEE ERROR HANDLING.

4.8.3 REBUILD_OUTPUT_QUEUE 1 REBOQ

THE PURPOSE OF THIS COMMAND IS TO REBUILD AN ENTRY IN THE
KNOWN OUTPUT LIST (KOL) FROM INFORMATION RETAINED IN THE SYSTEM
LABEL OF THE FILE REPRESENTING THE OUTPUT BEING PROCESSED. THIS
COMMAND IS TO BE USED DURING THE PROCESS OF RECOVERING THE OUTPUT
QUEUES DURING A RECOVERY DEADSTART.
    REBUILD_OUTPUT_QUEUE [NAME=<NAME>]
    [STATUS=<STATUS VARIABLE>]

    NAME: THIS PARAMETER SPECIFIES THE FILE NAME OF THE FILE
REPRESENTING THE OUTPUT. AN ATTEMPT IS MADE TO PROCESS THE
SPECIFIED FILE WITHIN THE CATALOG PATH OF WHERE JOB OUTPUT QUEUES
ARE KNOWN TO RESIDE.

    STATUS: SEE ERROR HANDLING.
N.N.N ACTIVATE_SYSTEM_TASK 1 ACTST

    THE PURPOSE OF THIS COMMAND IS TO INITIATE ANY OF THREE TASKS
TYPICALLY INITIATED FROM THE DS PROC.
    ACTIVATE_SYSTEM_TASK TASK_NAMES = <LIST OF <RHINPUT, RHOUTPUT,
IFEXEC>>
    [STATUS=<STATUS_VARIABLE>]
N.N.N DEACTIVATE_SYSTEM_TASK 1 DEAST

    THE PURPOSE OF THIS COMMAND IS TO TERMINATE ANY OF THREE TASKS
TYPICALLY INITIATED FROM THE DS PROC.
    DEACTIVATE_SYSTEM_TASK TASK_NAMES = <LIST OF <RHINPUT,
RHOUTPUT, IFEXEC>>
    [STATUS=<STATUS_VARIABLE>]

4.9 ROUTE_AN_INPUT_FILE_FROM_C170_TO_C180

    THROUGH THE SYSTEM CONSOLE, ENTER

X.DIS.
USER,A,B.
GET,FILENAME.
    WHERE FILENAME IDENTIFIES THE INPUT FILE TO BE ROUTED.
ROUTE,FILENAME,DC=LP,FC=RH.

4.10 CONFIGURATION MANAGEMENT

IN ORDER TO RUN NOS/VE IN A DUAL STATE ENVIRONMENT, CERTAIN HARDWARE CONFIGURATION RULES MUST BE IN EFFECT. ALL STORAGE DEVICES AND CHANNELS THAT NOS/VE WILL REQUEST MUST BE IN A 'DOWN' STATE IN THE NOS EST. FOR MORE INFORMATION ON THE NOS EST, SEE THE NOS VERSION 2 OPERATOR/ANALYST HANDBOOK, 60459310, SECTION 4. REFER TO THE E DISPLAY.

TO RUN S2 S/N 104 IN THE STANDARD CONFIGURATION, CHANNELS 2 AND 32 MUST BE DOWNED AS SHOWN BELOW.

```
DOWN,CH2.
DOWN,CH32.
```

4.10.1 SYSTEM CORE COMMANDS

CONFIGURATION MANAGEMENT (CM) SYSTEM CORE COMMANDS DEFINE THE SYSTEM MASS STORAGE DEVICE. THE COMMANDS ARE DESCRIBED IN DETAIL IN THE NOS/VE INSTALLATION COMMAND INTERFACE ERS, SECTION 3. THE CM SYSTEM CORE COMMANDS DEFINE THE HARDWARE CONFIGURATION OF THE SYSTEM DEVICE AND THE CONTROLLER ACCESSING THE SYSTEM DEVICE. AN EXAMPLE FOR S2 S/N 104 IS SHOWN HERE.

```
SETDCT $7155_1
SETDD $885_11 40(8)
```

4.10.2 JOB TEMPLATE COMMANDS

DURING AN INSTALLATION DEADSTART, THE PHYSICAL CONFIGURATION UTILITY (PCU) AND THE LOGICAL CONFIGURATION UTILITY (LCU) MUST BE RUN. THE PCU IS RUN TO DESCRIBE THE PHYSICAL CONFIGURATION THAT NOS/VE MAY ACCESS. THE LCU WILL LOGICALLY INSTALL ALL OR PART OF THE PHYSICAL CONFIGURATION AND ALLOW NOS/VE TO ACCESS THE EQUIPMENT. COMMANDS FOR BOTH UTILITIES ARE DESCRIBED IN THE NOS/VE INSTALLATION COMMAND INTERFACE ERS, SECTION 2.

THE SYSTEM DEVICE WHICH WAS DEFINED BY THE SYSTEM CORE COMMANDS MUST THE DEFINED VIA THE PCU, USING THE SAME CONTROLLER AND CHANNEL NUMBER. ADDITIONAL DEVICES MAY BE DEFINED AS LONG AS THE DEVICE IS IN A DOWN STATE, AND ALL CHANNELS DEFINED ARE ALSO DOWN. AN EXAMPLE IS LISTED HERE.

```
EXECUTE_TASK SP=MANAGE_PHYSICAL_CONFIGURATION
    SET_MAINFRAME_DEFINITION MAINFRAME=S2
    SET_DATA_CHANNEL_DEFINITION CHANNEL_NUMBER=2
    SET_DATA_CHANNEL_DEFINITION CHANNEL_NUMBER=18
    SET_CONTROLLER_DEFINITION ELEMENT=CT1 ..
```

```
           PRODUCT_IDENTIFICATION=$7155_1 ..
           SERIAL_NUMBER=2 CHANNEL_CONNECTION=CHANNEL2
      SET_CONTROLLER_DEFINITION ELEMENT=CT2 ..
           PRODUCT_IDENTIFICATION=$7155_1 ..
           SERIAL_NUMBER=3 CHANNEL_CONNECTION=CHANNEL18
      SET_STORAGE_DEVICE_DEFINITION ELEMENT=FMD40_A ..
           PRODUCT_IDENTIFICATION=$885_11 SERIAL_NUMBER=1234 ..
           UNIT_NUMBER=40(8) CONTROLLER_CONNECTION=CT1
      SET_STORAGE_DEVICE_DEFINITION ELEMENT=FMD40_B ..
           PRODUCT_IDENTIFICATION=$855_11 SERIAL_NUMBER=3456 ..
           UNIT_NUMBER=40(8) CONTROLLER_CONNECTION=CT2
      INSTALL_PHYSICAL_CONFIGURATION MAINFRAME=S2
QUIT

EXECUTE_TASK SP=MANAGE_LOGICAL_CONFIGURATION
     INSTALL_LOGICAL_CONFIGURATION INCLUDE=ALL
QUIT
```

4.10.3 MULTIPLE VOLUME CONSIDERATIONS

IF A CONFIGURATION IS USED DEFINING MULTIPLE MASS STORAGE
VOLUMES, THE EXECUTION OF THE PCU AND LCU WILL ONLY PERMIT ACCESS
TO THE SYSTEM DEVICE.   IN ORDER TO BRING ADDITIONAL VOLUMES
ONLINE, THE LCU MUST BE RUN AGAIN.   THE USER MAY WANT TO
INITIALIZE A FRESH VOLUME. TO DO THIS, USE THE LCU SUBCOMMAND
INITMV. TO ADD A VOLUME TO THE NOS/VE SET, I.E. TO ALLOW NOS/VE
USE OF THE VOLUME, USE THE LCU SUBCOMMAND ADDMTS.   AN EXAMPLE IS
SHOWN HERE.

```
EXECUTE_TASK TP=MANAGE_LOGICAL_CONFIGUATION
     INITIALIZE_MS_VOLUME ELEMENT=FMD40_B RECORDED_VSN='VSN002'
     ADD_MEMBER_TO_SET MEMBER_VSN='VSN002'
QUIT
```

DURING A NONINSTALLATION DEADSTART, AN ATTEMPT WILL BE MADE TO
BRING ALL LOGICALLY CONFIGURED VOLUMES ONLINE AND REACTIVATE
THEM.   THIS MEANS THE ABOVE SEQUENCE NEED ONLY OCCUR WHEN A NEW
VOLUME IS BEING BROUGHT ONLINE THE FIRST TIME.

REFERENCE HAS BEEN MADE TO THE NOS/VE INSTALLATION COMMAND
INTERFACE ERS. TO GET A COPY OF THIS DOCUMENT ON S/N 101, ENTER
SES,MAD.LISTNIH.

4.11 K_DISPLAY_ASCII

SUPPORT OF 6-12 ASCII FROM THE CONSOLE (K DISPLAY) CAUSES THE
FOLLOWING CHANGES

INPUT	TRANSLATED_TO	INPUT	TRANSLATED_TO

CYCLE 11, JANUARY 1983

```
/1                              /(           [
/2          "                   /)           ]
/3          #                   /+           >
/4          $                   /-           <
/5          \                   /=           .
/6          ;                   /*           ' (SINGLE QUOTE)
/7          ?                   //           /
/8          0                   /,
/9          2                   /A TO /Z  A - Z (LOWER CASE)
/0          _ (UNDERSCORE)
```

4.12 EDD AND DSDI INFORMATION

MOST OF THE STEPS LISTED HERE ARE USED TO GATHER INFORMATION FOR RECOVERY PROBLEMS, ETC. IF YOU DO NOT WISH TO DO THIS, EXECUTE STEPS D AND E ONLY (EDD DUMP AND NOS LEVEL 3 DEADSTART).

A. ENTER DR I AT THE MDD DISPLAY.

B. ENTER HP AT THE MDD DISPLAY.

C. IF IOU.FS1 NOT = 0 OR IOU.FS2 NOT = 0 THEN DO NOT ENTER DU AT THE MDD DISPLAY ELSE DO ENTER DU AT THE MDD DISPLAY.

D. PRESS DEADSTART BUTTON AND PERFORM EDD

1) MOUNT SCRATCH TAPE (RING IN) ON A 9-TRACK DRIVE.

2) PUSH D/S BUTTON.

3) SELECT U (UTILITIES) DISPLAY.

4) SELECT E (EDD) DISPLAY.

5) SET CHANNEL (S2=13).

6) SET ECUU (S2=01UU)

E = EQUIPMENT

C = 1 FOR 67X DRIVES, 2 FOR 66X DRIVES

UU = UNIT NUMBER OF THE TAPE DRIVE TO BE USED.

7) ANSWER "DUMP NUMBER" WITH A CR.

8) ANSWER "NON ZERO INHIBITS REWIND" WITH A CR.

9) ANSWER "CHANNEL CONTROLWARE" WITH A CR. WARNING: IF THIS STEP IS OMITTED, DSDI CANOT PROCESS THE DUMP TAPE.

E. PERFORM A LEVEL 3 NOS DEADSTART (SEE SECTION 4.14).

F. IF YOU ARE RUNNING ON THE S3, ENTER THE FOLLOWING AT THE CONSOLE

```
X.DIS.
USER,INT1,INT1X.
GET,DMPDFS/UN=JLG.
BEGIN,,DMPDFS.
  .
  .
DROP.
```

THIS IS A PROCEDURE TO DUMP SEVERAL DAYFILES ASSOCIATED WITH THE NVEXXXX JOB AND ARE NEEDED ALONG WITH THE EDD TAPE TO ADEQUATELY DEBUG A NOS/VE HANG.

SOME OF THE CRASHES THAT NOS/VE ENCOUNTERS WILL CAUSE NOS/VE TO TERMINATE SUCH THAT THE OPERATOR WILL NOTICE THE K-DISPLAY WHICH REQUESTS THAT A K.*RUN. BE ENTERED. BEFORE A K.*RUN. IS ENTERED THE EDD SHOULD BE PERFORMED.

TO CREATE A LISTING OF THE EDD TAPE, TYPE SES,INT1.DSDI SEE THE PROCEDURE'S HELP DOCUMENTATION FOR PARAMETER DESCRIPTIONS.

C170 DSDI INFORMATION CAN BE FOUND IN CHAPTER 10 OF THE NOS SYSTEM MAINTENANCE MANUAL.

A170 DSDI INFO CAN BE FOUND IN DOCUMENT ARH3060 -- GID FOR A170 NOS/S2.

4.13 NOS/VE_TERMINATION

O BRINGING DOWN DUAL STATE

```
K,NVE.
K.TERMINATE_SYSTEM
```

O IF NOT A NORMAL TERMINATION

```
K,NVE.

K.*RUN.
K.*ENDLST.
K.*ENDRUN.
```

4.14 A170_NOS_SHUTDOWN

BEFORE LEAVING THE MACHINE, IT IS NECESSARY TO BRING NOS DOWN. IF NOS HAS CRASHED, A LEVEL 3 DEADSTART MUST BE ATTEMPTED EVEN IF

THE ONLY REASON IS TO BRING NOS DOWN. TO DO A LEVEL 3 DEADSTART

1) PUSH D/S BUTTON

2) SELECT "O" DISPLAY

3) SELECT "P" DISPLAY

4) ENTER I=3

5) ENTER (CR)

6) ENTER DATE/TIME

   IF A DUMP IS DESIRED BUT A CRASH HAS NOT OCCURRED STEP.
SHOULD ALWAYS BE ENTERED BEFORE PUSHING THE DEADSTART BUTTON.
AFTER THE DUMP HAS BEEN TAKEN A LEVEL 3 DEADSTART SHOULD BE
PERFORMED.

   TO BRING NOS DOWN, DO THE FOLLOWING

1) ENTER

   CHE
   THE SCREEN WILL DISPLAY
   CHECK POINT SYSTEM.
   ENTER: CARRIAGE RETURN

2) MAKE SURE NO MASS STORAGE DEVICE HAS A CHECKPOINT REQUESTED.
   TO DO THIS, ENTER: E,M. IF THE DISPLAY SHOWS THERE ARE NO
   "C"S IN THE STATUS FIELD, THEN ALL DEVICES ARE CHECKPOINTED
   AND YOU MAY CONTINUE.

3) ENTER

STEP.

4) PUSH DEADSTART BUTTON.

5.0 RECOVERY_OF_NOS/VE_PERMANENT_FILES

5.1 SETVE_FORMAT

THE GENERAL FORMAT OF THE SETVE COMMAND IS

X.SETVE(PN=FFFF,UN=UN,VSN=VSN,D=D,P=P,B=B,C=C,CH=CH)

FFFF IS A STRING OF NO MORE THAN FOUR CHARACTERS. SETVE APPENDS FFFF TO 'NVE' TO CONSTRUCT THE NAME OF A PROCEDURE FILE WHICH, WHEN INVOKED, WILL DEADSTART NOS/VE. THE DEFAULT IS TST.

UN SPECIFIES THE USER NUMBER FROM WHICH TPXXXK IS ATTACHED. UN IS THE FIRST CATALOG SEARCHED FOR OTHER FILES USED IN DEADSTARTING AND TERMINATING NOS/VE. THE DEFAULT IS INT1.

C SPECIFIES THE DEADSTART COMMAND DECK TO BE USED WHEN DEADSTARTING NOS/VE. THE FUNCTION SERVED BY THE DEADSTART COMMAND DECK IS ANALOGOUS TO THE FUNCTION SERVED BY THE CMRDECK OF NOS. CURRENTLY SUPPORTED VALUES FOR C AND THEIR RESPECTIVE USES ARE

        1   ARDEN HILLS S3 S/N 02 OPEN SHOP/HANDS-ON TIME
        3   ARDEN HILLS S3 S/N 02 CLOSED SHOP
        6   ARDEN HILLS S2 S/N 104 OPEN SHOP/HANDS-ON TIME
        10   SUNNYVALE S2 CLOSED SHOP

    THE DEFAULT IS SET IN THE FILE CMDS1. CURRENTLY THE DEFAULT IS 6. WARNING!__SPECIFYING_C=6_ON_THE_ARDEN HILLS_S3_WILL_DESTROY_THE_NOS/VE_PERMANENT_FILE_BASE.

B SPECIFIES AN ALTERNATE CATALOG TO BE SEARCHED FOR THE VARIOUS FILES USED IN DEADSTARTING AND TERMINATING NOS/VE. THE DEFAULT IS INT1.

D IS USED TO INDICATE THAT THE SYSTEM CORE COMMAND PROCESSOR SHOULD ACCEPT COMMANDS FROM THE CONSOLE. SPECIFYING D=T IN THE SETVE COMMAND ALLOWS THE OPERATOR TO ENTER COMMANDS FROM THE CONSOLE AFTER PROCESSING THE DEADSTART COMMAND DECK. IF THE OPERATOR WISHES TO INITIALIZE THE SYSTEM DEVICE AND/OR INSTALL A NEW VERSION OF NOS/VE, D=T MUST BE SPECIFIED. THE DEFAULT IS SET IN THE FILE CMDS1. CURRENTLY THE DEFAULT IS D=F.

P SPECIFIES THE PASSWORD FOR THE CATALOG INDICATED BY THE UN PARAMETER. IF THIS PARAMETER IS OMITTED THE PASSWORD WILL BE GENERATED BY APPENDING AN 'X' TO THE UN PARAMETER.

    CH   SPECIFIES THE OCTAL CHANNEL TO BE USED FOR NOS/VE DISK
           I/O.   THE DEFAULT IS SET IN THE FILE CMDS1. CURRENTLY
           THE DEFAULT IS 2.

    VSN   SPECIFIES   THE   VSN   OF   A   DEADSTART   TAPE.   IF   THIS
           PARAMETER   IS USED THEN NOS/VE WILL BE DEADSTARTED FROM
           THE TAPE SPECIFIED. IF IT IS OMITTED, THEN NOS/VE WILL
           BE DEADSTARTED FROM THE PERMANENT FILE TPXXXK.

5.2 SETVE_USAGE

    EARLIER VERSIONS OF NOS/VE REQUIRED THAT TWO SETVE COMMANDS BE
ISSUED IF THE SYSTEM WAS TO BE INSTALLED AND SUBSEQUENTLY
RECOVERED.   THE CURRENT SYSTEM DOES NOT REQUIRE THIS. THE ONLY
REASON FOR ISSUING TWO SETVE COMMANDS IS TO PROVIDE A DEADSTART
PROCEDURE THAT DOES NOT REQUIRE/PERMIT OPERATOR INTERVENTION.

    IT SHOULD BE NOTED THAT ONCE THE SETVE COMMAND HAS BEEN
ISSUED, IT NEED NOT BE ISSUED AGAIN UNLESS...

    1) THERE IS A NEED TO CHANGE ONE OR MORE OF THE PARAMETERS
       SPECIFIED IN SETVE.

    2) THE COMMAND FILE, NVEFFFF, GENERATED BY THE SETVE COMMAND
       IS PURGED FROM THE SYSTEM.

    TWO EXAMPLES OF SETVE USAGE AND SUBSEQUENT NOS/VE DEADSTART
ARE GIVEN BELOW.   THE FIRST EXAMPLE SHOWS A "HANDS ON" USER
WORKING WITH RECOVERY. THE SECOND ILLUSTRATES THESE CONCEPTS FOR
A TYPICAL NOS/VE CLOSED SHOP. THIS WRITER HOPES THAT THE READER
WILL FIND BOTH EXAMPLES USEFUL AND ILLUMINATING.

    I. A "HANDS ON" USER

        THE COMMAND FILE NVERSD IS BUILT AND INSTALLED IN NOS BY
      TYPING

          X.SETVE(RSD,UN=RSD,VSN=TAPE,B=DEV1,C=6,D=T)

      SEVERAL OF THE CHOICES OF PARAMETERS ARE WORTH NOTING.

      1) BY SPECIFYING THE VSN PARAMETER THE USER HAS BUILT A
         COMMAND FILE THAT WILL DEADSTART THE SYSTEM DIRECTLY
         FROM THE TAPE PRODUCED BY NVESYS.

      2) THE USER IN THIS EXAMPLE HAS SPECIFIED DEADSTART
         COMMAND DECK 6 (C PARAMETER) AND HAS ALLOWED THE
         NOS/VE DISK I/O CHANNEL (CH PARAMETER) TO DEFAULT TO
         2. ONE CONCLUDES THAT THE USER IS RUNNING ON THE
         ARDEN HILLS S2.

3) THE USER HAS SPECIFIED D=T. THIS IS IMPORTANT. THE DEADSTART COMMAND WHICH TRIGGERS INSTALLATION OF A RECOVERABLE SYSTEM CANNOT BE READ FROM A DEADSTART COMMAND DECK. IT MUST BE ENTERED FROM THE CONSOLE AT DEADSTART TIME. SPECIFYING D=T ALLOWS THE OPERATOR TO ENTER COMMANDS FROM THE CONSOLE.

NOS/VE IS DEADSTARTED BY TYPING

NVERSD.

AT THE CONSOLE.

THE USER BRINGS UP THE K DISPLAY BY TYPING

K,NVE.

PRESENTLY, THE DEADSTART COMMAND DECK IS DISPLAYED AND THE USER IS PROMPTED FOR INPUT. THE DEADSTART COMMAND DECK USED IN THIS EXAMPLE LOOKS LIKE THIS

```
SET_SYSTEM_ATTRIBUTE HALTRING 15
ENAP
USE_CONFIGURATION_PROLOG S2SN104_DEVELOPMENT
USE_INITIALIZATION_PROLOG EMPTY
SET_DEADSTART_CONTROLLER $7155_1
SET_DEADSTART_DEVICE $885_12 41(8)
```

THE USER TYPES

```
K.INITDD VSN001.
K.GO.
```

THE SYSTEM ACCEPTS THE COMMANDS AND INSTALLS AND DEADSTARTS NOS/VE.

AFTER THE SYSTEM COMES DOWN, VIA EITHER CONTROLLED TERMINATION OR A CRASH, THE SYSTEM CAN BE RECOVERED (IF NECESSARY) AND REDEADSTARTED BY TYPING

NVERSD.

WHEN THE DEADSTART COMMAND DECK IS DISPLAYED, THE USER TYPES

K.GO.

THIS WILL CAUSE NOS/VE TO BE DEADSTARTED WITHOUT INITIALIZING THE SYSTEM DEVICE.

II. A TYPICAL "CLOSED SHOP"

TWO COMMAND FILES, NVECLSH AND NVEINST, ARE CREATED BY TYPING

X.SETVE(CLSH,UN=CLSH,B=DEV1,C=10,CH=1)
X.SETVE(INST,UN=CLSH,B=DEV1,C=10,CH=1,D=T)

ONE NOTES THAT

1) CLOSED SHOP IS DEADSTARTED FROM A TPXXXK FILE IN THE CLSH CATALOG.

2) SPECIFYING D=T, FOR NVEINST, CAUSES DEADSTART TO PAUSE FOR OPERATOR INTERVENTION.

A NORMAL DEADSTART IS USED WHEN BRINGING UP NOS/VE AT THE BEGINNING OF CLOSED SHOP OR FOLLOWING A SYSTEM FAILURE. THE OPERATOR, IN THIS EXAMPLE, TYPES

NVECLSH.

NOS/VE WILL RECOVER (IF NECESSARY) AND DEADSTART. WHEN RECOVERY RUNS, THE OPERATOR MAY BE REQUIRED TO RESPOND TO A REQUEST FOR INPUT FROM THE MANLC UTILITY. SEE NOTE 4 AT THE END OF THIS SECTION FOR MORE INFORMATION.

IF IT IS NECESSARY TO REINITIALIZE THE SYSTEM DEVICE, THE OPERATOR KEYS

NVEINST.

THIS CAUSES DEADSTART TO PAUSE AND WAIT FOR OPERATOR INPUT WHEN DEADSTART COMMANDS ARE BEING PROCESSED.

THIS AMOUNTS TO AN INSTALLATION DEADSTART. THIS SHOULD BE USED ONLY WITH FULL KNOWLEDGE THAT ANY FILES WHICH MAY HAVE EXISTED ON MASS STORAGE PRIOR TO THIS DEADSTART, WILL BE BLASTED INTO OBLIVION BY IT.

AN INSTALLATION DEADSTART WILL BE REQUIRED IF

1) THIS IS THE INITIAL INSTALLATION OF A RECOVERABLE VERSION OF NOS/VE.

2) IF THE FILE SYSTEMS OF THE SYSTEM BEING INSTALLED AND THE SYSTEM BEING SUPERCEDED ARE NOT COMPATIBLE.

3) IF THE FILE SYSTEM HAS BEEN DAMAGED BEYOND THE POSSIBILITY OF RECOVERY.

AN INSTALLATION DEADSTART IS EFFECTED BY TYPING

K.INITDD VSN001.

CYCLE 11, JANUARY 1983

K.GO.

WHEN THE DEADSTART COMMAND DECK IS DISPLAYED AND THE OPERATOR IS PROMPTED FOR INPUT.

NOTES

1. THE CMDS1 FILE USED TO DEADSTART NOS/VE MUST HAVE THE DEBUG2 FLAG SET TO TRUE. WHEN NOS/VE IS DEADSTARTED, THE CATALOG SPECIFIED BY THE UN PARAMETER IS FIRST IN THE SEARCH ORDER FOR CMDS1, FOLLOWED BY THE CATALOG SPECIFIED BY THE B PARAMETER.

2. SEE SECTION 3.3 OF THE INTEGRATION PROCEDURES NOTEBOOK FOR OTHER INFORMATION ABOUT THE CMDS1 FILE.

3. IF NOS/VE CRASHES AND A DUMP IS DESIRED (IN THE CONTEXT OF OUR SECOND EXAMPLE)

I. TYPE DU AT THE MDD CONSOLE. THE MESSAGE "WRITING IMAGE FILE" SHOULD APPEAR IMMEDIATELY. THE MESSAGE "IMAGE FILE COMPLETE" SHOULD APPEAR A FEW MOMENTS LATER.

II. PUSH THE DEADSTART BUTTON.

III. TAKE THE EDD DUMP.

IV. DO A LEVEL 3 NOS DEADSTART.

ALTERNATIVELY THE OPERATOR CAN SKIP STEP I IF SHE/HE IS SURE TO REDEADSTART NOS/VE AFTER STEP IV. IN THIS EVENT THE SYSTEM WILL DETECT THAT THE IMAGE FILE WAS NEVER CREATED, WILL CREATE ONE, AND WILL RECOVER FROM IT.

4. A RECOVERY DEADSTART WHICH IS UNDER THE CONTROL OF A SETVE COMMAND IN WHICH D=T WAS SPECIFIED WILL PAUSE IN THE RECOVERY SYSTEM WITH THE MESSAGE "OPERATOR INTERVENTION BEFORE RECOVERY BEGINS". THE SYSTEM IS EXECUTING THE LOGICAL CONFIGURATION UTILITY AT THIS POINT. IN ORDER TO EXIT THIS UTILITY AND ALLOW RECOVERY TO PROCEED THE OPERATOR TYPES K.QUIT (NO PERIOD). FOR MORE DETAILS ON THE USE OF THE LCU SEE SECTION 4.9 OF THIS DOCUMENT AND THE NOS/VE INSTALLATION COMMAND INTERFACE ERS.

5.3 MIGRATING FROM LEVEL N TO LEVEL N±1: INSTALLING JOB TEMPLATES

THE READER IS STRONGLY ADVISED TO READ THIS ENTIRE PROCEDURE BEFORE BEGINNING IT.

1) USING NVELINK, LINK THE SYSTEM JOB TEMPLATES.

THE OUTPUT FROM THE LINKER IS TWO MASS STORAGE FILES CALLED PJBXXYY AND RJBXXYY. THESE 2 FILES NEED TO BE REFORMATTED INTO THE FORMAT USED IN BRINGING UP NOS/VE. THIS REFORMATTING IS ACCOMPLISHED BY THE BLDJTMP PROCEDURE, WHICH IS INVOKED AS FOLLOWS

SES,INT1.BLDJTMP JTF=FILE1 NJTF=FILE2

WHERE FILE1 = EITHER PJBXXYY OR RJBXXYY AND FILE2 = THE NEW NAME FOR THE REFORMATTED VERSION OF THE JOB TEMPLATE.

THIS PROCEDURE WILL BE USED ONCE FOR EACH TEMPLATE.

EXAMPLE

SES,INT1.BLDJTMP JTF=PJBXXYY NJTF=PRO
SES,INT1.BLDJTMP JTF=RJBXXYY NJTF=REC

WHEN BOTH OF THESE CALLS ARE COMPLETE THE FILES PRO AND REC WILL BE IN THE USER'S CATALOG.

2) DUMP THE REFORMATTED FILES (PRO AND REC IN THE EXAMPLE) TO TAPE, AND RELOAD THE FILES ON A 170 CATALOG ON THE TARGET MACHINE.

3) BRING UP LEVEL N OF NOS/VE.

NOTE: SINCE THE COMMANDS TO INSTALL THE NEW SET OF JOB TEMPLATES MUST BE ISSUED BY THE OPERATOR, ALL COMMANDS SHOWN HERE ARE ASSUMED TO BE ISSUED FROM THE SYSTEM CONSOLE.

DO A GETF ON EACH OF THE LOADED FILES SPECIFYING B60 CONVERSION.

EXAMPLE: (ASSUME PRO AND REC ARE IN THE 170 CATALOG.)

GETF TO=PTEMPL FROM=PRO DC=B60
GETF TO=RTEMPL FROM=REC DC=B60

WHERE PTEMPL AND RTEMPL ARE 180 LOCAL FILE NAMES.

4) AFTER BOTH FILES ARE LOCAL, DO (FOR EACH TEMPLATE)

CONJT (LOCAL FILE NAME)

THIS COMMAND WILL CONVERT THE TEMPLATE TO 180 FORMAT (60 - 64 BIT CONVERSION). THIS COMMAND IS RESTRICTED TO THE OPERATOR JOB (I.E. SYSTEM CONSOLE).

CONTINUING_THE_EXAMPLE

    CONJT PTEMPL
    CONJT RTEMPL

5) CREATE THE ACTUAL JOB TEMPLATE FILE BY ENTERING

    CREJT TN=(TEMPLATE NAME) FN=(FILE NAME)

    WHERE (TEMPLATE NAME) IS THE NAME OF THE JOB TEMPLATE FILE
    BEING INSTALLED ON THE SYSTEM. THIS IS THE NAME THAT WILL
    BE USED TO IDENTIFY THE JOB TEMPLATE DURING FUTURE
    DEADSTARTS.   (FILE NAME) IS THE NAME OF THE CONVERTED JOB
    TEMPLATE FILE.

        NOTE: BEFORE USING THIS COMMAND DISPLAY THE JOB LOG (VED
    JL) AND CHECK THAT THE JMP$LOAD_JOB_TEMPLATE TASK COMPLETES
    BEFORE ATTEMPTING ANOTHER CREJT COMMAND OR GOING TO THE
    NEXT STEP.

        CONTINUING_THE_EXAMPLE

    VED JL
    CREJT TN=P10 FN=PTEMPL

    CHECK FOR TASK COMPLETION

    CREJT TN=R10 FN=RTEMPL

    CHECK FOR TASK COMPLETION

6) TAKE NOS/VE DOWN AND RECOVER THE LEVEL N SYSTEM. PERFORM A
    CONTINUATION DEADSTART USING THE DEADSTART (TPXXXK) TAPE
    FOR THE LEVEL N+1 SYSTEM.

7) WHEN THE SYSTEM IS WAITING FOR OPERATOR INTERVENTION, IN
    ADDITION TO ANY OTHER COMMANDS, DO

    USE_SYSTEM_JOB_TEMPLATE (TEMPLATE NAME) OR

    USESJT (TEMPLATE NAME)

    WHERE (TEMPLATE NAME) IS THE NAME USED ON THE CREJT
    COMMAND.

        CONTINUING_THE_EXAMPLE

    USE_SYSTEM_JOB_TEMPLATE P10

8) LET DEADSTART PROCEED.

9) IF A CRASH OCCURS OR A RECOVERY IS NECESSARY, WHEN THE
    RECOVERY SYSTEM IS WAITING FOR OPERATOR COMMANDS, TYPE

USE_RECOVERY_JOB_TEMPLATE (TEMPLATE NAME) OR

USERJT (TEMPLATE NAME)

WHERE (TEMPLATE NAME) IS THE NAME USED ON THE CREJT COMMAND.

CONTINUING_THE_EXAMPLE

USE_RECOVERY_JOB_TEMPLATE R10

10) AS AN ALTERNATE TO TYPING IN THE USE ... JOB TEMPLATE COMMANDS, A DEADSTART COMMAND FILE (DCFILE) CAN BE CREATED SPECIFYING BOTH THE USESJT XX AND USERJT YY COMMANDS IN THE FILE IN ADDITION TO OTHER COMMANDS. AGAIN, XX IS THE NAME USED FOR THE TN PARAMETER ON THE CREJT COMMAND FOR THE SYSTEM JOB TEMPLATE AND YY IS THE NAME USED FOR THE TN PARAMETER ON THE CREJT COMMAND FOR THE RECOVERY JOB TEMPLATE.

11) TO REMOVE OLD JOB TEMPLATES USE THE COMMAND DELETE_JOB_TEMPLATE (TEMPLATE NAME) OR DELJT (TEMPLATE NAME) WHERE (TEMPLATE NAME) IS THE NAME SPECIFIED ON THE CREJT COMMAND.

6.0 SYSTEM_CORE_DEBUGGER

THE SYSTEM CORE DEBUGGER PROVIDES A SET OF CAPABILITIES INTENDED TO ASSIST IN DEBUGGING THE OPERATING SYSTEM. SERVICES PROVIDED BY THE DEBUGGER ARE TASK ORIENTED: SELECTION OF THE TASKS TO BE DEBUGGED MUST BE MADE VIA DEBUGGER SUBCOMMANDS. NO TASKS WILL BE UNDER CONTROL OF THE DEBUGGER UNLESS THEY ARE SELECTED. THE SELECTION CAPABILITY ALLOWS ANY NUMBER OF TASKS TO BE DEBUGGED SIMULTANEOUSLY; FROM ONE TASK TO ALL TASKS IN THE SYSTEM. OBVIOUSLY A CAPABILITY THIS POWERFUL MUST BE USED WITH SOME CARE. THE SYSTEM CORE DEBUGGER USES THE DEBUG HARDWARE TO PROVIDE THESE CAPABILITIES.

6.1 SYSDEBUG

THE PURPOSE OF THIS COMMAND IS TO INITIATE EXECUTION OF THE SYSTEM CORE DEBUGGER. THIS COMMAND CAN BE ISSUED FROM THE DEADSTART COMMAND FILE OR AS A COMMAND IN ANY JOB.

SYSDEBUG

THIS COMMAND HAS NO PARAMETERS; ALL INFORMATION THE DEBUGGER REQUIRES IS PROVIDED VIA SUBCOMMANDS.

THE SYSTEM CORE DEBUGGER CAN ALSO BE INVOKED FROM THE MDD CONSOLE. THE FORMAT OF THE COMMAND IS

DO N.SYSDEBUG

WHERE N IS THE JOB ORDINAL OF THE DESIRED JOB. THE DEBUGGER IS BROUGHT UP IN THE JOB MONITOR TASK OF THE JOB. ALL SYSTEM CORE DEBUGGER SUBCOMMANDS ARE AVAILABLE, BUT MUST BE PREFIXED BY THE MDD COMMAND DO.

THE SYSTEM CORE DEBUGGER CAN ALSO BE BROUGHT UP (FROM THE MDD CONSOLE) BY SPECIFYING A GLOBAL TASK ID. THE FORMAT OF THE COMMAND IS

DO N.TOEBUG GGGGGG

THE VALUE OF N IS IGNORED, AND THE VALUE GGGGGG SPECIFIES THE NOS/VE GLOBAL TASK ID (3 HEX BYTES) OF THE TASK TO BRING THE DEBUGGER UP IN. IF THE TASK ID IS INVALID, THEN THE COMMAND WILL BE IGNORED.

6.2 SUBCOMMAND_PARAMETER_DEFINITIONS

<NAME>
<CONDITION> ::= READ1WRITE1RNI1BRANCH1CALL1DIVFLT1ARLOS1

```
                     AROVFL1EXOVFL1EXUNFL1FPLOS1FPINDEF1INVBDP
<BASE>
<OFFSET> ::= INTEGER
<LENGTH> ::= INTEGER
<FRAME> ::= 1..100
<COUNT> ::= 1..10000
<REGID> ::= X1A1P
<REGNO> ::= 0..1510..OF(16)
<VALUE> ::= INTEGER
<TIME>
<VSTRING> ::= 'CHARSTRING'
<DATATYPE> ::= HEX1ASCII1ASC1DEC
<CHANGE_COUNT> ::= 1..8
<SELECTOR> ::= FULL1AUTO1SAVE
```

6.3 SYSTEM_CORE_DEBUGGER_SUBCOMMANDS

WITHIN THE DESCRIPTIONS WHICH FOLLOW, OPTIONAL PARAMETERS ARE
ENCLOSED IN BRACKETS. DEFAULT VALUES FOR OPTIONAL PARAMETERS ARE
ALSO DEFINED.

6.3.1 SELECT

THE PURPOSE OF THIS SUBCOMMAND IS TO SELECT THE TASKS IN WHICH
THE SYSTEM CORE DEBUGGER IS TO BE ACTIVE. WHEN THE DEBUGGER IS
FIRST CALLED, IT IS NOT ACTIVE IN ANY TASK. TO USE THE DEBUGGER
THEREFORE, IT IS NECESSARY TO SELECT THE TASKS IN WHICH IT IS TO
BE ACTIVE.

SELECT <SELECTION OPTION> [<RING NUMBER> 1 <ACTIVE JOB LIST
ORDINAL>]

SELECTION_OPTION: THIS PARAMETER SPECIFIES ONE OF A SERIES OF
SELECTION OPTIONS USED TO CONTROL THE TASKS IN WHICH THE DEBUGGER
WILL BE ACTIVE AND SOME OTHER DEBUG OPTIONS. THE SELECTIONS ARE
REMAIN IN EFFECT UNTIL THEY ARE EXPLICITLY CHANGED WITH
SUBSEQUENT SELECT SUBCOMMANDS. VALID SELECTION OPTIONS ARE

<RIGHT1LEFT> - THIS SELECTS THE SCREEN FOR THE DEBUG
DISPLAY. THE DISPLAY STAYS ACTIVE WHEN THE SCREEN IS
SWITCHED.

<JOBMONITOR1NOJOBMONITOR> - THIS SELECTS WHETHER OR NOT
TO DEBUG JOB MONITOR TASKS.

<USER1NOUSER> - THIS SELECTS WHETHER OR NOT TO DEBUG USER
TASKS (I.E. THOSE THAT ARE NOT JOB MONITORS).

<HIGHRING> - THIS SPECIFIES THE HIGHEST RING IN WHICH
DEBUG TRAPS WILL BE RECOGNIZED. TRAPS OCCURRING IN RINGS
ABOVE THIS SELECTION WILL BE IGNORED.

<JOB1NOJOB> - THIS ENABLES OR DISABLES DEBUGGING FOR THE JOB AT THE SPECIFIED ACTIVE JOB LIST ORDINAL. THE SYSTEM JOB HAS AN ACTIVE JOB LIST ORDINAL OF ZERO.

<ALLJOBS1NOJOBS> - THIS ACTIVATES OR DEACTIVATES DEBUGGING IN ALL JOBS.

THE INITIAL SELECTIONS ARE: RIGHT, NOSTEP, NOJOBMONITOR, NOUSER, HIGHRING=3, NOJOBS.

6.3.2 BREAKPOINT 1 B

THE PURPOSE OF THIS SUBCOMMAND IS TO SELECT A PROGRAM INTERRUPT WHICH IS TO TAKE PLACE UPON OCCURRENCE OF A SPECIFIED CONDITION WITHIN A SPECIFIED VIRTUAL ADDRESS RANGE.

BREAKPOINT <NAME> <CONDITION> [<BASE>] [<OFFSET>] [<LENGTH>]

THE <NAME> IS ANY USER SUPPLIED NAME FOR IDENTIFYING THE BREAKPOINT. A MAXIMUM OF THIRTY TWO BREAKPOINTS CAN BE SELECTED. WHEN A TRAP OCCURS, THE <NAME> OF THE BREAKPOINT WHICH CAUSED THE TRAP IS DISPLAYED.

THE BASE PARAMETER IS REQUIRED WHEN SPECIFYING A NEW BREAKPOINT NAME; OFFSET AND LENGTH SPECIFICATIONS ARE OPTIONAL IN THIS CASE. WHEN ADDING A NEW CONDITION SELECTION TO AN EXISTING BREAKPOINT, BASE, OFFSET, AND LENGTH PARAMETERS MAY NOT BE SPECIFIED.

BASE, OFFSET, AND LENGTH PARAMETERS DEFINE THE DESIRED VIRTUAL ADDRESS RANGE: <BASE> + <OFFSET> YIELDS A FIRST-BYTE-ADDRESS; FIRST-BYTE-ADDRESS + <LENGTH> -1 YIELDS A LAST BYTE ADDRESS.

DEFAULT PARAMETER VALUES

<OFFSET>: 0
<LENGTH>: 1

6.3.3 REMOVE_BREAKPOINT 1 RB

THE PURPOSE OF THIS SUBCOMMAND IS TO DESELECT A PREVIOUSLY SELECTED PROGRAM INTERRUPT.

REMOVE_BREAKPOINT <NAME> [<CONDITION>]

IF ONLY THE NAME PARAMETER IS SPECIFIED, ALL CONDITIONS ASSOCIATED WITH THE BREAKPOINT ARE DESELECTED AND ALL EVIDENCE OF THE BREAKPOINT IS REMOVED. IF THE CONDITION PARAMETER IS SPECIFIED, ONLY THAT CONDITION IS DESELECTED; HOWEVER, IF THE SPECIFIED CONDITION IS THE ONLY CONDITION SELECTED, ALL EVIDENCE

OF THE NAMED BREAKPOINT IS REMOVED.

6.3.4 LIST_BREAKPOINT 1 LB

THE PURPOSE OF THIS SUBCOMMAND IS TO PROVIDE A LIST OF CURRENTLY SELECTED BREAKPOINTS AND ASSOCIATED CONDITIONS.

   LIST_BREAKPOINT [<NAME>]

IF THE NAME PARAMETER IS SPECIFIED, INFORMATION IS DISPLAYED FOR THE NAMED BREAKPOINT ONLY. IF THE NAME PARAMETER IS NOT SPECIFIED, INFORMATION IS DISPLAYED FOR ALL CURRENTLY DEFINED BREAKPOINTS.

6.3.5 CHANGE_BREAKPOINT 1 CB

THE PURPOSE OF THIS SUBCOMMAND IS TO CHANGE THE VIRTUAL ADDRESS RANGE OF A PREVIOUSLY SPECIFIED BREAKPOINT.

   CHANGE_BREAKPOINT <NAME> <BASE> [<OFFSET>] [<LENGTH>]

BASE, OFFSET, AND LENGTH PARAMETERS DEFINE THE DESIRED VIRTUAL ADDRESS RANGE: <BASE> + <OFFSET> YIELDS A FIRST-BYTE-ADDRESS; FIRST-BYTE-ADDRESS + <LENGTH> -1 YIELDS A LAST BYTE ADDRESS.

   DEFAULT PARAMETER VALUES

<OFFSET>: 0
<LENGTH>: 1

6.3.6 TRACE_BACK 1 TB

THE PURPOSE OF THIS SUBCOMMAND IS TO PROVIDE INFORMATION RELEVANT TO STACK FRAMES ASSOCIATED WITH AN INTERRUPTED PROCEDURE AND ITS PREDECESSOR PROCEDURES. VALIDATION OF PVA'S IS NOW PERFORMED.

INFORMATION DISPLAYED FOR EACH SELECTED STACK FRAME CONSISTS OF

- STACK FRAME NUMBER;

- CURRENT P-ADDRESS OF THE ASSOCIATED PROCEDURE;

- VIRTUAL ADDRESS OF THE START OF THE STACK FRAME;

- VIRTUAL ADDRESS OF THE STACK FRAME SAVE AREA.

   TRACE_BACK [<FRAME>] [<COUNT>] [FULL1SHORT] [PVA]

THE FRAME PARAMETER SPECIFIES THE NUMBER OF THE FIRST STACK FRAME FOR WHICH INFORMATION IS TO BE DISPLAYED. STACK FRAME NUMBER ONE IS ASSOCIATED WITH THE INTERRUPTED PROCEDURE, STACK FRAME TWO IS ASSOCIATED WITH THE INTERRUPTED PROCEDURE'S PREDECESSOR, ETC.

THE MODULE NAME PROVIDED ON THE TRACEBACK IS USUALLY CORRECT BUT NOT GUARANTEED.

THE COUNT PARAMETER SPECIFIES THE TOTAL NUMBER OF STACK FRAMES FOR WHICH INFORMATION IS TO BE DISPLAYED.

THE PVA PARAMETER SPECIFIES A STACK FRAME PVA TO BEGIN THE TRACEBACK. THIS IS USEFUL WHEN A STACK IS VERY LARGE AND YOU WISH TO DISPLAY IT RANDOMLY.

DEFAULT PARAMETER VALUES

<FRAME>: 1
<COUNT>: 1

6.3.7 DISPLAY_STACK_FRAME 1 DSF

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY SELECTED INFORMATION FROM A SPECIFIED STACK FRAME.

DISPLAY_STACK_FRAME [<FRAME>] [<SELECTOR>]

THE FRAME PARAMETER SPECIFIES THE NUMBER OF THE STACK FRAME FOR WHICH INFORMATION IS TO BE DISPLAYED. (STACK FRAME NUMBER ONE IS ASSOCIATED WITH THE INTERRUPTED PROCEDURE, STACK FRAME TWO IS ASSOCIATED WITH THE INTERRUPTED PROCEDURE'S PREDECESSOR, ETC.)

THE SELECTOR PARAMETER IDENTIFIES A REGION OF THE SPECIFIED STACK FRAME
AUTO: CAUSES THE AUTOMATIC REGION OF THE STACK FRAME TO BE DISPLAYED.
SAVE: CAUSES THE SAVE AREA OF THE STACK FRAME TO BE DISPLAYED.
FULL: CAUSES BOTH THE AUTOMATIC AND SAVE AREAS OF THE STACK FRAME TO BE DISPLAYED.

DEFAULT PARAMETER VALUES

<FRAME>: 1
<SELECTOR>: FULL

6.3.8 DISPLAY_REGISTER 1 DR

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY THE CONTENTS OF A
SPECIFIED REGISTER OF AN INTERRUPTED PROCEDURE.

DISPLAY_REGISTER <REGID> [<REGNO>] [<DATATYPE>]

DEFAULT PARAMETER VALUES

<REGNO>: 0
<DATATYPE>: HEX

6.3.9 DISPLAY_MEMORY 1 DM

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY THE CONTENTS OF A
SPECIFIED AREA OF VIRTUAL MEMORY. VALIDATION OF PVA'S   IS   NOW
PERFORMED.

DISPLAY_MEMORY <BASE> [<LENGTH>] [F1N]

THE F1N PARAMETER ALLOWS YOU TO FORCE (F) MEMORY TO BE
DISPLAYED EVEN IF IT IS BEYOND THE CURRENT SEGMENT LENGTH.   THE
DEFAULT IS NOT TO FORCE (N) THE DISPLAY.

DEFAULT PARAMETER VALUES

<LENGTH>: 8

6.3.10 CHANGE_MEMORY 1 CM

THE   PURPOSE   OF   THIS   SUBCOMMAND IS TO SET A SPECIFIED VALUE
INTO A SPECIFIED LOCATION OF VIRTUAL MEMORY FOR A SPECIFIED
NUMBER OF BYTES. VALIDATION OF PVA'S IS NOW PERFORMED.

CHANGE_MEMORY <BASE> <VALUE> <CHANGE_COUNT>

DEFAULT PARAMETER VALUES

<CHANGE_COUNT>: 1

6.3.11 RUN

THE   PURPOSE OF THIS SUBCOMMAND IS TO INVOKE PROGRAM EXECUTION
AFTER A SELECTED PROGRAM INTERRUPT HAS OCCURRED.

RUN

CYCLE 11, JANUARY 1983

6.3.12 SUPER_CHANGE_MEMORY 1 SCM

THE PURPOSE OF THIS SUBCOMMAND IS THE SAME AS THE CHANGE MEMORY SUBCOMMAND, THAT IS, TO CHANGE THE CONTENTS OF VIRTUAL MEMORY. IT DIFFERS FROM CHANGE MEMORY, HOWEVER, IN THAT IT WILL CHANGE THE ATTRIBUTES OF THE SEGMENT TO ALLOW MEMORY TO BE WRITTEN, AND THEN CHANGE THE ATTRIBUTES BACK TO THEIR ORIGINAL VALUES.

THE COMMAND FORMAT IS THE SAME AS THE CHANGE MEMORY SUBCOMMAND.

6.3.13 FORMAT 1 FMT

THE PURPOSE OF THIS SUBCOMMAND IS TO SET THE SYSTEM CORE DEBUGGER INTO A MODE WHERE ALL SUBCOMMAND OUTPUT IS SENT TO A LOCAL FILE WHICH IS PRINTED (PRIF) WHEN THE UNFMT SUBCOMMAND IS ENTERED. THIS IS DONE BY HAVING THE TASK RUNNING THE DEBUGGER COMMUNICATE WITH ANOTHER TASK RUNNING IN THE SYSTEM JOB. IT IS THIS OTHER TASK THAT ACTUALLY WRITES AND PRINTS THE FILE. THE ENTRY POINT OF THIS TASK IS OSP$BROKEN_JOB_DUMP_TASK. IT WILL NORMALLY BE INITIATED BY THE DS PROCEDURE. IF IT IS NOT RUNNING, A DIAGNOSTIC WILL BE ISSUED. THE PARAMETER TO THIS COMMAND IS A STRING WHICH WILL BE OUTPUT AS THE FIRST LINE OF THE FILE.

COMMAND FORMAT

FORMAT STRING

6.3.14 UNFORMAT 1 UNFMT

THE PURPOSE OF THIS SUBCOMMAND IS TO LEAVE THE OUTPUT MODE ESTABLISHED BY THE FORMAT COMMAND. OUTPUT WILL AGAIN BE SENT TO THE OPERATOR CONSOLE. AT THIS POINT THE PERMANENT FILE WILL BE FLUSHED TO MASS STORAGE.

COMMAND FORMAT

UNFORMAT

6.3.15 DISPLAY_MONITOR_FAULT 1 DISMF

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY ANY MONITOR FAULTS PRESENT IN THIS TASK. SEE THE SECTION TITLED 'NOS/VE PROCESSING OF JOB MODE SOFTWARE ERRORS' FOR MORE INFORMATION.

ALL MONITOR FAULT BUFFERS ARE DISPLAYED IN THE HOPE THEY WILL SHOW SOME TASK HISTORY. IF A GIVEN FAULT BUFFER IS INVALID THE

MESSAGE "FOLLOWING FAULT IS NOT PRESENT" IS DISPLAYED.

COMMAND FORMAT

DISPLAY_MONITOR_FAULT

6.3.16 DISPLAY_XCB 1 DISXCB

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY ALL OF THE FIELDS OF THE CURRENT TASK'S (I.E., THE TASK RUNNING THE DEBUGGER) EXECUTION CONTROL BLOCK.

6.3.17 DISPLAY_TASK_ENVIRONMENT 1 DISTE

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY THE XCB OF ALL TASKS RUNNING WITHIN THE CURRENT JOB (I.E., THE JOB WITH THE TASK RUNNING THE DEBUGGER). IF THE COMMAND IS ENTERED WHILE THE DEBUGGER IS IN FORMAT MODE, THEN A FULL XCB IS DISPLAYED, OTHERWISE JUST THE TASK NAME, XCB ADDRESS AND GLOBAL TASK ID ARE DISPLAYED.

6.3.18 +

THE PURPOSE OF THIS SUBCOMMAND IS TO INCREMENT THE LAST DM COMMAND ADDRESS BY THE LAST DM COMMAND COUNT AND EXECUTE THE RESULTING DM COMMAND. THERE ARE NO PARAMETERS TO THIS COMMAND.

6.3.19 -

THE PURPOSE OF THIS SUBCOMMAND IS TO DECREMENT THE LAST DM COMMAND ADDRESS BY THE LAST DM COMMAND COUNT AND EXECUTE THE RESULTING DM COMMAND. THERE ARE NO PARAMETERS TO THIS COMMAND.

6.3.20 DISPLAY_AUTO_COMMAND_LIST 1 DISACL

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY ALL DEBUGGER COMMANDS THAT WILL BE EXECUTED DURING AN AUTOMATIC DUMP. THERE ARE NO PARAMETERS TO THIS COMMAND.

6.3.21 SET_AUTO_COMMAND_LIST 1 SETACL

THE PURPOSE OF THIS SUBCOMMAND IS TO CHANGE OR ADD A COMMAND TO THE AUTOMATIC DUMP COMMAND LIST. THE AUTOMATIC DUMP EXECUTES COMMANDS FROM THE START OF THE LIST UNTIL IT ENCOUNTERS A BLANK (EMPTY) COMMAND.

COMMAND FORMAT

SETACL N 'STRING'

THE N PARAMETER SPECIFIES WHICH COMMAND IS TO BE CHANGED, STARTING WITH 1. THE STRING PARAMETER (ENCLOSED IN QUOTES) SPECIFIES THE COMMAND TO BE PLACED IN POSITION N.

6.3.22 DISPLAY_SYSTEM_LOG 1 DISSL

THE PURPOSE OF THIS COMMAND IS TO DISPLAY THE SEGMENT NUMBER AND CURRENT LENGTH OF A SYSTEM LOG.

COMMAND FORMAT

DISSL LOG START LEN

THE LOG PARAMETER SPECIFIES WHICH LOG TO DISPLAY, AND MAY BE ONE OF: SYSTEM, STATS, ENG, ACCOUNT. IF LOG IS SYSTEM, THEN THE START AND LEN PARAMETERS ARE INTERPRETED AS FOLLOWS.

THE START PARAMETER SPECIFIES THE FIRST BYTE OF THE LOG SEGMENT TO DISPLAY (ONLY FOR SYSTEM LOG).

THE LEN PARAMETER SPECIFIES THE NUMBER OF BYTES TO DISPLAY (ONLY FOR SYSTEM LOB).

6.3.23 DISPLAY_JOB_LOG 1 DISJL

THE PURPOSE OF THIS COMMAND IS TO DISPLAY THE SEGMENT NUMBER AND CURRENT LENGTH OF A JOB LOG.

COMMAND FORMAT

DISJL LOG START LEN

THE LOG PARAMETER SPECIFIES WHICH LOG TO DISPLAY AND MAY BE ONE OF: JOB, SYSTEM, STATS, ENG, ACCOUNT, JOBSTAT. IF LOG IS JOB, THEN THE START AND LEN PARAMETERS ARE INTERPRETED AS FOLLOWS.

THE START PARAMETER SPECIFIES THE FIRST BYTE OF THE LOG SEGMENT TO DISPLAY (ONLY FOR JOB LOG).

THE LEN PARAMETER SPECIFIES THE NUMBER OF BYTES TO DISPLAY (ONLY FOR JOB LOG).

6.3.24 DISPLAY_STACK_ENVIRONMENT 1 DISSE

THE PURPOSE OF THIS SUBCOMMAND IS TO DISPLAY THE CURRENT SEGMENT NUMBER AND TOP OF STACK VALUES FOR EACH RING IN THE

CURRENT TASK. IN FORMAT MODE THE CONTENTS OF THE SEGMENT FROM
ZERO TO TOP OF STACK ARE ALSO DISPLAYED.

   THERE ARE NO PARAMETERS TO THIS COMMAND.

6.3.25 HANG

   THE PURPOSE OF THIS SUBCOMMAND IS TO CAUSE THE CURRENT TASK TO
HANG IN A DELAY LOOP WHEN THE DEBUGGER IS EXITED.   THE HANG
COMMAND IS A TOGGLE; THAT IS, IF HANG IS SET ENTERING THE COMMAND
AGAIN WILL CLEAR IT AND VICE VERSA. TRAPS ARE LEFT ENABLED, SO
THAT THE TASK MAY BE RE-ENTERED. A TASK HUNG IN THIS FASHION
CANNOT BE RETURNED TO ITS PRE-HUNG STATE.

   THERE ARE NO PARAMETERS TO THIS COMMAND.

7.0 NOS/VE PROCESSING OF JOB MODE SOFTWARE ERRORS

7.1 INTRODUCTION

TASKS RUNNING IN JOB MODE WILL OCCASIONALLY CAUSE AN ERROR WHICH IS DETECTED EITHER BY THE HARDWARE OR NOS/VE MONITOR. THE ACTION TAKEN WHEN AN ERROR LIKE THIS OCCURS IS CONTROLLED BY VARIOUS SYSTEM ATTRIBUTES. THE PURPOSE OF THIS SECTION IS TO DISCUSS THE TYPES OF ERRORS AND THE EFFECT A GIVEN SYSTEM ATTRIBUTE WILL HAVE UPON THE HANDLING OF THE ERROR.

7.2 TYPES OF ERRORS

1) BROKEN TASK: A BROKEN TASK IS A TASK IN WHICH THE TRAP MECHANISM IS NOT ABLE TO FUNCTION CORRECTLY. NOS/VE MONITOR WILL ATTEMPT TO REPAIR THE TRAP MECHANISM AND SEND A BROKEN TASK FAULT TO THE TASK. THE SPECIFIC CASES OF A BROKEN TASK ARE

SYSTEM ERROR	JOB MODE SOFTWARE HAS DECLARED THE TASK TO BE BROKEN. (THIS IS A SPECIAL CASE OF BROKEN TASK.)
MONITOR FAULT BUFFER FULL	JOB MODE ERRORS ARE OCCURING BUT ARE NOT BEING PROCESSED BY JOB MODE.
TRAPS DISABLED	A JOB MODE ERROR HAS OCCURRED WHILE TRAPS WERE DISABLED.
INVALID A0	THE TASK'S A0 REGISTER WAS INVALID.
UCR/MCR TRAPS DISABLED	UCR/MCR ERROR OCCURRED WITH TRAPS DISABLED.

2) MCR FAULT: THIS ERROR SIGNIFIES THAT JOB MODE CAUSED A HARDWARE DETECTED MCR FAULT. THIS MAY BE CAUSED BY SOFTWARE OR HARDWARE DETECTED UNCORRECTABLE ERROR.

3) UNKNOWN SYSTEM REQUEST: THIS ERROR SIGNIFIES THAT JOB MODE ISSUED A MONITOR REQUEST THAT IS EITHER INVALID OR CANNOT BE ISSUED FROM THE RING IT WAS ISSUED FROM.

4) SEGMENT ACCESS FAULTS: THESE ERRORS SIGNIFY THAT JOB MODE ENCOUNTERED OR CAUSED ONE OF THE FOLLOWING ERRORS

   - PAGE FAULT FOR AN ADDRESS GREATER THAN EOI ON A READ-ONLY FILE (SEGMENT)

- DISK READ ERROR

THESE ERRORS EITHER ORIGINATE IN NOS/VE MONITOR OR CAUSE THE
HARDWARE TO EXCHANGE TO MONITOR. DEPENDING ON THE VALUES OF
CERTAIN SYSTEM ATTRIBUTES, MONITOR WILL HALT OR REFLECT THE ERROR
BACK TO JOB MODE AS A MONITOR DEFAULT.

IT IS AT THIS POINT THAT THE SYSTEM CORE DEBUGGER CAN BE
ACTIVATED. (SEE THE DEFINITION OF SYSTEM_DEBUG_RING IN THE NEXT
SECTION.)

THE NORMAL JOB MODE OS ACTIONS FOR THESE FAULTS ARE

        BROKEN TASK   EXIT
        MCR FAULT   CAUSE CONDITION
        INVALID SYSTEM REQUEST   EXIT
        SEGMENT ACCESS   CAUSE CONDITION

7.3 SYSTEM_ATTRIBUTES_FOR_ERROR_PROCESSING

THE FOLLOWING SYSTEM ATTRIBUTES CAN BE SET OR DISPLAYED BY THE
SETSA AND DISSA COMMANDS.

7.3.1 HALTRING

IF A BROKEN TASK OR MCR FAULT OCCURS AT OR BELOW THE VALUE OF
HALTRING (P REGISTER RING NUMBER), NOS/VE MONITOR WILL HALT THE
SYSTEM. BROKEN TASKS OCCURRING ABOVE HALTRING WILL CAUSE A
MONITOR FAULT TO BE SENT BACK TO JOB MODE.

7.3.2 SYSTEM_ERROR_HANG_COUNT

THIS IS THE NUMBER OF BROKEN TASK ERRORS ALLOWED TO OCCUR IN
ANY GIVEN TASK BEFORE THAT TASK IS CONSIDERED A HUNG TASK.

7.3.3 HALT_ON_HUNG_TASK

IF THIS ATTRIBUTE IS TRUE, THEN AN OCCURENCE OF A HUNG TASK
WILL CAUSE NOS/VE MONITOR TO HALT THE SYSTEM. IF THE ATTRIBUTE
IS FALSE, THE TASK WILL BE SENT A SIGNAL TO 'HANG' ITSELF, I.E.
TO GO INTO AN INFINITE WAIT DOING NOTHING. JOBS WITH HUNG TASKS
WILL HAVE A *H IN THE STATUS FIELD ON THE OPERATOR CP DISPLAY.

A HUNG TASK WILL ALSO OCCUR IF ANY ERROR HAPPENS IN JOB MODE
RING 1.

7.3.4 SYSTEM_DEBUG_RING

IF AN ERROR (BROKEN TASK, MCR FAULT, UNKNOWN SYSTEM REQUEST, OR SEGMENT ACCESS FAULT) OCCURS AT OR BELOW THE VALUE OF THIS ATTRIBUTE (P REGISTER RING NUMBER), THE SYSTEM CORE DEBUGGER WILL BE INVOKED WITHIN THE TASK. AT THAT POINT IN TIME THE TASK ENVIRONMENT CAN BE EXAMINED BY USING SYSTEM CORE DEBUGGER COMMANDS.

IF THE RUN COMMAND IS ISSUED TO THE DEBUGGER, THE SYSTEM WILL TAKE ITS NORMAL ACTION FOR THE SPECIFIC FAULT.

7.3.5 DUMP_WHEN_DEBUG

WHEN THE SYSTEM CORE DEBUGGER IS INVOKED BY A FAULT AT OR BELOW SYSTEM_DEBUG_RING AND THE DUMP_WHEN_DEBUG ATTRIBUTE IS TRUE, THE DEBUGGER WILL AUTOMATICALLY CREATE A DUMP OF THE TASK (SEE SYSTEM CORE DEBUGGER COMMAND FORMAT). WHEN THE DUMP IS COMPLETE, NORMAL FAULT ACTION WILL TAKE PLACE. THE FOLLOWING SYSTEM CORE DEBUGGER COMMANDS ARE EXECUTED DURING AN AUTOMATIC DUMP

```
FORMAT AUTOMATIC DUMP DATA
DISXCB
DISMF
TB 1 1000
DISTE
DISSE
DISJL JOB 0 100000
DM 00300000000 10000(16)
DM 00400000000 10000(16)
DM 00500000000 10000(16)
DM 00600000000 10000(16)
DISTE
UNFORMAT
```

THIS LIST CAN BE DISPLAYED OR CHANGED USING THE DISACL AND SETACL COMMANDS.

8.0 STAND_ALONE_DEADSTART

A STANDALONE DEADSTART TAPE IS A 9-TRACK, UNLABELED, I FORMAT, 1600 BPI DENSITY TAPE (PRODUCED BY THE NVESYS PROCEDURE). TO DEADSTART IN STANDALONE MODE, PERFORM THE FOLLOWING STEPS

1) SET THE DEADSTART PANEL FOR STANDARD DISK DEADSTART.

2) PERFORM AN ALTERNATE DEADSTART TO THE NVE DEADSTART TAPE USING CTI. THE CTI PARAMETERS ON THE 'P' DISPLAY SHOULD BE SET AS FOLLOWS
   L = 0
   C = <'C' PARAMETER OF THE SETVE PROCEDURE>
   D = <'D' PARAMETER OF THE SETVE PROCEDURE
       D = Y WHEN DISPLAY IS TRUE
       D = N WHEN NO DISPLAY>
   W14 = <'CH' PARAMETER OF THE SETVE PROCEDURE>

3) THE TAPE SHOULD MOVE AND EVENTUALLY THE MESSAGE "PROCEED" WILL APPEAR ON THE LOWER LEFT ON THE CONSOLE.

4) ENTER CR,MX,BR=0000000000000000 (EDITOR'S NOTE: 16 ZEROS!)

   WHERE MX IS M1 FOR S1
               M2 FOR S2
               M3 FOR S3

   THIS CLEARS THE MEMORY BOUNDS REGISTER WHICH IS SET BY THE NOS DEADSTART.

5) ENTER DR,P2 THIS DISPLAYS THE CPU REGISTERS. THE SIT SHOULD BE CHANGING; IF NOT, GIVE UP.

6) AT THIS POINT THE SYSTEM CORE IS LOADED AND CAN BE PATCHED. THE COMMANDS TO DISPLAY AND CHANGE MEMORY ARE DOCUMENTED IN IPNDOC.

7) TO START THE SYSTEM ENTER SS THE CPU REGISTERS SHOULD SPIN; IF THEY EVER STOP THE CPU HAS HALTED.

8) ENTER DD TO DISPLAY THE SYSTEM DAYFILE ON THE RIGHT SCREEN.

9) ENTER NVE DEADSTART COMMANDS WHEN THE SYSTEM REQUESTS THEM. ('D = Y' MUST BE SPECIFIED ON THE 'P' DISPLAY FOR THIS TO HAPPEN.)

   NOTE: STANDALONE DEADSTART IS ALWAYS A 'QUICK' DEADSTART, SO NO RECOVERY OF PERMANENT FILES IS POSSIBLE. 'QUICKDS' MODE IS SET INTERNALLY AND NEED NOT BE SPECIFIED.

10) ENTER DJ TO DISPLAY THE SYSTEM JOB DAYFILE WHEN THE SYSTEM

REQUESTS IT. THE 'DJ' DISPLAY IS ON THE LEFT SCREEN.

11) NOW THE SYSTEM IS OPERATIONAL AND COMMANDS CAN BE ENTERED.

9.0 INTERACTIVE PROJECT DUMP ANALYSIS PROCEDURES

THE FOLLOWING PROCEDURES WERE DEVELOPED BY THE INTERACTIVE PROJECT TO ASSIST THEM IN INTERPRETING DUMPS. THEY GUARANTEE THE PROCEDURES WORK IF YOUR USER NAME IS IFP; OTHERWISE CAVEAT EMPTOR. FOR MORE INFORMATION ABOUT THESE PROCEDURES, CONTACT FRED BISCHKE.

THE FOLLOWING DUMP ANALYSIS PROCEDURES ARE AVAILABLE IN THE IFP CATALOG

9.1 EDDSIM

THIS IS A CCL PROCEDURE WHICH BRINGS AN EDD DUMP TAPE ON A SPECIFIED

VSN INTO THE SIMULATOR. THE PROCEDURE CAN BE ACCESSED FROM THE IFP

CATALOG AS FOLLOWS

 GET,EDDSIM/UN=IFP

 BEGIN,,EDDSIM,VSN (VSN IS THE VSN OF THE EDD DUMP TAPE)

9.2 ANALEXC

THIS IS A SIMULATOR INCLUDE FILE WHICH DOES A PRELIMINARY ANALYSIS OF

THE CURRENT SIMULATOR EXCHANGE PACKAGE (WHEN THE SYSTEM CRASHES IN TASK

SERVICES, THIS WILL NORMALLY BE JPS). A QR EXC=MON OR QR EXC=RMA CAN BE

USED TO GET INTO ANOTHER EXCHANGE PACKAGE BEFORE DOING THE INCLUDE.

THE INCLUDE FILE IS ANALEXC/UN=IFP. IT CAN BE CALLED FROM THE SIMULATOR

AS FOLLOWS

'GET,ANALEXC/UN=IFP' ; INCLUDE ANALEXC

(CARRIAGE RETURN) A LONE CARRIAGE RETURN MUST BE ENTERED AFTER AN INCLUDE

IN ORDER TO START IT UP

9.3 SEGDUMP

THIS IS A CCL PROCEDURE WHICH CALLS DSDIV TO DUMP A SPECIFIED SEGMENT TO

A LIST FILE WHICH CAN THEN BE EXAMINED WITH AN EDITOR OR PRINTED.

THE PROCEDURE CAN BE ACCESSED FROM WITHIN THE SIMULATOR AS FOLLOWS

'GET,SEGDUMP/UN=IFP' ; 'BEGIN,,SEGDUMP,SEG,LEN,FILE,EXC,CPF'

THE SEGDUMP PARAMETERS ARE

SEG - SEGMENT NUMBER IN HEX (DEFAULT IS 1)

LENGTH - NUMBER OF BYTES TO DUMP IN HEX (DEFAULT IS 10000)

LIST - NAME OF THE LIST FILE (DEFAULT IS LIST)

EXC - REFERENCE EXCHANGE PACKAGE (DEFAULT IS JPS)

CPF - NAME OF CHECKPOINT FILE (DEFAULT IS CPF)

IN MOST CASES OF TASK SERVICES DEBUGGING, ONLY THE SEG PARAMETER IS

NEEDED.

9.4 ANALJOB

THIS IS A CCL PROCEDURE WHICH USES DSDIV, XEDIT AND THE

SIMULATOR TO

PERFORM AN ANALYSIS OF ALL TASKS IN A SPECIFIED JOB. THE PROCEDURE CAN

BE ACCESSED FROM WITHIN THE SIMULATOR AS FOLLOWS

'GET,ANALJOB/UN=IFP' ; 'BEGIN,,ANALJOB,SEG,CPF'

THE ANALJOB PARAMETERS ARE

SEG - THE MONITOR SEGMENT WHICH CONTAINS THE EXCHANGE PACKAGES OF THE JOB (14 IS THE SYSTEM JOB, 15 IS JOB 1 ETC.) (DEFAULT IS 14)

CPF - THE NAME OF THE SIMULATOR FILE (DEFAULT IS CPF)

AFTER THE PROCEDURE HAS COMPLETED, A LIST OF THE RMA'S OF THE

JOB'S EXCHANGE PACKAGES CAN BE OBTAINED BY DOING THE FOLLOWING

INCLUDE TPLIST

(CARRIAGE RETURN)

A TRACEBACK OF ALL TASKS IN THE JOB CAN BE OBTAINED BY DOING THE

FOLLOWING

INCLUDE TBLIST

(CARRIAGE RETURN)

INCLUDE TBRUN

(CARRIAGE RETURN)

1.0 HARDWARE OVERVIEW

    1.1 AN INTRODUCTION TO CYBER 180

    1.2 C180 INSTANT

    1.3 MODEL INDEPENDENT GENERAL DESIGN SPECIFICATION - ARH1700

2.0 NOS REFERENCE MANUALS

    2.1 XEDIT V3.0 - 60455730

    2.2 IAF V1.0 USER'S GUIDE - 60455260

    2.3 NOS VERSION 2 REFERENCE SET - VOL 3, 60459680 - VOL 4, 60459690

    2.4 NOS SYSTEMS PROGRAMMER'S INSTANT - 60459370

    2.5 NOS VERSION 2 OPERATOR/ANALYST HANDBOOK - 60459310

    2.6 NOS VERSION 2 DIAGNOSTIC INDEX - 60459390

    2.7 NOS A170 ERS

    2.8 NOS A170 GID - ARH3060

3.0 NOS/VE REFERENCE DOCUMENTS

    3.1 PROGRAM INTERFACE ERS - ARH3610 - OBTAINED FROM KAREN RUBEY (482-3966) OR VIA SES.TOOLDOC

    3.2 COMMAND INTERFACE ERS - ARH3609 - OBTAINED FROM KAREN RUBEY (482-3966) OR VIA SES.TOOLDOC

    3.3 NOS/VE PROCEDURES AND CONVENTIONS - SESD010 - OBTAINED BY SES.TOOLDOC

    3.4 LISTING OF ALL NOS/VE MODULES - OBTAINED BY SES,DEV1.LISTNVE. SEE INTEGRATION PROCEDURES NOTEBOOK FOR DETAILS.

    3.5 NOS/VE INTERNAL INTERFACE MAINTENANCE PROCEDURES MEMO AVAILABLE FROM S.C. WOOD.

    3.6 INTEGRATION PROCEDURES NOTEBOOK OBTAINED BY ACQUIRE,IPNDOC/UN=DEV1. SES.PRINT,IPNDOC.

    3.7 NOS/VE INSTALLERS HANDBOOK - OBTAINED BY ATTACH,IHB/UN=MDC. SES.PRINT IHB.

4.0 TOOLS REFERENCE DOCUMENTS

4.1 CYBIL INTERACTIVE DEBUGGER - ARH3142

4.2 SES USER'S GUIDE - ARH1833

4.3 CYBIL SPECIFICATION - ARH2298

4.4 C180 ASSEMBLER ERS - ARH1693

4.5 SIMULATOR ERS - ARH1729

4.6 VEGEN ERS - ARH2591

4.7 VELINK ERS - ARH2816

4.8 SIMULATED I/O ERS - ARH3125

4.9 OBJECT CODE UTILITIES ERS - ARH2922

4.10 CYBIL IMPLEMENTATION DEPENDENT HANDBOOK - ARH3078

4.11 CYBER 180 INTERACTIVE DEBUG EXTERNAL REFERENCE SPECIFICATION AND USERS GUIDE - S4028

4.12 CYBER 180 II ASSEMBLER ERS - ARH3945

4.13 ERS FOR SOURCE CODE UTILITY - ARH3883

D1.0 KEYPOINTS

D1.0 KEYPOINTS

        KEYPOINTS ARE USED TO GIVE AN EXECUTION TIME TRACE OF
PROGRAM FLOW BY SHOWING THAT A GIVEN FUNCTION IS BEING PERFORMED
(THAT IS, THAT A GIVEN PROCEDURE IS BEING EXECUTED) .
KEYPOINTS MAY ALSO BE USED TO DISPLAY REQUEST PARAMETERS,
STATUS AND ERROR CONDITIONS.

D1.1 ISSUING_KEYPOINTS_FROM_CYBIL_CODE

        THE GENERAL FORM OF THE KEYPOINT INSTRUCTION IS

   #INLINE ('KEYPOINT', KEYPOINT_CLASS, OSK$M * DATA, KEYPOINT_ID);

D1.1.1 KEYPOINT CLASSES

        A KEYPOINT IS IDENTIFIED BY BOTH CLASS, AND IDENTIFIER.
        THE FOLLOWING DECK EXPLAINS THE PARTITIONING OF THE KEYPOINT
CLASSES.

OSDKEYS
COMMON

   CONST

0 KEYPOINT CLASSES
0
0   THE 16 KEYPOINT CLASSES SUPPORTED BY THE HARDWARE ARE
0 PARTITIONED BETWEEN THE SYSTEM, PRODUCT SET AND USER AS FOLLOWS.

     OSK$SYSTEM_CLASS = 0 0 0 .. 5 2,
     OSK$PRODUCT_SET_CLASS = 6 0 6 .. 10 2,
     OSK$USER_CLASS = 11 0 11 .. 14 2,
     OSK$PMF_CONTROL = 15;

0 KEYPOINT MULTIPLIER
0
0   BY CONVENTION,
0 THE 32 BIT KEYPOINT CODE SUPPORTED BY THE HARDWARE
0 IS SPLIT INTO TWO FIELDS. THE RIGHT FIELD CONTAINS A KEYPOINT

D1.0 KEYPOINTS
D1.1.1 KEYPOINT CLASSES

O IDENTIFIER WHICH IS USED TO IDENTIFY A FUNCTION WITHIN A
O KEYPOINT CLASS.   FOR EXAMPLE, IF A PARTICULAR KEYPOINT CLASS
O REPRESENTS EXIT FROM A PROCEDURE,
O THEN THE KEYPOINT IDENTIFIER MIGHT IDENTIFY EXIT FROM
O PROCEDURE A VERSUS EXIT FROM PROCEDURE B.
O   THE LEFT FIELD IS USED AS A DATA PARAMETER APPROPRIATE TO THE
O FUNCTION IDENTIFIED BY THE KEYPOINT IDENTIFIER. IN THE
O PROCEDURE EXIT EXAMPLE ABOVE,
O THE DATA PARAMETER FIELD MIGHT BE USED TO INDICATE THE
O STATUS OF THE PROCEDURE CALL.
O   THE KEYPOINT MULTIPLIER IS USED TO PARTITION THE KEYPOINT
O CODE INTO THE TWO FIELDS. THE DATA PARAMETER SHOULD BE
O MULTIPLIED BY THE KEYPOINT MULTIPLIER TO PREVENT IT FROM
O OVERLAPPING THE KEYPOINT IDENTIFIER FIELD.

```
CONST
  OSK$M = 4096;
```

D1.1.2 NOS/VE KEYPOINT CLASSES

        FIVE KEYPOINT CLASSES NAMED ENTRY, EXIT, UNUSUAL, DEBUG,
AND DATA ARE DEFINED, TAKING FIVE OF THE AVAILABLE SIXTEEN CLASSES
BY THE HARDWARE.

    ENTRY — EVERY GATED PROCEDURE PLUS ALL MAJOR INTERNAL PROCEDURES
          (THOSE SHARED ACROSS FUNCTIONAL AREAS) SHOULD CONTAIN A
          KEYPOINT OF THIS CLASS. THESE KEYPOINTS SHOULD BE PLACED
          AS CLOSE AS POSSIBLE TO THE ENTRY TO THE PROCEDURE.

    EXIT — EVERY GATED PROCEDURE PLUS ALL MAJOR INTERNAL PROCEDURES
          (THOSE SHARED ACCROSS FUNCTIONAL AREAS) SHOULD CONTAIN A
          KEYPOINT OF THIS CLASS. THESE KEYPOINTS SHOULD BE PLACED
          AS CLOSED AS POSSIBLE TO THE EXIT TO THE PROCEDURE.

    UNUSUAL — EVERY SITUATION WHICH IS UNEXPECTED OR QUITE UNUSUAL
          SHOULD CONTAIN A KEYPOINT OF THIS CLASS. IT IS INTENDED
          THAT THESE KEYPOINTS WOULD BE ENABLED AT ALL TIMES. THE
          FREQUENCY OF ENCOUNTERING THESE KEYPOINTS SHOULD BE
          VERY LOW. THE DATA KEYPOINT CLASS IS NOT ALLOWED IN
          CONJUNCTION WITH A KEYPOINT OF CLASS UNUSUAL.

    DEBUG — THESE KEYPOINTS ARE FOR PROVIDING ADDITIONAL TRACE
          INFORMATION AS AN ASSIST IN DEBUGGING HARDWARE OR SOFTWARE
          PROBLEMS. DEBUG CLASS KEYPOINTS WOULD BE MOST USEFUL IN
          THE MORE COMPLEX AREAS OF THE SYSTEM.

    DATA — THIS KEYPOINT CLASS CAN BE USED WITH ENTRY, EXIT, AND

D1.0 KEYPOINTS
D1.1.2 NOS/VE KEYPOINT CLASSES

DEBUG KEYPOINTS FOR THE GATHERING OF EXTRA DATA. ALL DATA
KEYPOINTS ENCOUNTERED ARE SUPPLYING ADDITIONAL DATA WHICH
WILL BE ASSOCIATED WITH THE LAST ENTRY, EXIT, OR DEBUG
KEYPOINT.
DATA KEYPOINTS SHOULD BE USED WITH CARE SINCE THE PMF
HARDWARE CAN ONLY BUFFER UP 16 KEYPOINTS, KEYPOINT CLUSTEG
CAN CAUSE LOST KEYPOINTS.

THE FOLLOWING DECK DEFINES THE NOS/VE OS CLASS CONSTANTS.

OSDKEYC
COMMON
ODEFINE KEYPOINT CLASS CODES.

```
  CONST
    OSK$DATA = OSK$SYSTEM_CLASS + 0, 0 OS - DATA KEYPOINT2
    OSK$UNUSUAL = OSK$SYSTEM_CLASS + 1, OU OS - UNUSUAL KEYPOINT.2
    OSK$ENTRY = OSK$SYSTEM_CLASS + 2, OE OS - ENTRY KEYPOINT)2
    OSK$EXIT = OSK$SYSTEM_CLASS + 3, OX OS - EXIT KEYPOINT2
    OSK$DEBUG = OSK$SYSTEM_CLASS + 4; OD OS - DEBUG KEYPOINT.2
```

O*CALLC,OSDKEYS

D1.1.3 KEYPOINT DATA AND IDENTIFICATION

UPON SUCCESSFUL EXECUTION EACH KEYPOINT INSTRUCION WILL
PROVIDE A TOTAL OF 32 BITS OF INFORMATION. OUR CONVENTION USES
12 BITS OF THIS FOR KEYPOINT IDENTIFICATION AND THE REMAINING 20
BITS AS USER SUPPLIED DATA. TRY TO USE THIS 20 BITS TO SUPPLY
MEANINGFUL INFORMATION (TASKID, SEGMENT NUMBER, FILE IDENTIFIER,
QUEUE LENGTH, PAGE NUMBER, TIME, ETC.). THE KEYPOINT
IDENTIFICATION CODES ARE DEFINED IN THE ATTACHED COMMON DECK. ON
DATA CLASS KEYPOINTS THE DATA BELONGS TO THE PREVIOUS KEYPOINT
AND THE FULL 32 BITS IS AVAILABLE FOR ADDITIONAL USER DATA.

D1.1.4 EXAMPLE ISSUING KEYPOINTS

ENTRY KEYPOINT WITH DATA

```
  #INLINE('KEYPOINT', OSK$ENTRY, OSK$M*TASKID.INDEX,
    TMK$EXIT_TASK);
```


UNUSUAL KEYPOINT WITH NO DATA

D1.0 KEYPOINTS
D1.1.4 EXAMPLE ISSUING KEYPOINTS

```
    #INLINE ('KEYPOINT', OSK$UNUSUAL, O, MMK$NO_MEMORY);
```

ENTRY KEYPOINT WITH EXTRA DATA

```
    #INLINE ('KEYPOINT', OSK$ENTRY, OSK$M * SEGMENT_NUMBER,
      MMK$PAGE_FAULT);
    #INLINE ('KEYPOINT', OSK#DATA, OFFSET, O);
```


D1.1.5 KEYPOINT IDENTIFIERS

EACH AREA OF THE OPERATING SYSTEM HAS BEEN GIVEN A RANGE OF
IDENTIFIERS TO USE FOR KEYPOINTS. THE BASE FOR EACH AREA IS
DEFINED ON COMMON DECK OSDKEYD. EACH AREA SHOULD
HAVE A DECK XXDKEY (WHERE XX IS THE PRODUCT IDENTIFIER)
WHERE THE AREAS KEYPOINT CONSTANTS ARE DEFINED(E.G.TMK$EXIT_TASK).
PLEASE REFERENCE THE SECTION ON KEYPOINT DESCRIPTION DECKS, FOR AN
EXAMPLE OF ONE OF THESE DECKS.

OSDKEYD
COMMON
OTHIS DECK DEFINES CONSTANTS FOR USE WITH KEYPOINTS.

ODEFINE BASE KEYPOINT PROCEDURE IDENTIFIERS FOR EACH AREA OF THE
OOS.

```
    CONST
      AMK$BASE = 100,  0100 - 1492
      BAK$BASE = 200,  0200 - 2492
      CLK$BASE = 250,  0250 - 2992
      CMK$BASE = 300,  0300 - 3492
      DBK$BASE = 350,  0350 - 3992
      DMK$BASE = 400,  0400 - 5492
      FMK$BASE = 550,  0550 - 5992
      ICK$BASE = 600,  0600 - 6492
      IFK$BASE = 650,  0650 - 6992
      IIK$BASE = 700,  0700 - 7492
      INK$BASE = 750,  0750 - 7992
      JMK$BASE = 800,  0800 - 8492
      LGK$BASE = 850,  0850 - 8992
      LLK$BASE = 900,  0900 - 9492
      LOK$BASE = 950,  0950 - 9992
      LUK$BASE = 1000, 01000 - 10492
      MLK$BASE = 1050, 01050 - 10992
      MMK$MONITOR_BASE = 1100, 01100 - 11492
      MMK$JOB_BASE = 1150, 01150 - 11992
```

D1.0 KEYPOINTS
D1.1.5 KEYPOINT IDENTIFIERS

```
      MSK$BASE = 1200, 01200 - 12492
      MTK$BASE = 1250, 01250 - 12992
      OCK$BASE = 1300, 01300 - 13492
      OFK$BASE = 1350, 01350 - 13992
      OSK$BASE = 1400, 01400 - 14492
      PFK$BASE = 1500, 01500 - 15492
      PMK$BASE = 1600, 01600 - 16992
      RHK$BASE = 1750, 01750 - 17992
      SRK$BASE = 1800, 01800 - 18192
      STK$BASE = 1850, 01850 - 18992
      TMK$MONITOR_BASE = 1900, 01900 - 19492
      TMK$JOB_BASE = 1950, 01950 - 19992
      JSK$MONITOR_BASE = 2000, 02000 - 20492
      JSK$JOB_BASE = 2050, 02050 - 20992
      AVK$BASE = 2100, 02100 - 21492
      SFK$BASE = 2150, 02150 - 21992
      IOK$BASE = 2200, 02200 - 22492
      RMK$BASE = 2250, 02250 - 23002
      MTK$ASSEMBLY_LANGUAGE_BASE = 4000; 04000 - 40952
O     OS ASSEMBLY LANGUAGE        4000 - 40952
O*CALLC,OSDKEYC
```

D1.2 COLLECTING_KEYPOINTS

D1.2.1 ON THE SIMULATOR

    WHEN EXECUTING ON THE SIMULATOR ALL KEYPOINT INSTRUCTIONS CAUSE
AN ENTRY TO BE ADDED TO THE LOCAL FILE SESSMKF.

D1.2.2 ON THE HARDWARE

    SOFTWARE KEYPOINT COLLECTION IS AVAILABLE FOR COLLECTING SYSTEM
AND JOB KEYPOINTS. SYSTEM KEYPOINTS ARE THOSE KEYPOINTS IN THE
ENTIRE SYSTEM AND JOB KEYPOINTS ARE ONLY THOSE DEALING WITH A
PARTICULAR JOB. ONLY ONE SYSTEM KEYPOINT COLLECTOR
CAN BE ACTIVE AT ONE TIME, BUT EACH JOB MAY HAVE AN ACTIVE
JOB KEYPOINT COLLECTOR. SOFTWARE KEYPOINTS ARE COLLECTED ON A
FILE LOCAL TO THE JOB IN WHICH THE KEYPOINT COLLECTION TASK IS
RUNNING. AFTER KEYPOINT COLLECTION IS TERMINATED THIS FILE CAN
SAVED ON THE 170 SIDE AND ANALYZED BY THE KEYPOINT ANALYZER.

D1.0 KEYPOINTS
D1.2.2 ON THE HARDWARE

    THREE COMMANDS ARE SUPPLIED TO UTILIZE THE KEYPOINT FEATURE
ACTK, DEAK, AND EMIK.

    D1.2.2.1 ACTK_COMMAND

    THE ACTK COMMAND INITIATES KEYPOINT RECORDING AND COLLECTING.
IT HAS THE FORM OF

    ACTK,MODE,ENVIRONMENT,MONITOR_MASK,JOB_MASK,
        START_CLASS,STOP_CLASS,KEYPOINT_FILE,
        COLLECTOR_BUFFER_SIZE,COLLECTOR_DELAY

    MODE OR M = 'SOFTWARE', 'S',   PARAMETER IS
            'HARDWARE', OR 'H'   REQUIRED

    ENVIRONMENT OR E = 'JOB','J',   PARAMETER IS
            'SYSTEM', OR 'S'   REQUIRED

    MONITOR_MASK OR MM = A LIST OF INTEGERS   DEFAULT IS ALL
            RANGING FROM 0 TO 15
            EXAMPLE: MM=(2,6,9,12)

    JOB_MASK OR JM = A LIST OF INTEGERS   DEFAULT IS ALL
            RANGING FROM 0 TO 15
            EXAMPLE: JM=(0,3,5)

    START_CLASS OR START = AN INTEGER IN   DEFAULT IS 15
            THE RANGE FROM 0 TO 15
            THIS SPECIFIES THAT KEYPOINT COLLECTION
            SHOULD NOT START UNTIL A KEYPOINT OF THIS
            CLASS IS ENCOUNTERED.

    STOP_CLASS OR STOP = AN INTEGER IN   DEFAULT IS 15
            THE RANGE FROM 0 TO 15
            THIS SPECIFIES THAT KEYPOINT COLLECTION
            SHOULD STOP WHEN A KEYPOINT OF THIS
            CLASS IS ENCOUNTERED.

    KEYPOINT_FILE OR KF = FILE NAME   DEFAULT IS
            THIS SPECIFIES THE FILE ON WHICH $LOCAL.KEYFILE
            KEYPOINTS ARE TO BE SAVED.
            THIS PARAMETER IS USED
            ONLY WITH SOFTWARE KEYPOINTS.

    COLLECTOR_BUFFER_SIZE OR CBS = AN INTEGER DEFAULT IS 10,000
            IN THE RANGE FROM 1000 TO 100,000

D1.0 KEYPOINTS
D1.2.2.1 ACTK COMMAND

```
                    THIS PARAMETER IS USED ONLY WITH
                    SOFTWARE KEYPOINTS.
        COLLECTOR_DELAY OR CD = AN INTEGER          DEFAULT IS 50
                    IN THE RANGE FROM 10 TO 100,000
                    THE VALUE SPECIFIED IS THE DELAY
                    PERIOD IN MILLISECONDS.  THIS
                    PARAMETER IS USED ONLY WITH
                    SOFTWARE KEYPOINTS.
```

D1.2.2.2 DEAK_COMMAND

THE DEAK COMMAND TERMINATES KEYPOINT COLLECTION.

DEAK,ENVIRONMENT

```
ENVIRONMENT OR E = 'JOB','J',              DEFAULT IS JOB
        'SYSTEM', OR 'S'
```

D1.2.2.3 EMIK_COMMAND

THE EMIK COMMAND IS USED TO ISSUE KEYPOINTS.

EMIK,CLASS,CODE

```
CLASS = AN INTEGER IN THE RANGE            DEFAULT IS 15
        FROM 0 TO 15

CODE = AN INTEGER IN THE RANGE             DEFAULT IS 0
        FROM 0 TO 0FFFFFFFF(16)
```

AFTER KEYPOINT COLLECTION IS TERMINATED THE KEYPOINT FILE,
CAN BE SAVED ON THE 170 BY A REPLACE_FILE WITH B56
CONVERSION. FOR EXAMPLE.
    REPLACE_FILE,KEYFILE,KEYFILE,B56

ON THE 170 SIDE THIS CAN BE ANALYZED BY USING NVEKEY,
FORMAT = HDW.

D1.0 KEYPOINTS
D1.3 KEYPOINT ANALYZER UTILITY

D1.3 KEYPOINT_ANALYZER_UTILITY

D1.3.1 NVEKEY

THE SESSMKF FILE PRODUCED ON THE SIMULATOR, OR THE
KEYFILE PRODUCED ON THE HARDWARE CAN BE REFORMATTED INTO A
READABLE LISTING BY EXECUTING THE FOLLOWING PROCEDURE.

SES.NVEKEY [KPF=] [FORMAT=] [KD=] [AREA=]
NVEKEY CREATES A SIMULATOR GENERATED KEYPOINT TRACE FILE.
THE "KPF" PARAMETER IS THE KEYPOINT FILE USED AS INPUT.
THE "KD" PARAMETER IS A FILE OR LIST OF FILES WHICH DEFINE(S)
THE KEYPOINT DESCRIPTIONS.

----PARAMETER-------DEFAULT--------ALLOWABLE VALUES------
     KPF   'SESSMKF'   FILE NAME
                      'KEYFILE'   IF FORMAT=HDW
     FORMAT   'SIM'   SIM,HDW
     KD   'KEYDESC'   FILE NAME(S)
     AREA   &USER&   USER NAME

IF RUN INTERACTIVELY, WHEN THE PROCEDURE TERMINATES THE
REFORMATTED LISTING IS ON LOCAL FILE KEYFILE.

D1.3.2 KEYPOINT DESCRIPTION FILE

THE KEYPOINT DESCRIPTIONS ARE USED BY THE KEYPOINT
ANALYZER UTILITY TO DIRECT THE REFORMATTING OF THE
KEYPOINT INFORMATION.

D1.3.2.1 KEYPOINT_DECKS

EACH AREA HAS A KEYPOINT CONSTANT DECK XXDKEY (WHERE XX
IS THE PRODUCT ID). THE KEYPOINT DESCRIPTIONS ARE NOW
INCLUDED IN THIS DECK IMMEDIATELY FOLLOWING THE KEYPOINT
CONSTANTS (SIMILIAR TO THE MESSAGE TEMPLATES).

EACH DESCRIPTION HAS THE FOLLOWING FORMAT.
NOTE: EACH ELEMENT (IF GIVEN) IS POSITIONALLY DEPENDANT.

OCLASS [SUB_ID_FIELD] KEYPOINT_LABEL [DATA_LABEL] [DATA_FIELD]

CYCLE 11, JANUARY 1983

D1.0 KEYPOINTS
D1.3.2.1 KEYPOINT DECKS

       CLASS OF KEYPOINT - REQUIRED
          E ENTRY
          X EXIT
          U UNUSUAL
          D DEBUG

       SUB_ID_FIELD - OPTIONAL - (DESCRIBED LATER)

       KEYPOINT_LABEL - REQUIRED - THIS IS A STRING THAT
          DESCRIBES THE PURPOSE OF THE KEYPOINT.

       DATA_LABEL - OPTIONAL - THIS IS A STRING OF UP TO 8
          CHARACTERS DESCRIBING THE DATA PORTION OF THE KEYPOINT.

       DATA_FIELD_DESCRIPTOR - OPTIONAL - THIS CONSISTS OF DATA
          FORMAT AND LENGTH.
          DATA_FORMAT
             H HEX
             I INTEGER (DECIMAL)
             A ASCII
          CONCATENATED TO THIS IS THE LENGTH OF THE DATA PORTION OF
          THE KEYPOINT, IN DECIMAL BITS.
          FOR EXAMPLE: I20

       D1.3.2.1.1 EXAMPLE KEYPOINT DECK

       STDKEY
       COMMON

       O PURPOSE
       O   THIS DECK CONTAINS ALL OF THE SET MANAGER KEYPOINT CONSTANTS.

          CONST
             STK$CREATE_SET = STK$BASE + 1,
                OE 'STP$CREATE_SET' 'RING   ' H 2
                OX 'STP$CREATE_SET' 'STATUS   ' I20 2

              STK$PURGE_SET = STK$BASE + 2,
                OE 'STP$PURGE_SET' 2
                OX 'STP$PURGE_SET' 'STATUS   ' I20 2

             STK$CANT_DM_STORE_SET_ORD = STK$BASE,
                OU   'CANT DMP$STORE_AVT_SET_ORDINAL' 'AVTINDX ' H20 2

             STK$PF_ROOT_SIZE = STK$BASE + 5;
                OD 'PF_ROOT_SIZE' 'ROOTSIZ ' H20 2

D1.0 KEYPOINTS
D1.3.2.1.1 EXAMPLE KEYPOINT DECK

```
?? PUSH (LIST := OFF) ??
O*CALLC OSDKEYD
?? POP ??
```


D1.3.2.1.2 SUB_ID_FIELD

THIS OPTIONAL FIELD ALLOWS A MEANS OF SUBDIVIDING A SINGLE
KEYPOINT INTO SEVERAL DESCRIPTORS. THE PARTICULAR DESCRIPTOR
IS CHOSEN ON THE BASIS OF A SELECTABLE NUMBER OF BITS OF THE
DATA FIELD. THIS FIELD HAS THE FOLLOWING FORMAT

   SUB_ID_LENGTH.SUB_ID_MATCH

  SUB_ID_LENGTH - THIS SPECIFIES THE NUMBER OF BITS (RIGHT MOST)
     OF THE DATA FIELD TO USE, TO DETERMINE WHICH
     DESCRIPTOR TO CHOOSE.

  SUB_ID_MATCH - THIS SPECIFIES THE INTEGER IDENTIFIER USED TO
     MATCH THE DATA PORTION.

 EXAMPLE
   MMK$PAGE_FAULT = MMK$MONITOR_BASE + 6,
      OE 'PAGE FAULT PROCESSOR' 2
      OE 4.1 'PAGE FOUND IN AVAIL QUEUE' 'PFTI   ' H162
      OE 4.2 'PAGE FOUND IN AVAIL MODIFIED QUEUE ' 'PFTI   ' H162

      IF THIS KEYPOINT WAS ISSUED AND PRODUCED DATA OF 2, THE
      DESCRIPTOR WITH THE SUB_ID_MATCH FIELD OF 2 WOULD BE
      USED ('PAGE FOUND IN AVAIL MODIFIED QUEUE').
      THESE KEYPOINTS WERE ISSUED WITH A SUB_ID_LENGTH = 4,
      THUS THE 4.X.   FOR EXAMPLE
      #INLINE ('KEYPOINT', OSK$ENTRY, OSK$M *
        (PFTI * 16 02 ** SUB_ID_LENGTH2 + 2OSUB_ID_MATCH2),
        MMK$PAGE_FAULT);

D1.3.2.2 GENERATING_THE_DESCRIPTOR_FILE

    THE KEYPOINT DESCRIPTIONS ARE KEPT ON A FILE CALLED KEYDESC
ON THE INTEGRATION CATALOG. THIS FILE MAY BE PRODUCED BY

  SES.GENCOMP M=OSMKEYS AB=((NOSVEPL,OSLPI,INT2)) CF=KEYDESC

THE USER MAY ADD KEYPOINTS TO HER XXDKEY DECK LOCALLY, AND
THE KEYDESC FILE MAY BE PRODUCED AS ABOVE, SPECIFYING THE
ADDITIONAL LOCAL BASES. THE KEYDESC FILE MAY THEN BE SAVED

D1.0 KEYPOINTS
D1.3.2.2 GENERATING THE DESCRIPTOR FILE

ON HER CATALOG.

IF NEW KEYPOINT DECKS ARE ADDED, *CALLC 'S TO THESE NEW DECKS
CHOULD BE ADDED TO THE DECK OSMKEYS, AND THE APPROPRIATE
BASE CONSTANTS ADDED TO DECK OSDKEYD.

WHEN TRANSMITTING CHANGES TO KEYPOINT DECKS, BE SURE TO INFORM
INTEGRATION, VIA THE TRANSITTAL FORM, TO RECREATE THE FILE
KEYDESC.

D1.3.2.3 OSMKEYS_FORMAT

THIS SECTION WILL ONLY BE USEFUL TO THOSE DESIRING TO ADD
ADDITIONAL KEYPOINT CLASSES, KEYPOINT CLASS BASE CONSTANTS,
OR NEW KEYPOINT DESCRIPTION DECKS.
THE CLASSES, IDENTIFIERS, AND DESCRIPTIONS ARE EACH BUFFERED
BY A COMMENT. FOR EXAMPLE, TO ADD ANOTHER KEYPOINT CLASS
O$$$ START KEYPOINT CLASSES $$$2
CONST
   PSK$ENTRY = OSK$PRODUCT_SET_CLASS + 1; OE PS - ENTRY KEYPOINT2
O$$$ END KEYPOINT CLASSES $$$ 2
NOTE: THE E FOLLWOING THE "O" WILL BE USED IN THE DESCRIPTION.

THIS NEW SECTION SHOULD BE APPENDED TO THE END OF THE KEYDESC
FILE. READERS DESIRING MORE INFORMATION SHOULD REFERENCE THE
ATTACHED BNF, AND THE ATTACHED DECKS OSMKEYS.

THE FOLLOWING REPRESENTS A SAMPLE OF HOW TO SET UP
THE DESCRIPTION MODULE.
NOTE: COMMENT PUT AROUND *CALL FOR SAKE OF DOCUMENTATION ONLY.

```
OSMKEYS
?? LEFT := 1, RIGHT := 110 ??
MODULE KEYPOINT_DESCRIPTION_FILE;
O*CALLC, OSDKEYS
O$$$ START KEYPOINT CLASSES $$$2
O*CALLC,OSDKEYC
O$$$ END KEYPOINT CLASSES $$$2
O$$$ START KEYPOINT IDENTIFIER BASES $$$2
O*CALLC, OSDKEYD
O$$$ END KEYPOINT IDENTIFIER BASES $$$2
O$$$ START KEYPOINT DESCRIPTIONS $$$2
O*CALLC, AMDKEY
O*CALLC, BADKEY
O*CALLC, CLDKEY
O*CALLC, CMDKEY
```

D1.0 KEYPOINTS
D1.3.2.3 OSMKEYS FORMAT

```
      O*CALLC,DBDKEY
      O*CALLC,DMDKEY
      O*CALLC,FMDKEY
      O*CALLC,ICDKEY
      O*CALLC,IFDKEY
      O*CALLC,IIDKEY
      O*CALLC,INDKEY
      O*CALLC,JMDKEY
      O*CALLC,LGDKEY
      O*CALLC,LLDKEY
      O*CALLC,LODKEY
      O*CALLC,LUDKEY
      O*CALLC,MLDKEY
      O*CALLC,MMDMKEY
      O*CALLC,MMDJKEY
      O*CALLC,MSDKEY
      O*CALLC,MTDKEY
      O*CALLC,OCDKEY
      O*CALLC,OFDKEY
      O*CALLC,OSDKEY
      O*CALLC,PFDKEY
      O*CALLC,PMDKEY
      O*CALLC,RHDKEY
      O*CALLC,SRDKEY
      O*CALLC,STDKEY
      O*CALLC,TMDMKEY
      O*CALLC,TMDJKEY
      O*CALLC,JSDMKEY
      O*CALLC,JSDJKEY
      O*CALLC,AVDKEY
      O*CALLC,SFDKEY
      O*CALLC,IODKEY
      O*CALLC,RMDKEY
      O$$$ END KEYPOINT DESCRIPTIONS $$$2
      MODEND KEYPOINT_DESCRIPTION_FILE;
```


D1.4 REFORMATTED_FILE_DESCRIPTION

     THE OUTPUT FROM PROCEDURE NVEKEY IS A FILE CALLED KEYFILE.
THIS REFORMATTED LISTING CONTAINS TWO SECTIONS. THE FIRST
SECTION IS A LISTING OF ALL THE KEYPOINTS IN THE ORDER THEY WERE
ISSUED. THE SECOND SECTION IS A SUMMARY OF THE NUMBER OF TIMES
EACH KEYPOINT OCCURED.
     EACH LINE IN THE FIRST SECTION HAS THE FOLLOWING FORMAT;

   RT TSL DATA DATA_LABEL S TN AREA_ID KP_LABEL

D1.0 KEYPOINTS
D1.4 REFORMATTED FILE DESCRIPTION

THE RT FIELD DESIGNATES THE VALUE OF THE FREE RUNNING
MICROSECOND CLOCK (TIME SINCE DEADSTART) WHEN THE KEYPOINT WAS
EXECUTED. ON THE SIMULATOR THE CLOCK IS INCREMENTED BY 1 FOR
EACH INSTRUCTION EXECUTED.

THE TSL FIELD DESIGNATES THE TIME (MICROSECONDS) SINCE THE
LAST KEYPOINT INSTRUCTION WAS EXECUTED.

THE DATA FIELD SPECIFIES THE VALUE OF THE DATA PORTION OF THE
KEYPOINT IN THE FORMAT DESCRIBED IN THE KEYPOINT DESCRIPTION
FILE FOR THIS KEYPOINT.

THE DATA_LABEL FIELD IS THE DATA LABEL FIELD FROM THE
KEYPOINT DESCRIPTION FILE FOR THIS KEYPOINT.
THIS IDENTIFIES THE DATA BEING DISPLAYED.

THE S FIELD SPECIFIES THE STATE OF THE MACHINE WHEN THE
KEYPOINT WAS ISSUED AND IS ONE OF THE FOLLOWING

        M - MONITOR MODE
        J - JOB MODE

        AN * PRECEDING THE S FIELD INDICATES THAT
        TRAP PROCESSING IS ACTIVE, THAT IS THE TRAP HANDLER HAS
        BEEN ENTERED, BUT NOT EXITED.

THE TN FIELD GIVES THE GLOBAL TASK ID OF THE TASK THAT
WAS EXECUTING WHEN THE KEYPOINT WAS ISSUED.
THE SYSTEM IS TASK 1.

THE AREA_ID FIELD IS THE AREA IDENTIFIER FOR THE AREA
ISSUING THE KEYPOINT.

THE KP_LABEL IS THE KEYPOINT LABEL FIELD FROM THE KEYPOINT
DESCRIPTION FILE. THIS DESCRIBES THE KEYPOINT.

NOTE: FOR AN UNDEFINED KEYPOINT, THAT IS, ONE WHICH HAS NO
        DESCRIPTOR ENTRY, THE AREA_ID FIELD CONTAINS THE
        INTEGER FOR THE KEYPOINT CLASS, THE CLASS FIELD
        ON THE OUTPUT IS SPECIFIED AS "UND", AND THE KP_LABEL
        BECOMES THE ID_NUMBER OF THE KEYPOINT.

D1.0 KEYPOINTS
D1.5 BNF KEYPOINT DESCRIPTION

D1.5 BNF_KEYPOINT_DESCRIPTION

<ANALYZER_DESCRIPTOR_INPUT> ::= <KEYPOINT_CLASS_ALLOCATION_DECK>
                                [<DEFINITION_DECK> ...]

<KEYPOINT_CLASS_ALLOCATION_DECK> ::= <CYBIL CODE AND/OR COMMENTS>
                                [<CLASS_BASE_DEFINITIONS> ...]
                                <CYBIL CODE AND/OR COMMENTS>

<CLASS_BASE_DEFINITIONS> ::= <CLASS_BASE_ID> <SPC> = <SPC> <BASE>

<CLASS_BASE_ID> ::= OSK$SYSTEM_CLASS 1 OSK$PRODUCT_SET_CLASS 1
                    OSK$USER_CLASS 1 OSK$PMF_CONTROL

<SPC> ::= [<SPACE> ...]

<BASE>

<DEFINITION_DECK> ::= <CLASS_DEFINITION_DECK> 1
                      <BASE_DEFINITION_DECK> 1
                      <KEYPOINT_DEFINITION_DECK>

<CLASS_DEFINITION_DECK> ::= O$$$ START KEYPOINT CLASSES $$$2
                            <CYBIL CODE AND/OR COMMENTS>
                            [<CLASS_DEFINITIONS> ...]
                            <CYBIL CODE AND/OR COMMENTS>
                            O$$$ END KEYPOINT CLASSES $$$2

<CLASS_DEFINITIONS> ::= <KEYPOINT_CLASS> <SPC> = <SPC>
                        <CLASS_BASE_ID> <OFFSET>
                        0 <KEYPOINT_CLASS_ID> <CYBIL COMMENT>

<KEYPOINT_CLASS>

<OFFSET> ::= + <SPC> <INTEGER> <DELIMITER>

<DELIMITER> ::= , 1 ;

<KEYPOINT_CLASS_ID> ::= <CHARACTER>

<BASE_DEFINITION_DECK> ::=
                    O$$ START KEYPOINT IDENTIFIER BASES $$$2
                            <CYBIL CODE AND/OR COMMENTS>
                            [<RANGE_BASE_DEFINITIONS> ...]
                            <CYBIL CODE AND/OR COMMENTS>
                            O$$$ END KEYPOINT IDENTIFIER BASES $$$2

<RANGE_BASE_DEFINITIONS> ::= <KEYPOINT_BASE> <DELIMITER>

D1.0 KEYPOINTS
D1.5 BNF KEYPOINT DESCRIPTION

```
                              <BASE_RANGE>

   <KEYPOINT_BASE> ::= <SPC> <BASE_ID> <SPC> = <SPC> <BASE>

   <BASE_ID> ::= <IDENTIFIER>

   <BASE_RANGE> ::= <SPC> O <LOW_BASE> <SP> - <HIGH_BASE> [ 2 ]

   <LOW_BASE> ::= <INTEGER>

   <HIGH_BASE> ::= <INTEGER>

   <KEYPOINT_DEFINITION_DECK>
                              O$$$ START KEYPOINT DESCRIPTIONS $$$2
                                 <CYBIL CODE AND/OR COMMENTS>
                                 [ <XXDKEY_DECK> ... ]
                                 <CYBIL CODE AND OR COMMENTS>
                              O$$$ END KEYPOINT DESCRIPTIONS $$$2

   <XXDKEY_DECK> ::= [ <CYBIL CODE AND/OR COMMENTS> ]
                     [ <KEYPOINT_INFO> ... ]

   <KEYPOINT_INFO> ::= <KEYPOINT_CONSTANT_LINE> <DELIMITER> <EOL>
                              [ <KEYPOINT_DESCRIPTOR> ... ]
                              [ <BLANK LINES> ]

   <KEYPOINT_CONSTANT_LINE>  ::= <KEYPOINT_CONSTANT> <SPC> = <SPC>
                              <KEYPOINT_BASE> <SPC> [ <OFFSET> ] <SPC>

   <KEYPOINT_CONSTANT> ::= <IDENTIFIER>

   <KEYPOINT_BASE> ::= <IDENTIFIER>

   <KEYPOINT_DESCRIPTOR> ::= O <KEYPOINT_DESCRIPTOR_LIST> <SPC> [ 2 ]
                              <EOL>

   <KEYPOINT_DESCRIPTOR_LIST>
                              [ <SPECIAL_CASE_CODE> ] <SPC>
                              [ <SUB_ID_FIELD> ] <SPC> <KEYPOINT_LABEL>
                              <SPC> [ <DATA_FIELD> ]

   <SPECIAL_CASE_CODE> ::= M 1 N 1 S 1 T
            (M = MTR, N = NOS, S = TASK SWITCH, AND T = TRAP)

   <SUB_ID_FIELD> ::= <SUB_ID_LENGTH> . <SUB_ID_MATCH>

   <SUB_ID_LENGTH> ::= <FIELD_LENGTH>

   <FIELD_LENGTH> ::= 0..52                                    (IN BITS)
```

D1.0 KEYPOINTS
D1.5 BNF KEYPOINT DESCRIPTION

   <SUB_ID_MATCH> ::= <SMALL_INTEGER>

   <SMALL_INTEGER> ::= 0..0-FFFFFFFFFFFFF(16)

   <KEYPOINT_LABEL>

   <LABEL> ::= ' <CHARACTER_STRING> '

   <CHARACTER_STRING> ::= ANY VISIBLE CHARACTERS EXCEPT '

   <DATA_FIELD> ::= <DATA_LABEL> <SPC> [<DATA_FIELD_DESCRIPTOR>]

   <DATA_LABEL> ::= <LABEL>

   <DATA_FIELD_DESCRIPTOR> ::= <DATA_FORMAT> [<DATA_FIELD_LENGTH>]

   <DATA_FORMAT> ::= A 1 H 1 I
           (A = ALPHANUMERIC, H = HEX, I = INTEGER)

   <DATA_FIELD_LENGTH> ::= <FIELD_LENGTH>

   (NOTE: <SUB_ID_LENGTH> + <DATA_FIELD_LENGTH> MUST BE <= 52 BITS)
   (NOTE: OPERATING SYSTEM <KEYPOINT_CLASS_ID> = 0D,E,U,X2)
   (NOTE: <KEYPOINT_CLASS_ID> FOR ANY KEYPOINT USED FOR
   ADDITIONAL INFORMATION TO PREVIOUS KEYPOINTS
   MUST BE A SPACE)
   (NOTE: A <DEFINITION_DECK> REMAINS IN EFFECT UNTIL
   SUPERCEEDED BY A DECK WHICH REDEFINES THE
   AREA TO WHICH IT PERTAINS)

TABLE OF CONTENTS