PHINTS

BUILD P HELPFUL HINTS

This paper describes helpful hints on how to use build P of NOS/VE. It is intended to supplement, rather than to replace, the standard NOS/VE documentation. If you have any qustions or suggestions, please see Tom McGee or Bonnie Swierzbin. Appendix A lists background documents and how to obtain them.

To obtain additional copies of this document while running on SN101 at Arden Hills, please type:

```
SES,DEV1.LISTPH C=<number of copies>
```

To obtain a copy with revision bars against the Helpful Hints of the previous build, please type:

```
SES,DEV1.LISTPH REVB C=<number of copies>
```

The C parameter is optional and defaults to one.

Update History

Date	Changes
12/22/80	Revisions for NOS/VE Phase C
2/12/81	Additional revisions for NOS/VE Phase C
6/9/81	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
11/6/81	Revisions for NOS/VE Build P

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

- All existing commands have been updated to Revision 8 of the ERS. In addition, parameter abbreviations have been added for all commands. This involved some renaming of parameters from what was previously implemented, and in some cases, from what was described in the ERS.

  A description of the parameter names that have been changed and the algorithm for deriving the abbreviations can be found in the COMMAND INTERFACE STATUS section of this document.

- The long awaited message generator is available! Henceforth, in most cases, rather than a condition code and raw status record text, you will see formatted messages and their designated severity. The exceptions will be for any remaining HCS condition codes and, of course, conditions for which no message templates have been defined. Feedback with regard to the intelligibility and usefulness of the messages you encounter would be greatly appreciated. comments on this subject should be given to C.G. Nelson (ARH263, X2750) preferably in writing.

  When a job is initiated, the message mode is set to brief. This means that the identifier and condition code for a message will not be shown; i.e. you will only see a severity designator and a formatted message; the severity designator for informative messages is omitted. If you really want to see the condition code, use the SET_MESSAGE_MODE FULL command. In full mode the severity designator for informative messages is included.

- Interactive Usage Restrictions: When logging in to NOS/VE (i.e. HELLO,TAF etc.) do not enter a terminate break (CTRL t) or a pause break (CTRL p) before the 'welcome message' appears at the terminal. A pause or terminate break entered before the interactive NOS/VE job has completed it's initialization may crash the system.

  A REQUEST_TERMINAL command in a batch job will crash the system. This can happen accidently through a REQUEST_TERMINAL command in a user prolog when the user runs a batch job since the prolog is executed for both interactive and batch jobs. The problem can be avoided by making the REQUEST_TERMINAL

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

command in the prolog conditional on the job type as follows:

```
IF $JOB(MODE) <> 'BATCH' THEN
  REQUEST_TERMINAL
IFEND
```

- The LOGIN and LOGOUT commands have been implemented. This includes the processing of system and user prologs and epilogs. Since prolog processing is now done automatically at login, the SETUP command has been deleted. The default for the FAMILY_NAME parameter is inherited from the submitting job rather than set to an operator provided default as specified in the ERS.

  The system prolog consists of processing the $SYSTEM.SYSPROF file as was done previously by SETUP. If any errors occur during system prolog processing, the user job is aborted in both interactive and batch modes.

  User prolog processing consists of attempting to attach a file called PROLOG in the $USER catalog. If unsuccessful, an attempt is made to get the file PROLOG from the user's 170 catalog; if this is successful, the file is saved in the $USER catalog for subsequent accesses. If the file PROLOG is now available, it is INCLUDEd; otherwise nothing further is done. If any errors occur during user prolog processing, batch jobs are aborted and interactive jobs continue.

  User epilog processing consists of the same steps as user prolog processing except that the file involved is EPILOG and if errors occur, processing continues with the system epilog.

  The system epilog is an empty procedure for the time being.

  LOGIN automatically issues a LINK_USER command for the user logging in. In addition, LINK_USER now works as documented in the ERS, that is issuing a LINK_USER after login changes the job setting rather than aborting.

- The contents of the $SYSTEM catalog can be displayed using the DISPLAY_CATALOG $SYSTEM command.

- Up until now if an error occurred at the "first level" in batch jobs (i.e. the command came from the file COMMAND) the error was reported and the job continued processing commands. This is no longer true; an error at the first level will cause batch jobs to be aborted.

  The file $RESPONSE is no longer connected to the file OUTPUT

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

in batch jobs. The CONNECT_FILE command may be used to make this connection if desired.

The "Welcome to NOS/VE ..." banner is no longer written to $RESPONSE. It is now written to OUTPUT and only in interactive jobs (i.e. not in batch or console jobs).

- The GET_FILE and REPLACE_FILE commands have been implemented and they take the place of the procedures that were available in previous builds. The HCS GET and REPLACE commands have been withdrawn.

- Permanent file catalogs and object libraries can be included in the command list via the SET_COMMAND_LIST command. Both SCL procedures and program descriptions can be directly executed with the necessary files being automatically attached.

- In PROC declarations, FILEREF is no longer a valid SCL parameter type. The FILE specification should be used instead and is the default.

  Also the APPLICATION specification has been changed from:
  APPLICATION <sp> <application value procedure name>
  to:
  <application value name [<sp> <application value procedure name>]

- The DISPLAY_FILE_STATUS command has been renamed to DISPLAY_PRINT_STATUS.

- The CATALOG parameter for the DISPLAY_CATALOG command now defaults to the working catalog rather than being required.

- The new version of DISPLAY_LOG has the following parameters:

```
DISPLAY_LOG DISPLAY_OPTIONS: DISPLAY_OPTION, DO: INTEGER or
     KEY all, last=all
   OUTPUT, O=$output
   STATUS
```

  The parameters are defined as documented in the command interface, however, regardless of the number of lines requested to be displayed, only the last line is displayed. The TYPE parameter has been removed. DISPLAY_LOG can only display the user's job log.

- CLP$PUT_STND_OUT has been deleted. Calls to CLP$PUT_STND_OUT should be replaced by calls to either CLP$PUT_JOB_OUTPUT (deck

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

CLXOUT) or CLP$PUT_JOB_COMMAND_RESPONSE (deck CLXRESP) which will write to the files OUTPUT and $RESPONSE respectively.

- The modifications made to the type pmt$program_description (deck PMDPRGX) require that users of PMP$EXECUTE recompile.

- In addition to the renaming of parameters and adding all parameter abbreviations that occurred in this build, the command names will be renamed as well. In this build the only commands that have been reimplemented in their new form are the subcommands for the CREATE_OBJECT_LIBRARY utility. The new subcommands are documented in the new version of the command interface. To obtain a copy of the new command interface for CREATE_OBJECT_LIBRARY simply type SES,MAD.LISTNCI S=11 on SN101 at Arden Hills.

  The old subcommands for the library generator are supported as well as the new subcommands. The only incompatiblity with the old ERS is the removal of the I parameter from the CREATE_OBJECT_LIBRARY command. The equivalent capability can be realized by calling CREATE_OBJECT_LIBRARY and doing an INCLUDE on the file that would have been specified with the I parameter.

  The new parameter names and abbreviations can be determined by using the DISPLAY_COMMAND_INFORMATION command.

- Any product or utility that is placed in the $SYSTEM catalog (or any frequently loaded program) should be bound using the CREATE_MODULE subcommand of the CREATE_OBJECT_LIBRARY utility. This will minimize overhead associated with loading the product or utility.

- The PMP$GENERATE_UNIQUE_NAME program interface has been made available. This is of particular importance to all utilities and products that use names for intermediate files or local queues; unique names should be used for these names to allow successful asynchronous execution of the utility or product.

- Date and time are now copied across from NOS at every deadstart.

- Debug responds to terminal breaks when a program is being debugged. However, entering a pause or terminate break when debug is active (i.e. the DB/ prompt has appeared and the user has not issued the RUN command) will cause the task to terminate.

- Permanent Files still are permanent only until a crash or

1.0 MAJOR CHARACTERISTICS OF THIS BUILD

deadstart.

- You can cause SCL to output the commands being executed from a procedure or include file by doing:

  connect_file $ECHO OUTPUT

- When sharing executable files via permanent files (i.e. compilers, libraries, etc.) you should make the file an object library via the CREATE_OBJECT_LIBRARY utility. By sharing object libraries instead of object files, the code is actually shared among all tasks using the library; the library is not copied to another segment but is executed directly.

- The file(s) specified by the FILE parameter on the EXECUTE command may not be object library files.

- The program option PRESET now works.

1.1 NOS/VE USAGE EXAMPLES

1.1.1 EXECUTING PROGRAMS

LIMITATIONS

- The file(s) specified by the FILE parameter on the EXECUTE command may not be object library files.

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:

  - Attaching them from the system catalog, either explicitly or via prolog

  or

  - Creating the library file via the object library generator

  or

  - Staging the library file from NOS to NOS/VE using the GET_FILE command with B56 conversion mode specified.

- Get the file from NOS and convert the object text file from

1.0 MAJOR CHARACTERISTICS OF THIS BUILD
1.1.1 EXECUTING PROGRAMS

the CI data mapping to II data mapping by executing the CONVERT_OBJECT_FILE command.

- Load and execute the program via the EXECUTE command, specifying the necessary libraries with the LIBRARY parameter; alternatively SET_OBJECT_LIST may be used to include the libraries in all subsequent EXECUTE 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 command if running on the hardware.

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 CITXTRS PARAMETER='program parameters'
PRINT 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 user library 'mylib' and the CYBIL run-time library. These libraries have been generated on NOS/VE and saved on NOS.

```
GET_FILE MYLIB DC=B56
SET_PROGRAM_OPTIONS LOAD_MAP_OPTIONS=(BLOCK,ENTRY_POINT,SEGMENT)
CONVERT_OBJECT_FILE CITXTRS
EXECUTE CITXTRS 'program parameters' LIBRARY=MYLIB
PRINT LOADMAP
```

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

Notes:

1.0 MAJOR CHARACTERISTICS OF THIS BUILD
1.1.2 CREATE OBJECT LIBRARY ON NOS/VE AND SAVE IT ON NOS

- CLG0170 is NOS permanent file name for file containing CI object text for modules to be included in the library.
- IITEXT180 is NOS/VE local file name for file containing II object text for modules to be included in the library.
- LIBRARY180 is NOS/VE local file name for the library being created.
- 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

- ILIB170 is NOS permanent file name for file containg the old library
- LIBRARY180 is NOS/VE local file name for file containing the old library
- CMOD170 is NOS permanent file name for file containing CI object text for the new module
- NEWIIMODULE is NOS/VE local file name for file containing II object text for the new module
- NEWLIBRARY is NOS/VE local file name for the library being created
- NLIB170 is NOS local file name for new library

NOS/VE Job Commands

```
GET_FILE LIBRARY180 ILIB170 DC=B56
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 H,O display) with the name IRHFxxx 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 K display:

```
PRINT LINKMAP
```

2.0 COMMAND INTERFACE STATUS

2.0 COMMAND_INTERFACE_STATUS

2.1 ACCESS_TO_NOS/VE_IN_DUAL_STATE

2.1.1 LOGIN TO NOS/VE

To initially login to NOS/VE via TAF, you must cause the first login attempt to fail. This can be done by responding to the "FAMILY:" login prompt with something like: "A,A,A". This must be done because the system will try to connect the terminal to IAF on the first login attempt no matter what is typed. To access TAF do the following on the second "FAMILY:" prompt:

,user,password,TAF

You can access TAF from IAF by doing "HELLO,TAF" or by answering TAF 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 TAF.

3) Terminal breaks (control-t and control-p) now work. It is possible to terminate a task or command and to 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 commands are available:

2.0 COMMAND INTERFACE STATUS
2.1.2 TERMINAL USAGE

RESUME - resume execution at the point of interruption.

TERMINATE - 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.

PDI_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

2.0 COMMAND INTERFACE STATUS
2.2 COMMAND AND PARAMETER NAMES

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.

2.2.1 PARAMETER NAMES AND ABBREVIATIONS

The primary convention for a parameter name abbreviation is to use the first character from each word in the parameter name, for example, line_number would abbreviate to ln.

The status parameter is never abbreviated. Since this parameter is available on every command, the abbreviation 's' would not be available for any other parameters.

Some abbreviations can be 'cluster universal' within a command cluster, i.e. within the source code utility or the permanent file commands. These abbreviations should be made the same even if an abbreviation could get by with fewer characters in some commands.

Where a parameter can accept a list of one or more values and a plural form of the parameter name makes sense, both the plural and singular form are allowed.

In some situations a parameter name is used as a parameter name on one command and as a value on another command or function (DISPLAY_FILE_ATTRIBUTES command and $FILE function for example). The abbreviations must be the same in both places.

It is not possible to follow this convention in all situations (i.e. some commands contain parameter names which would result in duplicate abbreviations). In this case judgement and consideration of human factors should be applied to determine abbreviations. When there is already a well known English abbreviation, use it.

The abbreviations for PASSWORD: PW, MINIMUM: MIN and MAXIMUM: MAX do not allow the single letter convention to be followed, so an abbreviation was selected.

2.2.2 PARAMETER NAME CHANGES

COMMAND NAME	OLD PARAMETER NAME	NEW PARAMETER NAME
attach	share	share_mode
change	new_charge	new_account_project
"	new_pfn	new_file_name
change_priority	name	job_name
delete_catalog_permit	family	family_name
delete_permit	"	"
display_catalog	type	display_options
display_command_list	info_level	"
display_file	type	"
display_file_attributes	attributes	"
display_job_status	name	job_name
display_log	from	display_options
display_terminal_attributes	attributes	"
drop_job	name	job_name
execute	map	load_map
"	map_options	load_map_options
"	preset	preset_value
"	procedure	starting_procedure
file	max_block_length	maximum_block_length
"	max_record_length	maximum_record_length
"	min_block_length	mimimum_block_length
"	min_record_length	minimum_record_length
"	user_info	user_information
get_file	conversion	data_conversion
login	family	family_name
"	name	job_name
permit	application_info	application_information
"	family	family_name
"	share	share_mode
permit_catalog	application_info	application_information
"	family	family_name
"	share	share_mode
print	conversion	data_conversion
"	form	forms_code
replace_file	conversion	data_conversion
request_terminal	eoi_string	end_of_information
"	eop_string	end_of_partition
"	no_format_effectors	format_effectors
"	page_wait	hold_page
"	transparent_end_count	transparent_character_count
set_message_mode	info_level	information_level
set_program_options	map	load_map
"	map_options	load_map_options
"	preset	preset_value

2.0 COMMAND INTERFACE STATUS
2.2.2 PARAMETER NAME CHANGES

submit	name	job_name
switch	"	"
terminal	eoi_string	end_of_information
"	eop_string	end_of_partition
"	no_format_effectors	format_effectors
"	page_wait	hold_page
"	transparent_end_count	transparent_character_count

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	new
$CONDITION	new
$SEVERITY	new
$PROCESSOR	new
$JOB	new

2.4 SYSTEM_ACCESS_COMMANDS

Commands_	Status

2.0 COMMAND INTERFACE STATUS
2.4 SYSTEM ACCESS COMMANDS

LINK_USER	Updated to Rev 8 ERS
LOGIN	Updated to Rev 8 ERS - *1
LOGOUT	unchanged
PASSWORD	new

*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
FILE	unchanged - *1
COPY	unchanged
DUMP_FILE	unchanged
COMPARE	unchanged
DISPLAY_FILE_ATTRIBUTES	unchanged
SKIP	unchanged

*1 Multiple FILE commands will not be merged.

2.7 PERMANENT_FILE_MANAGEMENT

Command_	Status
HCS GET	removed
HCS REPLACE	removed
GET_FILE	new
REPLACE_FILE	new
DEFINE	unchanged
ATTACH	WAIT parameter supported
PURGE	unchanged
CHANGE	unchanged
PERMIT	unchanged

2.0 COMMAND INTERFACE STATUS
2.7 PERMANENT FILE MANAGEMENT

DELETE_PERMIT	unchanged
DEFINE_CATALOG	unchanged
PURGE_CATALOG	unchanged
DELETE_CATALOG_PERMIT	unchanged
PERMIT_CATALOG	unchanged
DISPLAY_CATALOG	unchanged
DISPLAY_FILE	unchanged
SET_WORKING_CATALOG	new

Permanent File Deficiencies

1. Permanent files on NOS,/VE are only permanent until a NOS/VE deadstart.

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
DECLARE_VARIABLE	unchanged - *1
REMOVE_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	unchanged
COLLECT_TEXT	unchanged
DISPLAY_VALUE	unchanged
EXIT_PROC	unchanged
ACCEPT	unchanged
DO	unchanged
CONNECT_FILE	unchanged
DISCONNECT_FILE	unchanged
DISPLAY_CONNECTION	unchanged
change HCS variable	unchanged
display HCS variable	unchanged

*1 Variables can no longer be declared with the same names as

the boolean constants.

2.9 INTERACTIVE_COMMANDS

Command_	Status
RESUME	unchanged
TERMINATE	unchanged
TERMINAL	unchanged
DISPLAY_TERMINAL_ATTRIBUTES	unchanged

2.10 OBJECT_CODE_MAINTENANCE

Command_	Status
CREATE_OBJECT_LIBRARY	new - *1
DISPLAY_LIBRARY	unchanged
SELECT_DISPLAY_LEVEL	unchanged
ADD	unchanged
REPLACE	unchanged
COMBINE	unchanged
CREATE_MODULE	unchanged
BIND_MODULE	unchanged
DEFINE_PROGRAM	unchanged
DELETE	unchanged
CHANGE	unchanged
SATISFY	unchanged
REORDER	unchanged
GENERATE	unchanged
QUIT	unchanged
CI to II Conversion	command name change - *2

*1 CREATE_OBJECT_LIBRARY and the abbreviations CREOL and COL are available as commands rather than SCL procedures in this build. In addition, CREATE_OBJECT_LIBRARY supports both its previous names for subcommands as well as the new names documented in the command and parameter name changes DAP.

The I parameter supported by the old COL procedure is not supported by the new commands. The equivalent capability can be realized by calling CREATE_OBJECT_LIBRARY and doing an INCLUDE on the file that would have been quoted on the I parameter.

*2 Users of CITOII and OBJLIST should use the new procedures

2.0 COMMAND INTERFACE STATUS
2.10 OBJECT CODE MAINTENANCE

CONVERT_OBJECT_FILE and DISPLAY_OBJECT_TEXT. These procedures are described elsewhere in this document. The CITOII and OBJLIST procedures will be removed in the next build.

2.11 USER_SERVICES

Command_	Status
DISPLAY_LOG	new parameters
COMMENT	unchanged

2.12 FILE_ROUTING

Command_	Status
HCS JMROUTE	unchanged

2.13 PROGRAM_EXECUTION

Command_	Status
SET_OBJECT_LIST	unchanged
SET_PROGRAM_OPTIONS	unchanged
DISPLAY_PROGRAM	unchanged
EXECUTE	unchanged
"name call"	unchanged - *1
TASK/TASKEND	new
TERMINATE_TASK	new
WAIT	new
SET_DEBUG_RING	unchanged

*1 Warning - "name call" works only for SCL procedures unless a FILE command has been issued to specify that the FILE_CONTENTS are OBJECT and the FILE_ORGANIZATION is DATA or LIBRARY. The FILE command must be reissued every time the file is attached or brought over from NOS.

2.14 JOB_MANAGEMENT

Command_	Status

2.0 COMMAND INTERFACE STATUS
2.14 JOB MANAGEMENT

SUBMIT	unchanged
DISPLAY_JOB_STATUS	unchanged
DROP_JOB	unchanged
PRINT	unchanged
DROP_FILE	unchanged
DISPLAY_PRINT_STATUS	unchanged

2.15 NOS/VE_COMMANDS_IMPLEMENTED_AS_PROCS

In this build, several NOS/VE commands have been implemented as SCL procedures in order to make the system look more like the final version. Users are urged to use these procedures rather than their interim counterparts since the interim commands will ultimately be withdrawn.

These procedures will be made available via the default system prolog.

2.16 NON_STANDARD_COMMANDS

The following commands provide a nonstandard means of performing various frequently performed functions. They will be superceded in subsequent builds by standard commands and capabilities.

2.16.1 DISPLAY_COMMAND_INFORMATION ; 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>:all
                            [utility_name=create_object_library:
                              col:source_code_utility:scu:system]
                            [display_option=parameter_description_
                              pdt:notes:names:help]
                            [output=<file reference>]
                            [status=<status variable>]
```

command_name:cn: This parameter specifies the name of the

2.0 COMMAND INTERFACE STATUS
2.16.1 DISPLAY_COMMAND_INFORMATION ; DISCI

command about which information is to be displayed.

utility_name!un: This parameter specifies which utility the command belongs to. Omission will cause SYSTEM to be used.

display_option!do: This parameter specifies the type of display being requested. The options are:

parameter_description_table!pdt - selects a display of the parameter description 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.

output!o: This parameter specifies the file to which information will be displayed. Omission will cause $OUTPUT to be used.

status: See ERROR HANDLING.

2.16.2 CONVERT_OBJECT_FILE ; CONOF

The purpose of this command is to get a NOS/VE object file or object library 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>]
```

to ; t: This parameter specifies the NOS/VE file name on which the converted object file is to be written.

from ; f: 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.

user ; u: 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 LINK_USER command.

status: See ERROR HANDLING.

2.16.3 GET_OBJECT_FILE ; 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>]
                [password=<string>]
                [status=<status variable>]
```

to ; t: This parameter specifies the NOS/VE file name of the object file.

from ; f: 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.

user ; u: 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 LINK_USER command.

status: See ERROR HANDLING.

2.16.4 GET_OBJECT_LIBRARY ¦ 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>]
```

to ¦ t: This parameter specifies the NOS/VE file name of the object library.

from ¦ f: 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.

user ¦ u: 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 LINK_USER command.

status: See ERROR HANDLING.

2.16.5 DISPLAY_OBJECT_TEXT ¦ 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>]
```

file ¦ f: This parameter specifies the object file or object library containing the object text to be listed.

output ¦ o: This parameter specifies the file to which the display is to be written.

2.0 COMMAND INTERFACE STATUS
2.16.5 DISPLAY_OBJECT_TEXT : DISOT

Omission will cause the file $OUTPUT to be used.

status: See ERROR HANDLING.

2.16.6 CITOII

This command performs the same function as the CONVERT_OBJECT_FILE command and will be removed in the next build. Users are urged to use CONVERT_OBJECT_FILE instead.

The purpose of this command is to get a CI object file or object library from NOS and to convert it to an II object file suitable for processing by the NOS/VE loader and/or object library generator.

```
citoii ci=<NOS file name>
       [ii=<NOS/VE lfn>]
       [user=<NOS user name>]
       [status=<status variable>]
```

ci: This parameter specifies the NOS permanent file name of the CI object file or object library to be converted.

ii: This parameter specifies the NOS/VE file name on which the converted (i.e. II) object file is to be written.

Omission will cause the CI file name to be used.

user: This parameter specifies the NOS user name in whose catalog the CI object file is located.

Omission will cause the user name of the user issuing the command to be used.

status: See ERROR HANDLING.

2.16.7 EDIT_FILE:EDITF

The purpose of EDIT_FILE is to initiate the execution of the SCU editor on text file.(For details see ARH3883.)

```
edit_file:editf - edit lines on a source file.
(procedure file not necessarily in its final form)
```

2.0 COMMAND INTERFACE STATUS
2.16.7 EDIT_FILE:EDITF

parameters	defaults
file=file(source)	$REQUIRED
[result=file(source)]	$VALUE(FILE)
[input=file reference]	$COMMAND_OF_CALLER
[output-file reference]	$OUTPUT
[status]	--

2.16.8 JEDIT

The purpose of this command is to initiate execution of the JEDI editor built by Jack Bohnhoff. Anyone wanting information about the editor should contact Jack.

```
jedit from=<file>
      [status=<status variable>]
```

from ! f: This parameter specifies the file to be editted. This file is rewritten after the editor terminates.

status: See ERROR HANDLING in the NOS/VE Command Interface.

2.16.9 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.

2.16.10 OBJLIST

This command performs the same function as the DISPLAY_OBJECT_TEXT command and will be removed in the next build. Users are urged to use DISPLAY_OBJECT_TEXT instead.

The purpose of this command is to produce a formatted listing of an object file or object library produced on NOS/VE (i.e. II object text).

```
objlist object=<lfn>
        [list=<lfn>]
        [status=<status variable>]
```

object ! o: This parameter specifies the object file or

2.0 COMMAND INTERFACE STATUS
2.16.10 OBJLIST

object library to be listed.

list ; I: This parameter specifies the file on which the formatted listing is to be written.

Omission will cause the listing to be printed on the local printer.

status: See ERROR HANDLING.

2.16.11 LINK_USER ; LIU

The LINK_USER command is the same as documented in the NOS/VE command interface with the exception that the alias LIU is supported in the current system and 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).

2.16.12 JMROUTE

This command is an interim implementation of the final ROUTE command. For printing files, users should use the PRINT procedure now available as the JMROUTE command will be withdrawn in subsequent builds.

JMROUTE,jobname,filename,PR,NVE

jobname - name that the print file will have in the NOS/VE output queue.

filename - name of the local NOS/VE file created by BAM that is to be printed.

PR - specifies that the file is a print file (must always be PR).

NVE - name of the NOS/VE family for the print file (must always be NVE).

Example of JMROUTE command:

JMROUTE,LISTING,LINKMAP,PR,NVE

3.0 PROGRAM INTERFACE STATUS

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$GET_VALUE	unchanged
CLP$DECLARE_VARIABLE	unchanged
CLP$REMOVE_VARIABLE	unchanged
CLP$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$FORMAT_MESSAGE	updated - *1
OSP$SET_STATUS_ABNORMAL	unchanged
OSP$APPEND_STATUS_PARAMETER	unchanged
OSP$APPEND_STATUS_INTEGER	unchanged

3.0 PROGRAM INTERFACE STATUS
3.2 MESSAGE GENERATOR

*1 Formatted messages are now produced and a "current message level" option is supported.

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:

- auto_input
- transparent_mode
- prompt_file
- prompt_string

Files assigned to a terminal device can be accessed via the following BAM requests:

- AMP$OPEN
- AMP$GET_NEXT
- AMP$GET_DIRECT
- AMP$GET_PARTIAL
- AMP$PUT_NEXT
- AMP$PUT_DIRECT
- AMP$PUT_PARTIAL
- AMP$CLOSE

3.4 _PROGRAM_EXECUTION

Procedure_	Status
PMP$EXIT	unchanged
PMP$EXECUTE	changed program description
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

3.0 PROGRAM INTERFACE STATUS
3.4 PROGRAM EXECUTION

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$ESTABLISH_CONDITION_HANDLER	unchanged
PMP$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$ESTABLISH_DEBUG_CFF	unchanged
OSP$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

3.0 PROGRAM INTERFACE STATUS
3.7 PROGRAM SERVICES

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$GET_PROCESSOR_ATTRIBUTES	unchanged
RMP$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$DESCRIBE_NEW_FILE	unchanged
AMP$FILE	unchanged
AMP$GET_FILE_ATTRIBUTES	unchanged - *4
AMP$FETCH	unchanged
AMP$STORE	unchanged
AMP$COPY_FILE	unchanged
AMP$RENAME	unchanged
AMP$RETURN_LOCAL_FILE	unchanged
AMP$OPEN	unchanged

3.0 PROGRAM INTERFACE STATUS
3.9 FILE MANAGEMENT

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.

*4 AMP$GET_FILE_ATTRIBUTES
The EXISTING_FILE parameter is not implemented correctly. We have a PSR indicating that the value of existing_file is always set to FALSE. However, the code appears to always set the value to TRUE. In any case, the value cannot be relied upon.

3.10 PERMANENT FILE MANAGEMENT

Procedure	Status
PFP$DEFINE	unchanged

3.0 PROGRAM INTERFACE STATUS
3.10 PERMANENT FILE MANAGEMENT

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

Permanent file program interface deficiencies

1. Permanent files on the NOS/VE are only permanent until a NOS/VE deadstart.

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

3.0 PROGRAM INTERFACE STATUS
3.13 INTERACTIVE FACILITY

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

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
167,000 - 167999	CM	Configuration Mgmt
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

3.0 PROGRAM INTERFACE STATUS
3.14 NOS/VE EXCEPTIONS

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	DB	Deadstart/Recovery
310,000 - 319,999	MS	Maintenance Services
320,000 - 329,999	Reserved	
340,000 - 349,999	SF	Statistics Fac.
330,000 - 339,999	US	User Errors
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

4.0 DUAL_STATE_DEADSTART_AND_OPERATION

4.1 CURRENT_DUAL_STATE_CONFIGURATION

The Arden Hills S2 development systems are configured to run with three FMD units.

o FMD Unit 43

This unit contains the following:

- A170 NOS (5.3 as released), CTI, CMSE, EI binaries, NOS deadstart files
- Files associated with user name LIBRARY
- Files associated with user name SES
- Files associated with DEV1, DEV2, REL1, INT1.

o FMD Unit 41

This is a scratch unit. This unit can be initialized, if there is a system error during deadstart, by adding the following lines to the CMR deck:

```
INITIALIZE,1,AL.
GO.
```

o FMD Unit 42

This unit contains the following:

- NOS/VE Development Area PL's and Member PL's
- NOS/VE Deadstart Files to be tested (saved in individual user's catalogs)
- Files associated with user name INT2

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.

4.0 DUAL STATE DEADSTART AND OPERATION
4.2 USER NAMES AND PERMANENT FILES

- Your password will be these 3 letters followed by the letter 'x'.
- You must see COMSOURCE (R.K. Cooper - x3092) to be assigned a user index

2) PF dumping and loading

You may use "SES.DUMPPF" on SN/101 to dump your permanent fies 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 IO RELOAD CONTROLWARE FOR THE NOS/VE DISK DRIVER

The following commands need to be entered from deadstart

- Set deadstart panel to deadstart from unit 43
- Push D/S button
- Select "M" display
- Type in "6.20"
- Carriage return
- Type in "CW MA8 2" carriage return wait until LOADED
- Then redeadstart as described in the next section

4.4 A170 NOS DEADSTART

- Set the D/S panel to deadstart from the primary system disk. This is Unit 43 for all Build P systems.

- Push D/S button
- Select "O" display
- Select "H" display
- Enter CM=10000
- Enter (CR)
- Enter date/time

Wait for deadstart to complete.
Note: The deadstart tape DUAL6P (which is currently installed on unit 43) is found in the area in the northeast corner of the room where the tape cabinet is found.

4.0 DUAL STATE DEADSTART AND OPERATION
4.5 NOS/VE DEADSTART AND INSTALLATION

4.5 NOS/VE DEADSTART AND INSTALLATION

- 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.

- Bring up dual state:
  X.UPMYVE (CAT=mycat, DEV1=scat)
  where mycat = user catalog (as before)
  scat = system catalog - INT2, INT1, DEV1 or DEV2

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 catalogue) 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.

4.5.1 DS

The purpose of this command is to perform an installation deadstart or a deadstart of NOS/VE.

```
ds [install=<boolean>]
   [get_source_libraries=<boolean>]
   [get_products=<boolean>]
   [echo=<boolean>]
```

4.0 DUAL STATE DEADSTART AND OPERATION
4.5.1 DS

```
[alternate_user=<NOS_user_name>]
[save_install_files=<boolean>]
[validate_users=<boolean>]
[debug=<boolean>]
[help=<file reference>]
[status=<status variable>]
```

install ; i: This parameter specifies whether an installation deadstart is to be performed. Valid specifications are:
TRUE - installation deadstart to be performed. The system libraries (CYBILIB, SYSLIB, etc.) are built from CI object files.
FALSE - deadstart to be performed. The system libraries are obtained from the results of a previous installation deadstart.

Omission will cause a deadstart to be performed.

get_source_libraries ; gsl: This parameter specifies whether SCU libraries are to be installed. Valid specifications are:
TRUE - libraries are installed
FALSE - libraries are not installed

On the Arden Hills closed shop S2 system, the SCU libraries to be installed are:
OSLPI - operating system program interface
Subset of NOSVEPL - operating system source library

Omission will cause SCU libraries to be installed.

get_products ; gp: This parameter specifies whether the object libraries defining the current product set are to be installed. Valid specifications are:
TRUE - the products are installed
FALSE - the products are not installed

On the Arden Hills closed shop S2 system, the product set to be installed consists of:
CYBIL
SCU
JEDIT
BANNER

Omission will cause the product set to be installed.

echo ; e: This parameter specifies whether the commands

should be echoed to the console during execution. Valid specifications are:
TRUE - echo commands
FALSE - do not echo commands

Omission will cause commands not to be echoed.

alternate_user ; 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.

save_install_files ; sif: This parameter specifies whether to save the system libraries created by an installation deadstart. This parameter is ignored for a normal deadstart. Valid specifications are:
TRUE - save the installed system libraries
FALSE - do not save the installed system libraries

Omission will cause the files not to be saved.

validate_users ; vu: This parameter specifies whether to run the job that validates NOS/VE users. Valid specifications are:
TRUE - run the validation job
FALSE - do not run the validation job

Omission will cause the validation job to be run.

debug ; d: This parameter specifies whether the procedure should abort if an error condition is detected. Valid specifications are:
TRUE - do not abort on an error
FALSE - abort on an error

Omission will cause the procedure to abort when an error is encountered.

help ; h: This parameter specifies whether help information is to be written. If this parameter is specified, the help information will be written to the specified file and the procedure will terminate.

Omission will cause the procedure to execute and the help information not to be written.

status: See ERROR HANDLING in the NOS/VE ERS.

4.0 DUAL STATE DEADSTART AND OPERATION

4.5.1 DS

If you change any of the following decks you MUST use the installation deadstart from your own catalog (with files CYBILGO, XLJCOL, XLJOSL, and XLJLIB), or you must use the alternate_user parameter to specify a NOS catalog containing the files (e.g. DEV1).

AMMTSA	BAMDVR	BAMPC4	BAMPC2	BAMPC1	BAMPC3	IIMRSE	IIMRLE	IIMRUM
IFMEXEC	IIMA72H	IIMTDEL	IIMRUSM	IIMDC2S	OCMREO	OCMMUR	OCMBIM	
OCMSDL	OCMEND	OCMLP	OCMCPY	OCMCRM	OCMGEN	OCMMOMS	OCMDEF	OCMREP
OCMOLG	OCMCOL	OCMOBJ	OCMCHA	OCHOFH	OCMADD	OCMNP	OCMDEL	OCMDLB
OCMCOM	OCMSAT	RHMQAF		RHMQIP	RHMSFM	RHMLOF	RHMMLI	RHMQOP
RHMQTE	RHMLCF		RHMLOG		RHMQRE	RHMA12	RHM12A	RHMROU
RHMWLF	RHMRTN		RHMGDM	USORT	UUSER1	UUTL		

4.5.2 EXAMPLE OF NOS/VE INSTALLATION DEADSTART

Type
K,n. where n is the UPMYVE control point number.
K.LIU (your un,NVE) your_password.
K.GETF DS.
K.DS TRUE FALSE FALSE AL=scat.

The system is up when the following message comes up:

```
---  Deadstart Completed  ---
---  Ready for commands   ---
```

4.5.3 EXAMPLE OF NOS/VE 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 UPMYVE call.

Type (where DEV1 is the same as the CAT=value in the UPMYVE call):
K,n. where n is the UPMYVE control point number.
K.LIU (DEV1,NVE) DEV1X.
K.GETF DS.
K.DS.

The system is up when the following message comes up:

```
---  Deadstart Completed  ---
---  Ready for commands   ---
```

4.5.4 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 validated users. Note that if a job (batch or interactive) is to be run it will have to be run under user JAN.

Type
K,n. where n is the UPMYVE control point number.
K.LIU (your un,NVE) your_password.
K.GETF DS U=scat.
K.DS,,NO NO VU=NO AU=scat.

This deadstart takes a little less than two minutes.
The system is up when the following message comes up:

```
--- Deadstart Completed  ---
--- Ready for commands   ---
```

4.6 NOS/VE INTERACTIVE FACILITY OPERATION

4.6.1 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:
3.NAMS2.

3) If IAF is not up at control point 1, enter:
IAF.

4) Bring up A170 part of interactive:

- TAFNVE.

Control point three must be free or rolloutable (i.e. NAM should not be there). This also brings up PASSON and the MLI subsystem control points.

4.6.2 OPERATOR TERMINATION

To terminate NOS/VE interactive any of the following may be done:

- 3.CFO.DI,AP=TAF. (3 is the NAM control point number)

  This is the preferred method. To bring NOS/VE interactive back up, you must first do a 3.CFO.EN,AP=TAF.

- 3.CFO.DI,NE. (3 is the NAM control point number)

  This terminates the entire network including IAF,RBF, etc.

4.6.3 OTHER OPERATOR CAPABILITIES

- To logically turn the printer on, under DSD enter:
  ON32.
  FORM32,TM.

- To send a "shutdown warning" to all terminals logged on to TAF do:

  3.CFO.ID,AP=TAF. (3 is the NAM control point number)

- To send a message to all terminals do:

  3.CFO.MSG,ALL,mesage. (3 is the NAM control point number)

- 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 control point N do:

  N.ONSWX. (N.OFFSWX.)

  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

4.7 ROUTE AN INPUT FILE FROM C1Z0 TO C180

Through the system console, enter:

Type

X.DIS.
USER,A,B.
GET,filename.
where filename identifies the input file to be routed.
ROUTE,filename,DC=LP,FC=RH.

4.8 K DISPLAY ASCII

Support of 6-12 ASCII from the console (K display) causes the following changes:

INPUT	TRANSLATED TO	INPUT	TRANSLATED TO
/1	^	/(	[
/2	"	/)	]
/3	#	/+	>
/4	$	/-	<
/5	(reversed /)	/=	.
/6	;	/*	' (single quote)
/7	?	//	/
/8	{	/,	:
/9	}	/A to /Z	a - z (lower case)
/0	_ (underscore)		

(The major incompatibilities with earlier systems are for characters for ; and '. To get a semicolon, type /6, to get a ' (single quote), type /*

4.9 DSDI INFORMATION

To create an Express Deadstart Dump (EDD) tape:

1) Mount scratch tape (ring in) on a 9-track drive.

4.0 DUAL STATE DEADSTART AND OPERATION
4.9 DSDI INFORMATION

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 "non zero inhibits rewind" with a CR.

8) Answer "dump number" with a CR.

9) Answer "dump controlware" with a CR.

   * - Warning if this step is omitted, DSDI canot process the dump tape.

To create a listing of the EDD tape:

1) REQUEST,DUMP,NT,D=PE,F=S,LB=KU,PO=R,VSN=your choice.

2) GET,DSDI/UN=DEV1. (On S/N 101.)

   or

   GET,DSDI/UN=DEV1. (On S2.)

3) Create DSDI directives file:

   A DSDI directive file should include the following:

   IOUMR.
   PROMR.
   MEMMR.
   PRORF.
   W,first_byte_address,last_byte_adress,asid. (where the first_byte_address and last_byte_address are hex byte addresses and asid is the asid of the segment to be dumped)

4.0 DUAL STATE DEADSTART AND OPERATION
4.9 DSDI INFORMATION

4) Execute DSDI:

RFL,60000.
DSDI,M,D,I="input directives file".

5) To run (after the first time):

DSDI,I=n.

(Does not read tape again.)

6) To run interactively:

Same as above, except to do W command must first do:

OUTPUT,LISTFIL.

7) 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.

4.10 NOS/VE TERMINATION

o Bringing down dual state:

K.*BYEVE.

o If not a normal termination

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

4.11 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 bring NOS down, do the following:

1) Enter:

CHE

4.0 DUAL STATE DEADSTART AND OPERATION
4.11 A170 NOS SHUTDOWN

The screen will display:
CHECKPOINT SYSTEM.
Enter: carriage return

2) Make sure no mass storage device has a checkpoint rquested. 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 SYSTEM CORE DEBUGGER

5.0 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.

5.1 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.

5.2 SUBCOMMAND_PARAMETER_DEFINITIONS

```
<name> ::= 1-8 character breakpoint name
<condition> ::= READ¦WRITE¦RNI¦BRANCH¦CALL¦DIVFLT¦ARLOS¦
                AROVFL¦EXOVFL¦EXUNFL¦FPLOS¦FPINDEF¦INVBDP
<base> ::= process virtual address
<offset> ::= integer
<length> ::= integer
<frame> ::= 1..100
<count> ::= 1..100
<regid> ::= X¦A¦P
<regno> ::= 0..15¦0..0F(16)
<hex_vstring> ::= 'hex string'
<time> ::= 1..(2**31)-1
<vstring> ::= 'charstring'
<datatype> ::= HEX¦ASCII¦ASC¦DEC
```

5.0 SYSTEM CORE DEBUGGER
5.2 SUBCOMMAND PARAMETER DEFINITIONS

<selector> ::= FULL¦AUTO¦SAVE

5.3 SYSTEM_CORE_DEBUGGER_SUBCOMMANDS

Within the descriptions which follow, optional parameters are enclosed in brackets. Default values for optional parameters are also defined.

5.3.1 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 beactive.

select <selection option> [<ring number> ¦ <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:

<right¦left> - This selects the screen for the debug display. The display stays active when the screen is switched.

<jobmonitor¦nojobmonitor> - This selects whether or not to debug job monitor tasks.

<user¦nouser> - 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.

<job¦nojob> - 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.

5.0 SYSTEM CORE DEBUGGER
5.3.1 SELECT

<alljobs!nojobs> - This activates or deactivates debugging in all jobs.

The initial selections are: RIGHT, NOSTEP, NOJOBMONITOR, NOUSER, HIGHRING=0, NOJOBS.

5.3.2 BREAKPOINT ! B

The purpose of this subcommand is to selects a program interrupt which is to occur 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

5.3.3 REMOVE_BREAKPOINT ! 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.

5.3.4 LIST_BREAKPOINT ! 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.

5.3.5 CHANGE_BREAKPOINT ! 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

5.3.6 TRACE_BACK ! TB

The purpose of this subcommand is to provide information relevant to stack frames associated with an interrupted procedure and its predecessor procedures. This subcommand does not currently validate the PVAs in the stack it is tracing, therefore access violations may occur if the stack being traced has been destroyed.

Information displayed for each selected stack frame consists of:

5.0 SYSTEM CORE DEBUGGER
5.3.6 TRACE_BACK ! TB

- 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>] [FULL!SHORT]

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.

Default parameter values:

<frame>: 1
<count>: 1

5.3.7 DISPLAY_STACK_FRAME ! 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

5.0 SYSTEM CORE DEBUGGER
5.3.7 DISPLAY_STACK_FRAME ! DSF

frame to be displayed.

Default parameter values:

<frame>: 1
<selector>: FULL

5.3.8 DISPLAY_REGISTER ! 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

5.3.9 DISPLAY_MEMORY ! DM

The purpose of this subcommand is to display the contents of a specified area of virtual memory. This subcommand does not currently validate the PVA it is displaying, therefore ahrdware access violations will occur when attempting to display non readable memory.

display_memory <base> [<length>]

Default parameter values:

<length>: 8

5.3.10 CHANGE_MEMORY ! CM

The purpose of this subcommand is to set a specified value into a specified location of virtual memory. This subcommand does not currently validate the PVA it is changing, therefore hardware access violations will occur when attempting to change nonwritable memory.

change_memory <base> <hex_vstring>

5.0 SYSTEM CORE DEBUGGER
5.3.11 RUN

5.3.11 RUN

The purpose of this subcommand is to invoke program execution after a selected program interrupt has occurred.

run

6.0 STAND ALONE DEADSTART

6.0 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 disc deadstart:
Ch=1 U=43

2)Push deadstart button; select M display.

3)Enter: XYZ(cr)
XYZ is a special standalone deadstart routine.

4)A copy of a deadstart panel will appear on the screen. It is setup for CH=13, U=0. If you want to change any word, just type in:
N=xxxx
where N is the word number and xxxx is the octal value. To change the unit number, type in:
5=12u
where u is your deadstart tape physical_unit_#. When done making changes, hit the left_blank key.

5)When you are done making changes or if no changes were made, hit (cr). The tape should move and eventually the message "PROCEED" will appear on the lower left on the console.

6)Enter: CR,M2,BR=0000000000000000
This clears the memory bounds register which is set by NOS deadstart.

7)Enter: DR,P2
This displays the CPU registers. The SIT should be changing; if not, give up.

8)At this point the system core is loaded and can be patched. The commands to display and change memory are documented in IPNDOC.

9)To start the system enter:
SS
The CPU registers should spin; if they ever stop the CPU has halted. It takes a while to initialize the disc and load the

6.0 STAND ALONE DEADSTART

job template, but when or before that is done enter:

```
DD
```

which displays the system dayfile. Commands can now be entered.

7.0 INTERACTIVE PROJECT DUMP ANALYSIS PROCEDURES

7.0 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:

7.1 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 )
```

7.2 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
```

7.0 INTERACTIVE PROJECT DUMP ANALYSIS PROCEDURES
7.3 SEGDUMP

7.3 _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.

7.4 _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
```

7.0 INTERACTIVE PROJECT DUMP ANALYSIS PROCEDURES
7.4 ANALJOB

```
(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)
```

8.0 ARDEN HILLS DEVELOPMENT LAB SUPPORT BY INTEGRATION

8.0 ARDEN HILLS DEVELOPMENT LAB SUPPORT BY INTEGRATION

What we have established in the lab so far is the following:

- A 600 tape capacity tape rack for general use. If you project would like to reserve a section of this tape rack, contact Tim McGibbon or Bonnie Swierzbin.

- A tape and disk cabinet for storage of system support materials which this project will manage and keep up to date. (If you have been using this cabinet for unauthorized storage - beware. We have the key to the lock!) More will be published about the contents of this cabinet later, and a cabinet index will be posted in the lab to help locate where things are supposed to be placed within the cabinet. This cabinet is currently located in the southeast corner of the lab, is 6 ft. 8 in. tall, gray in color and with sliding door.

- A microfiche reader and a metal box containing the most recent microfiche of the system compilation listings and a link map. These items are kept on the table to the right of the console.

- A desk documentation rack for reference manuals and Tom McGee's collection of "how to" goodies. The objective is to have this reference information at arm's length of the console, but it is currently on top of the two-level unit by the West wall.

- At or near the console is a small notebook containing the NOS System Programmer's Instant, NOS Application Programmer's Instant, and the 180 Instruction codes.

Feel free to examine and use all of the above materials while in the lab. Do not remove or abuse any of these materials. Please notify Tim McGibbon or Bonnie Swierzbin of any problems or deficiencies of these materials. Leave a note if we are not available.

APPENDIX A NOS/VE BACKGROUND DOCUMENTS

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 Reference Manual - Vol 1, 60435400 - Vol 2, 60445300

2.4 NOS Instant

2.5 NOS Operators Guide - 60435600

2.6 NOS Diagnostic Handbook

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.

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

Table of Contents