CDC ADVANCED SYSTEMS DEVELOPMENT

15 Dec 81

ERS for CYBER 180 Simulator

ARH 1729

External Reference Specification

for

CYBER 180 Simulator

Version 6.5

Submitted: _____
A. C. Rupert

Approved: _____

_____

DISCLAIMER:
This document is an internal working paper only. It is subject to change and does not necessarily represent any official intent on the part of CDC.

REVISION DEFINITION SHEET

REV	DATE	DESCRIPTION
A	04/07/75	Version 2.0 Initial Release
B	11/24/75	Update for Version 3.0
C	08/06/76	Update for Version 4.0
D	11/15/76	Update for Version 4.1 (NOS conversion)
E	02/07/77	Update for Version 4.2 (MIGDS Rev. G)
F	08/12/77	Update for Version 4.3 (MIGDS Rev. H)
G	02/06/78	Update for Version 4.4 (MIGDS Rev. M)
H	05/18/79	Complete revision for Version 5.0. Addition of IOU simulation and update for MIGDS Rev. N and P.
I	02/11/80	Update for Version 6.0 (MIGDS Rev. Q)
J	07/23/80	Update for Version 6.1
K	10/31/80	Update for Version 6.2
L	03/02/81	Update for Version 6.3 (MIGDS Rev. R)
M	07/21/81	Update for Version 6.4 (MIGDS Rev. S)
N	12/15/81	Update for Version 6.5 Includes changes to conform to the CCL-based SES processor

1.0 INTRODUCTION

The CYBER 180 Simulator is a software development tool that provides a means of executing CYBER 180 Central Processor and Peripheral Processor programs prior to the availability of the CYBER 180 hardware.

The Simulator executes on the NOS operating system and is designed primarily for interactive use, but it may be easily used in batch mode . Descriptions in this ERS are aimed mainly at the interactive user.

1.1 SCOPE

The CYBER 180 Simulator provides a simulated CYBER 180 hardware environment and a repertoire of commands by which the user can exercise control over that environment. The simulation that is provided is a 'pure' hardware simulation, ie. no programs or software functions are provided by the Simulator itself.

It is the users responsibility to provide for the creation of a valid CYBER 180 software environment. This means that the user must supply the object programs, data, etc. which set up the necessary process state registers, processor state registers, etc. for a simulation run. For example, central memory tables and processor state registers necessary for the operation of the CYBER 180 virtual addressing mechanism must be established by the user. Other tools or products exist - such as GENCPF, CYBIL, C180 Assembler - to aid the user in accomplishing this, but a description as to their use is beyond the scope of this document.

A closely related product is the Simulated NOS/VE Program Interfaces package. This package exists as an integral part of the CYBER 180 Simulator, but a description of the capabilities and features of this product is not included in this document and can found in the Simulated NOS/VE Program Interfaces ERS.

The Hardware Checkout System (HCS) physical I/O simulation module also exists as a part of the Simulator, but a definition of its function and use is also beyond the scope of

1.0 INTRODUCTION
1.1 SCOPE

this ERS. This module is only used by the HCS and NOS/VE projects.

The Simulator executes under the control of the SES Command Processor which is described in the ERS for the Command Processor Interface. No attempt is made in this document to describe the features available under the Command Processor or any of its limitations.

2.0 APPLICABLE DOCUMENTS

The following documents reference related material which would be of value and interest to the Simulator user.

- NOS Version 1 Time-Sharing User's Reference Manual (60435500)
- CDC CYBER 180 Mainframe Model-Independent General Design Specification (MIGDS) (ARH1700)
- An Introduction To CYBER 180 (AD&C)
- SES User's Handbook (ARH1833)
- ERS for Simulated NOS/VE Program Interfaces (ARH3125)
- ERS for SES Virtual Environment Linker (ARH2816)
- ERS for SES Virtual Environment Generator (ARH2591)
- ERS for SES Command Processor Interface (SESD001)
- ERS for C180 Assembler (ARH1693)
- ERS for CYBER 180 PP Cross Assembler (S2501)
- CYBIL Language Specification (ARH2298)
- 7154/844 Disk Storage Subsystem MA401 Controlware ERS
- 7155 Disk Storage Subsystem MA721-A Controlware ERS
- 7155-14 Disk Storage Subsystem MA722-A Controlware ERS

3.0 DESCRIPTION

3.1 OVERVIEW

The Simulator provides the user with a capability to execute CYBER 180 CP and PP software on a simulated CYBER 180 hardware system. The Simulator provides interpretive execution of CP and PP instructions and a complete set of commands that give the user the necessary control over the simulated environment. The functions and control available offer the user the following capabilities:

-Central Processor-

1. Load object programs into Central Memory
2. Control the starting and stopping of interpretive execution of CP instructions. Optionally display the results of each instruction's execution.
3. Set and clear CP breakpoint addresses
4. Initiate and terminate instruction tracing
5. Inspect and change the contents of central memory
6. Dump portions of central memory to a specified file
7. Inspect and change the contents of the process state and the processor state registers
8. Remove and add CP simulation from/to the total system simulation
9. Display stack frame information
10. Monitor the performance or an executing program

-Peripheral Processors-

1. Load object programs into the Peripheral Processor's memory
2. Control the starting and stopping of interpretive execution of PP instructions. Optionally display the results of each instruction execution.
3. Set and clear PP breakpoint addresses in each simulated PP
4. Initiate and terminate PP instruction trace

3.0 DESCRIPTION
3.1 OVERVIEW

5. Inspect and change the contents of any PP memory
6. Dump portions of any PP memory to a specified file
7. Inspect and change the contents of any PP register
8. Remove and add individual PP's from/to the total system simulation

-General-

1. Checkpoint and restart the system at any point in the simulation session
2. Display pertinent system information
3. Display the list of Simulator commands and an abbreviated version of the syntax of each command

3.2 SIMULATED CONFIGURATION

The definition of the hardware that is simulated by the CYBER 180 Simulator is contained in the CYBER 180 Mainframe Model-Independent General Design Specification (MIGDS) and in the appropriate disk storage subsystem controlware ERS's.

The current simulated hardware configuration consists of the following components:

- Central Processor (P2)
- Central Memory - 16,777,216 bytes (decimal)
- Input / Output Unit (IOU)
  - 10 Peripheral Processors (PPs numbered 0 .. 9(10))
  - 12 bi-directional I/O channels (numbered 0 .. 13(8))
- 2 7154 disk controllers sharing 4 844-4x disk storage units (dsus)
- 2 7155-1 disk controllers sharing 2 844-4x dsus and 2 885-1x dsus
- 2 7155-14 disk controllers sharing 2 844-4x dsus, 2 885-1x dsus and 2 885-42 dsus

Each PP includes 4096 words of 16 bit memory. Each I/O channel (not connected to a disk controller) is a 16 bit bi-directional channel which may be used by any PP to communicate with any other PP.

The two 7154 disk controllers are on I/O channels 2 and 3. A 12 bit external interface is simulated for these channels. The four shared 844-4x dsus correspond to unit numbers 0,1,2 and 3.

The two 7155-1 disk controllers are on I/O channels 4 and 5, which are also 12 bit channels. The two shared 844-4x dsus correspond to unit numbers 0 and 1. The two shared 885-1x dsus are unit numbers 40 and 41 (octal).

The two 7155-14 disk controllers are on I/O channels 6 and 7 which are full 16 bit channels. The two shared 844-4x dsus correspond to unit numbers 0 and 1. The two shared 885-1x dsus and the two shared 885-42 dsus correspond to unit numbers 40,41,42 and 43 (octal), respectively.

CP and IOU (PPs) instruction simulation is controled via the ON, OFF and RUN, RUN_PP commands. The ON, OFF commands

are used to add or remove a particular processor from instruction simulation. A processor must be on to be included in instruction simulation. Initially, the default states of the processors are: CP on and all PPs off.

The RUN and RUN_PP commands initiate instruction simulation in the CP of PPs, respectively. Instructions are simulated in all PPs that are on in a manner comparable to the hardware "barrel and slot" mechanism.

During instruction simulation, control may be passed back and forth between the CP and the PP processors by means of specially defined "escape" instructions (see section on Special Instruction Processing). A CP escape instruction immediately initiates PP instruction simulation for all PPs in an on state. Likewise, an escape instruction encountered during PP instruction simulation immediately begins (or resumes) CP instruction simulation.

3.3 CALLING THE SIMULATOR

The SES procedure SIM180 is used to invoke the CYBER 180 Simulator.

SIM180 sets up the total system simulated environment and gives control to the Cyber 180 Simulator. At this point, the user may enter any valid Simulator command. The environment that is established is determined by the restart parameter and is either an initial 'empty' system or a previously checkpointed system.

The Cyber 180 Simulator is intended primarily for interactive use (local mode) but may be readily used in batch. An example of batch submittal of a Simulator run is given below. Parameters to SIM180 are:

restart or rs :
(optional) name of a checkpoint file from which to restart simulation. The checkpoint file may be one previously created by GENCPF (GENerate CheckPoint File), VEGEN (Virtual Environment GENerator) or the Simulator Checkpoint command. If you don't code the parameter, then a complete simulated system environment will be created.

cf :
(optional) name of a command file which is to be processed immediately upon activation of the Simulator. The command file consists of one or more valid Simulator command(s). This parameter is required when running in batch.

i180 or i :
(optional) name of the file which contains input for the special Simulator CPU I/O instruction (op code FF(16)). If you're running SIM180 interactively and don't code the i180 parameter, SIM180 takes its input from your terminal. If you run SIM180 in batch and use the CPU I/O instruction for input, you must supply a file containing the input. See section on Special Instruction Processing for more information.

o180 or o :

(optional) name of the file which is to receive output from the special Simulator I/O instruction (op code FF). If you're running SIM180 interactively and don't code the o180 parameter, SIM180 sends output to your terminal. If you run SIM180 in batch you must supply a file to recieve the output, otherwise it is lost. See section on Special Instruction Processing for more information.

Examples of SIM180 Usage

```
ses.sim180                                                    ;
** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)  ;
```

This example shows SIM180 used interactively to create a new simulated system environment. The user may enter any Simulator command once the Simulator banner is displayed.

```
ses.sim180 rs=check1 cf=errata                                ;
** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)  ;
```

This example shows SIM180 used to resume simulation from a checkpoint file. The command file errata will be processed before requesting additional Simulator commands from the user.

```
ses.sim180 checkx i=indata o=outdata                          ;
** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)  ;
```

This example shows SIM180 used interactively to resume simulation from the checkpoint file checkx. The special Simulator CPU I/O instruction, when encountered during CPU instruction simulation, is to read input from file indata and is to write output to file outdata.

```
ses.do batchn file=simcf cs=(..
..? 'ses.sim180 postds cf=simcf o=progout',..
..? 'save,seslog')
 15.54.28. AAAQCBN                                            ;
REVERT.    JOB DO SUBMITTED                                  ;
```

This example shows SIM180 submitted for batch execution via the SES.DO procedure. Simulation resumes from the checkpoint file **postds**, the Simulator commands in the file **simcf** are processed, and any output from the special Simulator CPU I/O instruction is directed to file **progout**.

3.4 RESOURCE REQUIREMENTS / PERFORMANCE CONSIDERATIONS

The Simulator executes on the NOS operating system and requires a minimum of 150,000B words of CM, and optionally uses up to 100,000B words of ECS if available. Usage of ECS is discussed below.

Several performance factors were taken into account in the design of the Simulator. It is to the advantage of the user to have some understanding of these in order to take full advantage of them and thereby keep job overhead and development center overhead to a minimum. The following performance discussion applies exclusively to the CP and PP instruction simulation initiated by the RUN or the RUN_PP commands. Performance considerations of the remaining Simulator commands are negligible in comparison.

As previously stated, instruction simulation may transfer back and forth between the CP and the IOU (PPs) under control of specially defined "escape" instructions. Each transfer represents one primary level overlay load.

The Simulator is designed so that primary level overlay loading is only done when transferring between CP and PP instruction simulation. Secondary level of overlaying is done if a particular feature is being utilized. For example, on the CP side of simulation, the Simulated NOS/VE I/O package is loaded if it is used. The same is true of the Simulated NOS/VE Program Management interfaces and the HCS I/O package. These are only loaded when called. On the IOU side, disk simulation is loaded at a secondary level when it is required.

The overlays are loaded either from an absolute file or from ECS, if it is available and if the user is validated for ECS access.

The following recommendations should be considered by each user when running simulations:

1. All users should be validated for a CM field length which exceeds the minimum requirement of 150,000B. A value of 230,000B is recommended. This is sufficient to obtain maximum performance on large simulation runs eg. NOS/VE OS simulations. Actual CM used will be less for smaller simulation runs.
2. All users should be validated for 200,000B ECS. This should be considered a requirement for any user

running the Simulator. The Simulator uses ECS for overlay loading when it is available, but will load from disk if it is not. Performance of certain Simulator runs and development center system I/O overhead will be adversely effected if ECS is not available.

3. Unused PPs should be in an off state to prevent unnecessary instructions from being simulated.

The on/off status of the individual processors can be inspected via the **display_info** command.

3.5 SIMULATOR FILES

Files created by the Simulator are described here in in order to acquaint the user with their names and usage. A general rule to follow to avoid file naming conflicts is to refrain from using any file names starting with the three letters 'SES'. Any manipulation of the Simulator files during a simulation session such as rewinding or returning will produce unpredictable results.

The following files exist normally, when applicable, as local files following a Simulator session.

SESLOG. This file is a legible file that contains a complete log of the simulation session. Included is all command input and all output resulting from the commands. This file may be printed via the SES. PRINT utility.

SESSMIE. This is a legible file that has all the trace output from traced CP and traced PP instructions. The trace output is in chronological order so CP and PP output may be interspersed. This file may be printed via the SES. PRINT utility.

SESSMKE. This file contains the output from CP keypoint instruction simulation. The file is a collection of binary keypoint records, and is not in a form to be printed. The format of the keypoint file records is explained in the following section on Special Instruction Processing.

SESSMOn. This represents four possible files where n= 0,1,2 or 3. These files contain the data for the four simulated 844-4x dsus shared by the the 7154 controllers, unit numbers 0,1,2 and 3. These files are only created if read from or written to, and only if they don't already exist as a local file.

SESSM1n. This represents four possible files where n= 0,1,2 or 3. These files contain the data for the four dsus shared by the 7155-1 controllers. The files correspond to the two simulated 844-4x dsus 0 and 1, and the two simulated 885-1x dsus 40 and 41, respectively. They are created in the same manner as described above for SESSMOn.

SESSM4n. This represents six possible files where n=

0,1,2,3,4 or 5. These files contain the data for the six dsus shared by the 7155-14 controllers. The files correspond to the two simulated 844-4x dsus 0 and 1, the two 885-1x dsus 40 and 41, and the two 885-42 dsus 42 and 43, respectively. They are created in the same manner as described above for SESSM0n.

3.6 SIMULATOR STATUS INFORMATION

Simulator users can obtain current status information on the simulator by entering SES,STATUS.SIM180. Information pertaining to changes and major problems is listed in reverse chronological order.

3.7 SIMULATOR COMMANDS

The user of the CYBER 180 Simulator controls and manipulates the simulated hardware environment via the individual commands. The commands all generally follow the syntactical rules of the CYBER 180 System Command Language (SCL) as described in the SES Users Handbook.

A general naming convention is followed for Simulator commands - each has a long descriptive name and a short name or 'alias' formed from the first letter of each word of the long name. This is true for all commands except restart and a couple that have no alias. The help command is available to aid the user in command usage.

Simulator commands may be entered at any time after SIM180 has been entered, the Simulator banner has been displayed and the prompt soliciting input is received. For example:

```
SES.SIM180                                                    :
** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)  :
?
```

Any Simulator command may now be entered. Refer to Appendix A for examples of typical Simulator sessions. A Simulator session is terminated by a bye or end command.

CAUTION: Use of the IAF user break 2 sequence to terminate output from or execution of a Simulator command may occasionally cause an immediate exit from the SIM180 procedure. The user break 1 sequence should always be used for these purposes.

NOS commands may also be entered at any time during a simulation session. This is accomplished by simply enclosing the command in single quotes. For example:

Examples:

```
'get,simcf'
'catlist,fn=ckpfile,lo=f'
'ses.rewrite i=filea o=filepm'
```

CAUTION: Use of the IAF user break 2 sequence to terminate output from or execution of a program submitted to NOS in this manner may cause an immediate exit from the SIM180 procedure.

The following sections describe in detail all of the commands available to the CYBER 180 Simulator user. Examples

of usage of each command are also given.

3.7.1 BREAKPOINT ; B

The purpose of the **BREAKPOINT** command is to establish a breakpoint at a specified address in the simulated CPU. Execution of the instruction at the breakpoint address causes the simulation to halt at that point. An appropriate informative diagnostic is issued which specifies the address of the breakpoint.

```
breakpoint address=<pva>
           [frequency=(<i>[..<j>][,<k>])]
           [exchange=<exchange_designator>]
```

address ; a: This parameter specifies the process virtual address (PVA) for the CPU breakpoint. Ring numbers need not be entered as part of the PVA. They are ignored if entered and ignored when testing for a breakpoint address match.

frequency ; f: This parameter specifies the frequency conditions which must be satisfied in order for the breakpoint to be honored. The frequency may be specified in one of three forms:

1. f= i This form will set a CPU breakpoint which will be honored when the specified address is encountered on the ith time and never thereafter.

2. f= i..j This form will breakpoint on the ith time the address is encountered and every time thereafter until the jth time.

3. f= (i..j,k) This form will breakpoint on the ith time the address is encountered and every kth time thereafter until the jth time is reached or exceeded.

Default. If the frequency parameter is not specified on the command, then the CPU breakpoint will be honored each time it is encountered.

exchange ; exc: This parameter specifies an exchange package which is to be used to "qualify" the address parameter. Values accepted for this parameter are:

1. exc= JOB This value indicates that the breakpoint address applies only to any JOB exchange package.

2. exc= MONITOR ; MON This value indicates that the breakpoint address applies only to any MONITOR exchange package.

3. exc= UNQUAL ; UNQ This value is only used to override any **exchange** parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is specified, any previous exchange package parameter qualification is overridden, and the breakpoint address will apply to any exchange package - JOB or MONITOR.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate a particular exchange package to be used to qualify the breakpoint address.

Default. If the exchange parameter is not entered for the command but there is a current exchange parameter qualification due to a previous **qualify_reference** command, then that value is used and interpreted exactly as described above. Otherwise, there is no qualification and the breakpoint address will apply to any exchange package - JOB or MONITOR.

Examples:

```
breakpoint address=1000004911(16) f=1 exchange=job
breakpoint a=3aab(16) frequency=(1..10,2)
b 2300009200(16) exchange=job
b 0b02300008765(16) f=10..15
```

3.7.2 BYE ; END

The BYE command terminates a simulation session and returns control to the NOS operating system.

bye

3.7.3 CHANGE_MEMORY ! CM

The purpose of the CHANGE_MEMORY command is to modify the contents of a field of simulated central memory.

```
change_memory first_byte_address=<pva or rma>
              [byte_count=<count>]
              [repeat_count=<count>]
              memory_value=<value>
              [exchange=<exchange_designator>]
              [address_mode=<mode_designator>]
```

first_byte_address ! fba: This parameter specifies the starting byte address of the the memory field that is to be changed. It is interpreted as a PVA or RMA according to the address_mode parameter. Ring numbers are optional and ignored if entered as part of a PVA.

byte_count ! bc: This parameter specifies the length in bytes of the field in CPU memory that is to be modified. The memory value entered is right justified in this field. The valid range of values for this parameter is 1..8.

Default. If no value is entered then a byte count field length of eight (8) is assumed.

repeat_count ! rc: This parameter specifies the number of times that the memory value is to be stored in successive fields - as defined by the byte_count parameter - in simulated CPU memory. The valid range of values for this parameter is 1..n.

Default. If no value is entered for this parameter, than the assumed value is one (1).

memory_value ! mv: This parameter specifies the value to be stored into the field (or fields) defined by the byte_count and starting at first_byte_address, repeated by repeat_count number of times. The memory value given by this is right justified into the defined field as it is stored into CPU memory.

exchange ! exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to

which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ! MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ! UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state

register - MPS or JPS.

address_mode ; am: This parameter dictates how the first_byte_address parameter is to be interpreted - whether as a process virtual address (PVA) or as a real memory address (RMA). Accepted values for this parameter are:

1. am= PVA This value indicates virtual addressing mode.

2. am= RMA This value indicates real memory addressing mode.

Default. If this parameter is not specified then the mode used is the value established by a previous QUALIFY_REFERENCE command. If no address mode had been established by a QUALIFY_REFERENCE command, then RMA is assumed.

Examples:

```
change_memory first_byte_address=3b00(16) memory_value=0
change_memory fba=200(16) byte_count=4 repeat_count=8 mv=4
cm fba=900002b00(16) mv=888844442222(16) am=pva
cm 801300010010(16) bc=6 rc=100 mv=0fff(16) exc=job am=pva
cm 1ff(16) 1 8 0f(16) am=rma
cm 10000302(16) rc=8 mv=0ffffffffffffffff(16)
```

3.7.4 CHANGE_PP_MEMORY ; CPM

The purpose of the CHANGE_PP_MEMORY command is to modify the contents of simulated PP memory of a specified PP.

```
change_pp_memory [pp_number=<pp_number>]
                 first_word_address=<pp_address>
                 memory_value=<value>
                 [repeat_count=<count>]
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

first_word_address ; fwa: This parameter specifies the PP address of the memory word to be changed, or if the repeat_count exceeds one (1), than it is address of the first PP memory word to be changed.

memory_value ; mv: This parameter gives the value which is to be stored into the specified PP's memory starting at first_word_address for repeat_count number of words. The valid range of values for this parameter is 0..0ffff(16).

repeat_count ; rc: This parameter specifies the number of times that the memory_value is to be stored into successive PP memory locations starting at first_word_address.

Default. If no value is entered for this parameter, then a value of one (1) is assumed.

Examples:

```
change_pp_memory pp=6 first_word_address=7700(8) mv=4
change_pp_memory pp_number=0 fwa=101(8) mv=0 rc=400
cpm 9 7776(8) mv=0ffff(16) rc=2
cpm,,0,0,2000(8)
cpm ,,1315(8) 6224(8)
```

3.7.5 CHANGE_PP_REGISTER ; CPR

The purpose of the CHANGE_PP_REGISTER command is to change the contents of simulated PP registers in a specified PP.

```
change_pp_register [pp_number=<pp_number>]
                   regs=<pp_register_name>
                   vals=<register_value>
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

regs This parameter specifies the name(s) of the PP register which is to be changed. Valid names for this parameter are A, P or R. These may be quoted in any combination or order.

vals This parameter specifies the value which is to placed into the PP register(s) specified by the regs parameter. There must be a one to one correspondence between the values given by vals and the register names specified by the regs parameter.

Examples:

```
change_pp_register pp_number=8 regs=p vals=2205(8)
change_pp_register pp=4 a 777(8)
cpr 4 regs=(a,p,r) vals=(2512(8),7700(8),0)
cpr 0 (a,p) (7777(8),101(8))
cpr ,,p 2011(8)
```

3.7.6 CHANGE_REGISTER ; CR

The purpose of the CHANGE_REGISTER command is to change the contents of a simulated CPU Processor State or Process State (exchange package) register. Parameters are processed and changes made to the registers in the order given below (ie. an S-register change will be made before an A or X register change).

```
change_register [sregs=<state_register_name>]
                [svals=<state_register_value>]
                [aregs=<a_register_number>]
                [avals=<a_register_value>]
                [xregs=<x_register_number>]
                [xvals=<x_register_value>]
                [tos=<top_of_stack_register_number>]
                [tvals=<tos_register_value>]
                [exchange=<exchange_designator>]
```

sregs ; s: This parameter specifies which named Processor State register or Process State register is to be changed. Valid names for this parameter are:

BC ; CFF ; DEC ; DI ; DLP ; DM ; EID ; FRC ; JPS ;
KCN ; KC ; KEF ; KM ; LPI ; LRN ; MCR ; MDF ; MDW ;
MM ; MPS ; OCF ; OI ; PFS ; PID ; PIT ; P ; PND ;
PSM ; PTA ; PTL ; PTM ; SIT ; SS ; STA ; STL ; TE ;
TP ; UCR ; UM ; UP ; UVMID ; VMCL ; VMID

svals ; sv: This parameter specifies the value which is to be placed into the corresponding named state register as given by the SREGS parameter. There must be a one to one correspondence between the the register names specified by the SREGS parameter and the values given for the SVALS parameter. A value quoted here must fit in the register for which it is intended.

aregs ; a: This parameter specifies the number of the A-register to be changed. Multiple A-registers and/or ranges may be requested to be displayed. The limit is five (5) A-registers or ranges per command.

avals ; av: This parameter specifies the value which is to be placed into the corresponding A-register as

given by the AREGS parameter. There must be a one to one correspondence between the the register names specified by the AREGS parameter and the values given for the AVALS parameter. A value quoted here must fit in the register for which it is intended.

xregs ; x: This parameter specifies the number of the X-register to be changed. Multiple X-registers and/or ranges may be requested to be displayed. The limit is five (5) X-registers or ranges per command.

xvals ; xv: This parameter specifies the value which is to be placed into the corresponding X-register as given by the XREGS parameter. There must be a one to one correspondence between the the register names specified by the XREGS parameter and the values given for the XVALS parameter.

tos ; t: This parameter specifies the number of the TOS register to be changed. Multiple TOS registers and/or ranges may be requested to be displayed. The limit is five (5) TOS registers or ranges per command.

tvals ; tv: This parameter specifies the value which is to be placed into the corresponding TOS register as given by the TOS parameter. There must be a one to one correspondence between the the register names specified by the TOS parameter and the values given for the TVALS parameter. A value quoted here must fit in the register for which it is intended.

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the

contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

Examples:

```
change_register sregs=ucr svals=0 exchange=job
change_register s=(ucr,mcr,te) sv=(0,0,2)
cr a=(0,3) av=(800000004010(16),0b0000000004040(16))
cr x=8..b xv=(512,0f3b(16),0fffffffffffffee(16),256)
```

3.7.7 CHECKPOINT ; C

The purpose of the CHECKPOINT command is to save the complete simulation environment on to the specified file. This includes the complete central memory and all configured PP memories, all CPU process and processor state registers, all PP registers, all PP channel information and all internal Simulator control tables and information.

The checkpoint facility allows the user to checkpoint to a file at any point during a simulation run and to use the file as a restart file on a subsequent simulation run. This file may be specified for the restart command or the restart parameter on the SIM180 procedure. The environment and conditions will be reestablished exactly as they were at the checkpoint time.

The checkpoint file may be used as input to the DeadStart Dump Interpreter (DSDI).

checkpoint file_name=<file_name>

file_name ; fn: This parameter specifies the name of the file on which the checkpoint file is to be written.

Examples:

```
checkpoint file_name=cpfile
checkpoint fn=ckp1
c cpfile
```

3.7.8 CLOSE_LOGICAL_FILE ; CLF

The purpose of the CLOSE_LOGICAL_FILE command is to logically close a Simulated NOS/VE I/O file which had been left in an open state from a logical I/O simulation. Refer to the Simulated NOS/VE Program Interfaces ERS for a complete description of this function and other logical I/O simulation features.

close_logical_file [file_name=<p1>]

file_name ; fn: This parameter specifies the name of the file which is to be logically closed. Up to five (5) file names may be quoted for this parameter.

Default. If this parameter is not quoted on the command, then all simulated NOS/VE I/O files will be logically closed.

Examples:

```
close_logical_file file_name=datain
close_logical_file fn=(datain,dataout)
clf twitty
clf (fil1,fil2,fil3,fil4,fil5)
clf
```

3.7.9 DISPLAY_EXCHANGE_PACKAGE ; DEP

The purpose of the DISPLAY_EXCHANGE_PACKAGE command is to display the contents of a selected exchange package. The contents of all registers comprising the exchange package are displayed except for the A, X, and Top Of Stack (TOS) registers.

display_exchange_package [exchange=<exchange_designator>]

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

Examples:

```
display_exchange_package exchange=monitor
dep job
dep
```

Sample output:

```
display_exchange_package exc=job
    P=0000 300E 0000 823C   MCR=0000                    UCR=0000
  MDW=0000 0000 0000 0000    MM=FFFC                     UM=FE00
  UTP=0017 0000 0000       VMID=   0                  UVMID=   0
   TP=2006 0000 0000         TE=   2                   LPID=00
  DLP=0000 0000 0000         DM=00                       DI=00
  STA=0001 2FA0             STL=0023                    LRN=   F
   KC=0000 0000              KM=FFFF                    KCN=   0
  PIT=7FFF FC57             CFF=   0                    OCF=   0
  KEF=   0                  PND=   0                     BC=0000
  MDF=0000
```

3.7.10 DISPLAY_INFO ; DI

The purpose of the DISPLAY_INFO command is to display certain pertinent information about the current state of the simulation. The information displayed includes:

1. Qualifiers established by qualify_reference command:
   - o address mode
   - o exchange
   - o PP number
   - o file name
   - o dbug file
2. Incremental and total run counts for both CPU and IOU from previous run or run_pp commands
3. Incremental and total CP instruction times (if CP timing is on)
4. CPU and PP processor on/off states
5. Breakpoint addresses and number of breakpoints remaining (maximum 9999) for both CPU and PPs (if active)
6. Trace limits for both CPU and PPs (if active)
7. CP SPY address limits (if active)

display_info

Sample output (some lines truncated due to ERS line length):

```
display_info
QUALIFIERS:  ADDRESS_MODE= PVA  EXCHANGE= JOB  PP= 03D
             FILE_NAME= UNQUAL  DBUG_FILE= UNQUAL
LAST_CP_RUN_COUNT=          142  TOTAL=       126142
LAST_PP_RUN_COUNT=          100  TOTAL=       126000
CP_TIMING  LAST RUN   0.000 073 886     TOTAL  0.063 452 331

CONFIGURATION:      CPU       PP00 PP01 PP02 PP03 PP04 PP05 PP06
STATUS:             ON         OFF  ON   OFF  ON   OFF  OFF  OFF
BRKPT:REF XP        JOB
      ADDRSS  00E 0000 8800       7711       4220
      REMAIN          8             2       9999
TRACE:REF XP       0001 2C80
      LIMITS  00E 0000 1000       0700
              00E 0000 FFFF       5500
SPY : REF XP        MON
      LIMITS  00A 0000 1600
              00A 0000 1700
```

3.7.11 DISPLAY_MEMORY ! DM

The purpose of the DISPLAY_MEMORY command is to display the contents of a field of simulated central memory. The field of memory displayed is specified as a starting byte address plus optional offset and a byte count of 1 to 8 bytes. Successive fields may be displayed by specifying the number desired as a value for the repeat_count parameter.

Memory "dumps" may be taken with this command by specifying a filename to receive the output.

```
display_memory first_byte_address=<rma or pva>
               [offset=<count>]
               [byte_count=<count>]
               [repeat_count=<count>]
               [file_name=<local file name>]
               [exchange=<exchange_designator>]
               [address_mode=<mode_designator>]
```

first_byte_address ! fba: This parameter specifies the starting byte address of the the memory field that is to be displayed. It is interpreted as a PVA or RMA according to the address_mode parameter. Ring numbers are optional and ignored if entered as part of a PVA.

offset ! o: This parameter specifies a byte offset which is to be added to the first_byte_address to form the starting byte address of the memory field to be displayed.

Default. If no value is specified for this parameter, than a value of zero (0) is assumed.

byte_count ! bc: This parameter specifies the length of the field of central memory that is to be displayed. The valid range for this parameter is 1..8 bytes.

Default. If no value is entered for this parameter, then 8 bytes (1 word) is assumed.

repeat_count ! rc: This parameter specifies the number of fields to be displayed. The valid range for this

parameter is 1..n.

Default. Omission of this parameter will cause a default of one (1) field to be assumed.

file_name ; fn: This parameter specifies the name of a local file which is to receive the memory display output. A wide line printer format is used when a file name is quoted, and each command execution writes a separate record on the file. Values accepted for this parameter are:

1. fn= <local file name> Any valid NOS file name is accepted. This file name temporarily overrides any qualified file_name established previously by a QUALIFY_REFERENCE command.

2. fn= UNQUAL ; UNQ This value is used to negate any file_name quailification in effect due to a previous QUALIFY_REFERENCE command. The memory contents display is then output to the terminal.

Default. If this parameter is not specified and there is no file_name qualification in effect, then the memory contents display is directed to the terminal. If there is a file_name qualification in effect, then the output will be to the qualified file.

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS)

register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

address_mode ; am: This parameter dictates how the first_byte_address parameter is to be interpreted - whether as a process virtual address (PVA) or as a real memory address (RMA). Accepted values for this parameter are:

1. am= PVA This value indicates virtual addressing mode.

2. am= RMA This value indicates real memory addressing mode.

Default. If this parameter is not specified then the mode used is the value established by a previous QUALIFY_REFERENCE command. If no address mode had been established by a QUALIFY_REFERENCE command, then RMA is assumed.

Examples:

```
display_memory first_byte_address=4009(16) am=rma
display_memory fba=2000(16) byte_count=4 repeat_count=20
dm fba=800000309a(16) bc=2 rc=8 exc=job am=pva
dm 1100009000(16) rc=500 exchange=monitor
dm 0b00000000a00(16) o=16,,40 exc=3000(16)
dm 1200000000(16) rc=0fff(16) exc=job am=pva fn=dump1
```

Sample output (some lines truncated due to ERS line length):

```
display_memory fba=100a00001692(16) rc=12 am=pva exc=job
SEGMENT=  100A
00001692  20212223 24252627   28292A2B 2C2D2E2F   !"# $%&' ()*+
000016A2  30313233 34353637   38393A3B 3C3D3E3F  0123 4567 89:;
000016B2  40414243 44454647   48494A4B 4C4D4E4F  @ABC DEFG HIJK
000016C2  50515253 54555657   58595A5B 5C5D5E5F  PQRS TUVW XYZ[
000016D2  60616263 64656667   68696A6B 6C6D6E6F  `abc defg hijk
000016E2  70717273 74757677   78797A7B 7C7D7E7F  pqrs tuvw xyz{
```

3.7.12 DISPLAY_MEMORY_INDIRECT ; DMI

The purpose of the DISPLAY_MEMORY_INDIRECT command is to display the contents of a field of simulated central memory by specifying the starting byte address indirectly. The starting byte address is obtained from the named state register, A-register, X-register or TOS register specified in the command, added to the offset parameter value, if quoted.

Memory "dumps" may be taken with this command by specifying a filename to receive the output.

```
display_memory_indirect [sreg=<state_register_name>]
                        [areg=<a_register_number>]
                        [xreg=<x_register_number>]
                        [tos=<tos_register_number>]
                        [offset=<count>]
                        [byte_count=<count>]
                        [repeat_count=<count>]
                        [file_name=<local file name>]
                        [exchange=<exchange_designator>]
                        [address_mode=<mode_designator>]
```

sreg ; s: This parameter specifies which named state register contains the first byte address of the field of central memory to be displayed. Only one (1) register may be quoted. Valid names are:

BC ; CFF ; DEC ; DI ; DLP ; DM ; EID ; FRC ; JPS ;
KCN ; KC ; KEF ; KM ; LPI ; LRN ; MCR ; MDF ; MDW ;
MM ; MPS ; OCF ; OI ; PFS ; PID ; PIT ; P ; PND ;
PSM ; PTA ; PTL ; PTM ; SIT ; SS ; STA ; STL ; TE ;
TP ; UCR ; UM ; UP ; UVMID ; VMCL ; VMID

areg ; a: This parameter specifies which A-register contains the first byte address of the field of central memory to be displayed. Only one (1) register may be quoted.

xreg ; x: This parameter specifies which X-register contains the first byte address of the field of central memory to be displayed. Only one (1) register may be quoted.

tos ; t: This parameter specifies which TOS register contains the first byte address of the field of

central memory to be displayed. Only one (1) register may be quoted.

offset ; o: This parameter specifies a byte offset which is to be added to the contents of the register given by the SREG, AREG, XREG or TOS parameter to form the starting byte address of the memory field to be displayed.

Default. If no value is specified for this parameter, than a value of zero (0) is assumed.

byte_count ; bc: This parameter specifies the length in bytes of the field in central memory to be displayed. The valid range of values is 1..8.

Default. If no value is given for this parameter, then a field length of eight (8) bytes will be displayed.

repeat_count ; rc: This parameter specifies the number of successive memory fields to be displayed starting at the address formed by the contents of the quoted register plus the offset value. The valid range for this parameter is 1..n.

Default. If no value is quoted for this parameter, then a repeat count of one (1) is assumed.

file_name ; fn: This parameter specifies the name of a local file which is to receive the memory display output. A wide line printer format is used when a file name is quoted, and each command execution writes a separate record on the file. Values accepted for this parameter are:

1. fn= <local file name> Any valid NOS file name is accepted. This file name temporarily overrides any qualified file_name established previously by a QUALIFY_REFERENCE command.

2. fn= UNQUAL ; UNQ This value is used to negate any file_name quallification in effect due to a previous QUALIFY_REFERENCE command. The memory contents display is then output to the terminal.

Default. If this parameter is not specified and there is no file_name qualification in effect, then the memory contents display is directed to the terminal. If there is a file_name qualification in effect, then the output will be to the qualified file.

exchange ! exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ! MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ! UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had

qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

address_mode ; am: This parameter dictates how the first byte address value in the register is to be interpreted - whether as a process virtual address (PVA) or as a real memory address (RMA). Accepted values for this parameter are:

1. am= PVA This value indicates virtual addressing mode.

2. am= RMA This value indicates real memory addressing mode.

Default. If this parameter is not specified then the mode used is the value established by a previous QUALIFY_REFERENCE command. If no address mode had been established by a QUALIFY_REFERENCE command, then RMA is assumed.

Examples:

```
display_memory_indirect sreg=jps repeat_count=52
display_memory_indirect a=4 bc=6 address_mode=pva
dmi x=0b(16) o=40(16) bc=4 rc=20
dmi t=14
dmi s=p rc=1000(16) am=job fn=dumpm
```

Sample output (some lines truncated due to ERS line length):

```
dmi s=p bc=6 rc=8 exc=job am=pva
SEGMENT=  100A
00001692  20212223 2425       26272829 2A2B       !"# $%   &'()
0000169E  2C2D2E2F 3031       32333435 3637      ,-./ 01   2345
000016AA  38393A3B 3C3D       3E3F4041 4243      89:; <=   >?@A
000016B6  44454647 4849       4A4B4C4D 4E4F      DEFG HI   JKLM
```

3.7.13 DISPLAY_PP_MEMORY ; DPM

The purpose of the DISPLAY_PP_MEMORY command is to display the contents of a word or a consecutive number memory words of a specified simulated PP.

Memory "dumps" may be taken with this command by specifying a filename to receive the output.

```
display_pp_memory [pp_number=<pp_number>]
                  first_word_address=<word_address>
                  [repeat_count=<count>]
                  [file_name=<local file name>]
                  [format=<format designator>]
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

first_word_address ; fwa: This parameter specifies the starting word address of simulated PP memory that is to be displayed.

repeat_count ; rc: This parameter specifies the number of consecutive PP memory words that are to be displayed.

Default. If no value is given for this parameter than a value of one (1) is assumed.

file_name ; fn: This parameter specifies the name of a local file which is to receive the memory display output. A wide line printer format is used when a file name is quoted, and each command execution writes a separate record on the file. Values accepted for this parameter are:

1. fn= <local file name> Any valid NOS file name is accepted. This file name temporarily overrides any qualified file_name established previously

by a QUALIFY_REFERENCE command.

2. fn= UNQUAL ; UNQ This value is used to negate any file_name quallification in effect due to a previous QUALIFY_REFERENCE command. The memory contents display is then output to the terminal.

Default. If this parameter is not specified and there is no file_name qualification in effect, then the memory contents display is directed to the terminal. If there is a file_name qualification in effect, then the output will be to the qualified file.

format ; f: This parameter selects the format of the pp memory display. Values accepted for this parameter are:

1. f= OCT ; O Addresses and memory contents are dumped in octal format. Interpretation of memory contents is presented in Display Code.

2. f= HEX ; H Addresses and memory contents are displayed in hex format with interpretation of memory contents given in ASCII.

Default. Octal format is the default.

Examples:

```
display_pp_memory pp_number=5 first_word_address=100(8) 20
display_pp_memory pp=5 fwa=2000(8) repeat_count=40
dpm 5 2000(8) 40
dpm 0 10(8)
dpm,,70(8) 8
dpm 3 0 rc=4096 fn=dumppp3 f=hex
```

Sample outputs:

```
display_pp_memory pp=3 fwa=5670(8) rc=10
PP03
5670  151530 001531 001532 001533  MX MY MZ M0
5674  001534 001535 001536 001537  M1 M2 M3 M4
5700  001540 001541  M5 M6

dpm 3 fwa=200(16) rc=23 f=hex
PP03
```

3.0 DESCRIPTION
3.7.13 DISPLAY_PP_MEMORY : DPM

```
200  3030 3131 3232 0033 0034 0035 3636 0037   00 11 22  3  4  5
208  0038 3939 003A 003B 003C 003D 003E 003F    8 99  :  ;  <  =
210  0040 0041 0042 0043 0044 0045 0046    @  A  B  C  D  E  F
```

3.7.14 DISPLAY_PP_REGISTER ; DPR

The purpose of the DISPLAY_PP_REGISTER command is to display the contents of the simulated A, P and R registers of a specified PP.

```
display_pp_register [pp_number=<pp_number>]
                    [regs=<register_name>]
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

regs This parameter specifies the name or names of the PP register that is to be displayed. Valid values for this parameter are A, P or R.

Default. If no register name is entered for this command, then the A and P registers will be displayed.

Examples:

```
display_pp_register pp_number=4 regs=(a,p,r)
display_pp_register pp=4 (a,r)
dpr 4 p
dpr 9
dpr ,,r
dpr
```

Sample output:

```
display_pp_register pp=9 regs=(p,a,r)
09D  P= 4211  A= 053126  R= 00100400
```

3.7.15 DISPLAY_REGISTER ; DR

The purpose of the DISPLAY_REGISTER command is to display the contents of specified CPU register(s). Registers which may be displayed include the Processor State Registers and the Process State Registers (Exchange Package registers).

```
display_register [sregs=<state_register_name>]
                 [aregs=<a_register_number>]
                 [xregs=<x_register_number>]
                 [tos=<top_of_stack_register_number>]
                 [exchange=<exchange_designator>]
```

sregs ; s: This parameter specifies a named state register (processor and/or process state register) which is to be displayed. Multiple State registers may be specified up to a maximum of five (5). Valid names which may be given are:

BC ; CFF ; DEC ; DI ; DLP ; DM ; EID ; FRC ; JPS ; KCN ; KC ; KEF ; KM ; LPI ; LRN ; MCR ; MDF ; MDW ; MM ; MPS ; OCF ; OI ; PFS ; PID ; PIT ; P ; PND ; PSM ; PTA ; PTL ; PTM ; SIT ; SS ; STA ; STL ; TE ; TP ; UCR ; UM ; UP ; UVMID ; VMCL ; VMID

aregs ; a: This parameter specifies which A-register or registers are to be displayed. Multiple A-registers and/or ranges may be requested to be displayed. The limit is five (5) A-registers or ranges per command.

xregs ; x: This parameter specifies which X-register or registers are to be displayed. Multiple X-registers and/or ranges may be requested to be displayed. The limit is five (5) X-registers or ranges per command.

tos ; t: This parameter specifies which TOS register or registers are to be displayed. Multiple TOS registers and/or ranges may be requested to be displayed. The limit is five (5) TOS registers or ranges per command.

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which

exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

3.0 DESCRIPTION
3.7.15 DISPLAY_REGISTER ; DR

Examples:

```
display_register sregs=p
display_register s=(jps,mps,ucr,mcr,p) exchange=job
dr aregs=9 xregs=(2,5,7,b,f) tos=(1..f)
dr s=ss a=(0,4..8,d..f) 0..f exc=monitor
```

Sample output:

```
display_register s=(p,ss) a=(0..2) exc=job
     P=0000 300E 0000 823C              SS=0000 0000 0000 0000
     A0 = 3017 0000 0220                A1 = 3017 0000 0220
     A2 = 3017 0000 01C8
```

3.7.16 DISPLAY_STACK_FRAME ; DSF

The DISPLAY_STACK_FRAME command provides the display of selected information from a specified stack frame or a number of consecutive stack frames.

```
display_stack_frame [stack_frame=<frame number>]
                    [repeat_count=<count>]
                    [address=<pva>]
                    [selector=<selector designator>]
                    [auto_length=<count>]
                    [exchange=<exchange_designator>]                ;
                    [dbug_file=<local_file_name>]                   ;
```

stack_frame ; sf: This 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 (the exchange package) or with the stack frame associated with the address parameter, if specified. Stack frame two is associated with the first stack frame procedure's predecessor, etc. The valid range for this parameter is 1 .. 999.

Default. If the stack frame parameter is not specified, then the first stack frame for which information is displayed will be the exchange package or the stack frame associated with the address parameter (if specified).

repeat_count ; rc: This parameter specifies the total number of stack frames for which information is to be displayed. The valid range for this parameter is 1 .. 999. If this value exceedes the number of stack frames, output will terminate with the last frame in the stack.

Default. Information for one stack frame is displayed.

address ; a: This parameter specifies the process virtual address (PVA) of a stack frame save area at which the trace back begins (stack frame number one). Ring numbers are optional and ignored if entered as part of a PVA.

Default. If the address parameter is not specified, stack frame number one is the interrupted procedure (the exchange package).

selector ; s: This parameter identifies a region of the specified stack frame(s) which is to be displayed. Accepted values for this parameter are:

1. s= AUTO This value causes the automatic region of the stack frame to be displayed.

2. s= SAVE This value causes the save area of the stack frame to be displayed.

3. s= FULL This value causes both the automatic and save areas of the stack frame to be displayed.

Default. The full mode is used when this parameter is not specified.

auto_length ; al: This parameter specifies the number of characters of the automatic region of the stack frame(s) to be displayed. Full words are displayed. The valid range for this parameter is 1 .. 1000000.

Default. If this parameter is not specified and the selector parameter is auto, full, or default, a maximum of 960 bytes of the automatic region are displayed.

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

dbug_file ; df: This parameter specifies the name of the local file which contains the debug module address table produced by VELINK. Values accepted for this parameter are:

1. df= <local file name> Any valid NOS file name is accepted. This file name temporarily overrides any qualified dbug_file established previously by a qualify_reference command.

2. df= UNQUAL ; UNQ This value is used to negate any dbug_file qualification in effect due to a previous QUALIFY_REFERENCE command. No module names and offsets appear in the stack frame

displays.

Default. If the dbug_file parameter is not entered for the command, the dbug_file name is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the dbug_file for subsequent commands, then that value is used as the dbug_file name for this command.

(2) If the dbug_file name was not previously qualified by QUALIFY_REFERENCE, the dbug_file remains unqualified.

Examples:

```
display_stack_frame rc=3 selector=save
display_stack_frame a=300000098(16) sf=2 s=full
dsf sf=4 repeat_count=6 exc=job al=1640
dsf selector=auto auto_length=10000
```

3.7.17 HELP ; H

The purpose of the HELP command is to provide a handy, concise presentation of the syntax of the Simulator commands. The intent of this command is to offer the interactive Simulator user a condensed syntax to serve as a memory refresher, and is not intended to be a tutorial on the command set of the Simulator nor is any attempt made to present the command syntax in the standard SCL notation.

The HELP command provides a means to (1) display all of the Simulator commands and their aliases, (2) list the abbreviated syntax of a particular command.

help [command=<command_name>]

command ; cmd: This parameter is used to specify a command whose syntax is to be displayed. The full command name or its alias may be given for this parameter.

Default. If this parameter is not specified on the HELP command, then a complete listing of the Simulator command names and their aliases will be displayed.

Examples:

```
help
help command=display_registers
help cmd=display_registers
h pp_trace
h t
```

Sample output:

```
help load_memory
LOAD_MEMORY  FILE_NAME= P1  FIRST_BYTE_ADDRESS= P2
LM  FN= P1  FBA= P2
```

3.7.18 INCLUDE ; INCL

The purpose of the INCLUDE command is to designate a command file to serve as the current input source. The command file consists of one or more valid Simulator command(s). Command text is taken from the new source immediately. If more command text follows the INCLUDE command, it is processed after the new command file contents are processed.

NOTE: INCLUDEing a command file does NOT cause it to be executed. Execution occurs upon entering a subsequent carriage return, either by itself or following the next command entered. Commands are always executed in the correct order.

include cf=<file_name>

cf: This parameter specifies the name of the local file which contains the Simulator commands to be included. Any valid NOS file name is accepted for this parameter.

Examples:

```
include filex
include cf=filey
incl cmdfile
```

3.7.19 LOAD_MEMORY ; LM

The purpose of the LOAD_MEMORY command is to load the contents of the specified local file into simulated central memory. The load file must be the format of the segment file which is output by the SES Virtual Environment Linker.

This command may be used repeatedly to load multiple segments from different load files if necessary.

CAUTION: Due to the differences in the CYBER 170 and CYBER 180 word lengths, there may be up to three (3) bytes of extraneous data loaded into simulated memory at the end of a load file.

```
load_memory file_name=<local_file_name>
            [first_byte_address=<rma>]
```

file_name ; fn: This parameter specifies the name of the local file which contains the CYBER 180 memory segment or segments. The format of this file is described in the ERS for SES Virtual Environment Linker. This file is rewound before loading.

first_byte_address ; fba: This parameter specifies the starting byte address (RMA) for the load.

Default. If this parameter is not specified then the first byte address of the load is obtained from from the load file. If the file does not contain the first byte address, then the load will start at real memory address zero (0).

Examples:

```
load_memory file_name=segm101 first_byte_address=2000(16)
load_memory fn=sega fba=40FF(16)
lm segfil 0
lm segfil
```

3.7.20 LOAD_PP_MEMORY ; LPM

The purpose of the LOAD_PP_MEMORY command is to load the contents of the specified local file into the simulated memory for the specified PP. The file may be either a binary file output from CYBER 170 Compass PP assembly or a binary file from a CYBER 180 Compass assembly. The file is rewound before loading.

```
load_pp_memory file_name=<local_file_name>
               [pp_number=<pp_number>]
               [first_word_address=<word_address>]
```

file_name ; fn: This parameter specifies the name of the local file which contains the binary memory file output from either a CYBER 170 or a CYBER 180 Compass PP assembly.

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

first_word_address ; fwa: This parameter specifies the starting word address for the PP memory load.

Default. If no value is quoted for this parameter, than the starting word address of the load will be obtained from the tables in the load file. If there is no starting address in the load file table, then ultimately the load will begin at PP word address zero (0).

Examples:

```
load_pp_memory file_name=ppload pp_number=0 fwa=0
load_pp_memory fn=pploadx pp=4 fwa=2000(8)
lpm filea 4 200(8)
lpm mtrbin
lpm mtrbin,,3000(8)
```

3.7.21 OFF

The OFF command is used to turn off a specified PP and/or the CPU. A PP or CPU in an 'off' state is removed from instruction simulation. The special "escape" instructions used to transfer control between CPU and PP instruction simulation are treated as NO-OP instructions if the processor or processors being transferred to off, i.e. CPU or all PPs are off.

It is also used to deactivate CPU instruction timing and "SPY" CPU performance monitoring previously activated by an ON command.

```
off [pp_number=<pp_number>]
    [cpu]
    [cp_timing]
    [spy]
```

pp_number ; pp: This parameter specifies the PP which is to be put in an 'off' state.

Default. If this parameter is not specified, then no PP will be turned off. This command does not default to a PP number qualified by the QUALIFY_REFERENCE command.

cpu ; cp: If this keyword is entered, then the CPU will be put in an 'off' state and removed from instruction simulation.

cp_timing ; ct: This keyword specifies that CPU instruction timing is to be turned off. Thereafter, an average CPU instruction time of 1 microsecond/instruction is used for updating hardware clocks.

spy This keyword deactivates the SPY P-register monitoring established by the ON command and causes the results to be written to the TRACE file SESSMTF. See Appendix B for an example of the SPY output.

Examples:

```
off pp_number=9
off cpu
```

3.0 DESCRIPTION
3.7.21 OFF

```
off 0
off 4 cp
off ct spy
```

3.7.22 ON

One function of the ON command is to turn on a specified PP and/or the CPU. A PP or CPU in an 'off' state is removed from instruction simulation. This command puts the specified PP or CPU back into an 'on' state where it will included in instruction simulation.

This command is also used to activate CPU instruction timing and to activate the "SPY" CPU performance monitoring feature. The current status of both CPU timing and SPY limits and qualification can be displayed via the DISPLAY_INFO command.

```
on [pp_number=<pp_number>]
   [cpu]
   [cp_timing]
   [spy]
   [limits=<start_pva..end_pva>]
   [exchange=<exchange_designator>]
```

pp_number ; pp: This parameter specifies the PP which is to be put in an 'on' state.

Default. If this parameter is not specified, then no PP will be turned on. This command does <u>not</u> default to a PP number qualified by the QUALIFY_REFERENCE command.

cpu ; cp: If this keyword is specified, the CPU will be put in an 'on' state.

cp_timing ; ct: This keyword specifies that CPU instruction timing is to be turned on. Once on, the actual P2 instruction times for the CPU instructions are used for updating hardware clocks. The accumulated CPU instruction times are available via the display_info command.

spy: This keyword activates CPU P-register monitoring for the range of addresses given by the limits parameter below. The P-register monitoring continues until a subsequent deactivation via the off command which causes the results (counts, percentage histogram) to be written to the trace file SESSMTF (See Appendix B for an example of the spy output). A second on spy

replaces the first without producing any output.

limits ; l: This parameter is used in conjunction with the SPY keyword to specify the range of PVA's in which the CPU P-register is to be monitored. The PVA's must be in the same segment. Ring numbers need not be entered as part of the PVA's. They are ignored if entered and ignored when testing against the P-register.

The **limits** parameter is required if the spy keyword is quoted.

exchange ; exc: This parameter specifies an exchange package which is to be used to "qualify" the spy limits. Values accepted for this parameter are:

1. exc= JOB This value indicates that the spy limits apply only to any JOB exchange package.

2. exc= MONITOR ; MON This value indicates that the spy limits apply only to any MONITOR exchange package.

3. exc= UNQUAL ; UNQ This value is only used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is specified, any previous exchange package parameter qualification is overridden, and the spy limits will apply to any exchange package - JOB or MONITOR.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate a particular exchange package to be used to qualify the spy limits.

Default. If the **exchange** parameter is not entered for the command but there is a current exchange parameter qualification due to a previous **qualify_reference** command, then that value is used and interpreted exactly as described above. Otherwise, there is no qualification and the spy limits will apply to any exchange package - JOB or MONITOR.

Examples:

3.0 DESCRIPTION
3.7.22 ON

```
on pp_number=9
on pp=0 cpu
on 5
on ct
on spy l=0400001000(16)..0400001fff(16) exc=job
```

3.7.23 PP_BREAKPOINT ; PB

The purpose of the PP_BREAKPOINT command is to specify a breakpoint address in a simulated PP. Execution of the instruction at the PP breakpoint address in the specified PP causes the simulation to halt and a diagnostic message is displayed giving the PP number and breakpoint address.

```
pp_breakpoint [pp_number=<pp_number>]
              address=<pp_address>
              [frequency=(<i>[..<j>][,<k>])]
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

address ; a: This parameter specifies which PP address to breakpoint.

frequency ; f: This parameter specifies the frequency conditions which must be satisfied in order for the breakpoint to be honored. The frequency may be specified in one of three forms:

1. f= i This form will set a PP breakpoint which will be honored when the specified address is encountered on the ith time and never thereafter.

2. f= i..j This form will breakpoint on the ith time the address is encountered and every time thereafter until the jth time.

3. f= (i..j,k) This form will breakpoint on the ith time the address is encountered and every kth time thereafter until the jth time is reached or exceeded.

Default. If the frequency parameter is not specified on the command, then the PP breakpoint will be honored each time it is encountered.

3.0 DESCRIPTION
3.7.23 PP_BREAKPOINT ; PB

Examples:

```
pp_breakpoint pp_number=6 address=3004(8)
pp_breakpoint pp=0 a=500(8) frequency=2
pb 7 500(8) f=5..10
pb ,,2105(8)
```

3.7.24 PP_TRACE ; PT

The purpose of the PP_TRACE command is to establish a range of addresses to be traced in a particular simulated PP. Instructions executed within the range limits of the specified PP are interpreted with the results written to the Simulator trace file 'SESSMTF'. The PP trace remains active until nullified by a REMOVE_PP_TRACE command or a subsequent PP_TRACE command. Since only one PP trace may be active at any one time, successive PP_TRACE commands override any previous one. This is true whether or not the specified pp_number is the same.

PP trace information may be interspersed with CPU trace information if the CPU trace is also active.

```
pp_trace [pp_number=<pp_number>]
         limits=<start_address..end_address>
```

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The pp_number parameter is optional only if there is a currently active qualified PP number which was set up by a previous QUALIFY_REFERENCE command.

limits ; l: This parameter specifies the limits or range of PP addresses which are to be traced.

Examples:

```
pp_trace pp_number=3, limits=4100(8)..4210(8)
pp_trace pp=3 l=200(8)..1300(8)
pt 9 7400(8)..7450(8)
pt,,7400(8)..7450(8)
```

Sample output from trace file SESSMTF (some lines may be truncated due to ERS line length)

```
 05D  4214    LDN  07  (A)=000000/000007
 05D  4215    PSN  00
 05D  4216J   UJN  20
 05D  4236    ZJN  13  (A)=000007
 05D  4237    LPML 2020,73 (73)=0017 (2037)=111023 (A)=000007/00
 05D  4241    SHN  76  (A)=000003/000001
```

3.7.25 QUALIFY_REFERENCE ; QR

The purpose of the QUALIFY_REFERENCE command is to qualify the interpretation of certain parameters on subsequent Simulator commands. The effect of this command is to establish default values for the parameters of the commands, thereby eliminating the necessity of entering the parameters with every Simulator command. Only certain parameters can be qualified with this command. These are:

```
address_mode ; am
exchange ; exc
pp_number ; pp
file_name ; fn                                                    ;
dbug_file ; df                                                    ;
```

All Simulator commands for which the address_mode, exchange and pp_number parameters are defined will use the default values set up by this command. The only exceptions to this rule are for the pp_number parameter for the on and off commands.

The file_name parameter applies only to the memory display commands.

The dbug_file parameter applies only to the display_stack_frame and trace_back commands. ;

If a value is specified for these parameters on any subsequent Simulator command, then its value will <u>always</u> override the qualified value established by this command. It will <u>not</u> effect the qualification for other commands, however.

Parameters qualified by this command remain in effect until a subsequent QUALIFY_REFERENCE to 'unqualify' or to set up a new value.

```
qualify_reference [address_mode=<mode_designator>]
                  [exchange=<exchange_designator>]
                  [pp_number=<pp_number>]
                  [file_name=<local file name>]                   ;
                  [dbug_file=<local file name>]                   ;
```

address_mode ; am: This parameter establishes the default

value to be used for the address_mode parameter on certain Simulator commands for which the parameter applies, but for which no value is entered. The address_mode parameter dictates how the command's first_byte_address is to be interpreted - whether as a process virtual address (PVA) or as a real memory address (RMA). Quoting a value for the address_mode parameter on a command will always override the value established here by this command. Accepted values for this parameter are:

1. am= PVA This value sets the address mode qualification to process virtual addressing.

2. am= RMA This value sets the address mode qualification to real memory addressing.

Commands for which the address_mode qualification applies are:

change_memory
display_memory
display_memory_indirect

exchange ; exc: This parameter establishes a default 'frame of reference' for any subsequent Simulator commands for which the exchange parameter applies but for which no value is entered. The frame of reference designates a particular exchange package to which a command applies or which exchange package is to be used in virtual address translation.

Quoting a value for the exchange parameter on a command will always override the value established here by this command. Accepted values for this parameter are:

1. exc= JOB This specifies that the default frame of reference is to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value specifies that the default frame of reference is to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value specifies that no particular frame of reference is to be the default and removes any previous exchange qualification. Therefore, if a subsequent Simulator command does not specify an exchange parameter then it's frame of reference will be the exchange package of the current mode. This exchange package is determined by the mode bit in the Status Summary (SS) register - monitor or job mode - and located by the contents of the corresponding MPS or JPS register.

4. exc= <rma> This value specifies an address (RMA) of an exchange package that is to be used as the default frame of reference.

Commands for which the exchange parameter qualification applies are:

breakpoint
change_memory
change_register
display_exchange_package
display_memory
display_memory_indirect
display_register
on
trace
translate_pva

pp_number ; pp: This parameter qualifies a PP number which is to be the defaulted value on any subsequent Simulator command for which the pp_number parameter applies but for which no value is entered. The only exceptions to this are the on and off commands. Accepted values for this parameter are:

1. pp= <pp_number> This specifies the PP number which will be the default.

2. pp= UNQUAL ; UNQ This value will nullify any current pp_number default value.

Quoting a value for the pp_number parameter on any command will always override the default value established here by this command. Commands for which the pp_number qualification applies are:

change_pp_memory
change_pp_register
display_pp_memory
display_pp_register
load_pp_memory
pp_breakpoint
pp_trace
remove_pp_breakpoint
remove_pp_trace

This qualification also applies to the display parameter of the run_pp command.

file_name ; fn: Specifies the name of a local file which is to receive the output from the display memory commands. A wide printer line format is used when this output is directed to a file. Accepted values for this parameter are:

1. fn= <local file name> Name of file to receive the output.

2. fn= UNQUAL ; UNQ This value will nullify any file_name qualification currently in effect.

Quoting a value for the file_name parameter on any command will always override the default value established here by this command. Commands for which the file_name qualification applies are:

display_memory
display_memory_indirect
display_pp_memory

dbug_file ; df: This parameter specifies the name of the local file which contains the debug module address table produced by VELINK. Accepted values for this parameter are:

1. df= <local file name> Name of the file containing the debug module address table.

2. df= UNQUAL ; UNQ This value will nullify any dbug_file qualification currently in effect.

Quoting a value for the dbug_file parameter will always override the default value established here by this command. Commands for which the dbug_file

3.0 DESCRIPTION
3.7.25 QUALIFY_REFERENCE ; QR

```
            qualification applies are:                                  ;

                 display_stack_frame                                    ;
                 trace_back                                             ;
```

Examples:

```
     qualify_reference address_mode=pva exchange=job
     qualify_reference pp_number=5
     qr exc=unqual
     qr pp=9 fn=ppdump
     qr rma pp=unq
```

3.7.26 REMOVE_BREAKPOINT ; RB

The purpose of the REMOVE_BREAKPOINT command is to remove any CPU breakpoint address established by a previous BREAKPOINT command.

remove_breakpoint

3.7.27 REMOVE_PP_BREAKPOINT ; RPB

The purpose of the **REMOVE_PP_BREAKPOINT** command is to nullify a PP breakpoint address for the specified PP.

remove_pp_breakpoint [pp_number=<pp_number>]

pp_number ; pp: This parameter specifies the number of the PP to which this command applies. The PP may be in either an 'on' or 'off' state.

Default. The **pp_number** parameter is optional only if there is a currently active qualified PP number which was set up by a previous **QUALIFY_REFERENCE** command.

3.7.28 REMOVE_PP_TRACE ; RPT

The purpose of the REMOVE_PP_TRACE command is to nullify a currently active PP trace which was previously established by the PP_TRACE command. Since only one PP trace may be active at any one time, it is not necessary to specify the PP number.

remove_pp_trace

3.7.29 REMOVE_TRACE ; RT

The purpose of the REMOVE_TRACE command is to deactivate any current CPU trace which was previously set up by the TRACE command.

remove_trace

3.7.30 RESTART ; RS

The purpose of the RESTART command is to establish the simulated environment that was captured on the checkpoint file specified. This checkpoint file may be one that was previously created by the GENerate CheckPoint File (GENCPF) procedure, the Virtual Environment GENerator (VEGEN), the Simulator checkpoint command or the Extended Deadstart Dump utility (EDD). This command performs the same function as the RS parameter on the SIM180 procedure.

```
restart [file_name=<local_file_name>]
```

file_name ; fn: This parameter specifies the name of the local checkpoint file.

Default. If this parameters is not specified, then a completely new, empty simulated system envrionment is created.

Examples:

```
restart file_name=cpfile
rs fn=cpfile
rs filex
```

3.7.31 RUN ! R

The purpose of the RUN command is to initiate or resume CPU instruction simulation. The CPU simulation resumes execution according to the current contents of the SS, MPS, JPS registers and the contents of the associated exchange package registers. The results of an instruction's execution may optionally be displayed at the terminal.

Instruction simulation will halt after the specified count of CPU instructions.

```
run [count=<cpu_instruction_count>]
    [display]
```

count ! c: This parameter specifies the number of CPU instructions that are to be executed before halting simulation.

Default. If this parameters is not specified, the CPU instruction count is set to an infinite value.

display ! d: This keyword causes the interpretive execution of the CPU instructions to be displayed at the interactive user's terminal. This parameter is only valid if the **count** parameter is also specified.

Examples:

```
run count=100000
run c=20 display
r 10 d
r
```

Sample output from RUN with display in job mode:

```
run 5 display
J 0000148E 96650155 BRXGT
J 00001492 8B440001 ADDXQ        X4=0000000000000002
J 00001496 2A47     ADDAX        A7=30170000008A
J 00001498 8517001A SA
                              PVA=3017 000002CA 30170000008A
J 0000149C 84470000 LA           A7=301700000088
```

3.7.32 RUN_PP ; RP

The purpose of the RUN_PP command is to initiate or resume PP instruction simulation. The PP simulation resumes execution according to the current contents of the p-registers. All PPs in an on state will be simulated in a manner comparable to the hardware "barrel and slot" mechanism. The results of the specified PP's instruction execution may optionally be displayed at the terminal.

Instruction simulation will halt after the specified count of PP instructions.

```
run_pp [count=<pp_instruction_count>]
       [display=<pp_number>]
```

count ; c: This parameter specifies the number of PP instructions that are to be executed before halting simulation. Each PP that is in an 'on' state will execute this number of instructions.

Default. If this parameter is not specified, then the PP instruction count is is set to an infinite value.

display ; d: This parameter causes the interpretive execution of the instructions of the designated PP to be displayed at the interactive user's terminal. If this parameter is specified as a keyword, ie. display with no PP number, then the instructions that are displayed are from the PP that was previously qualified by the QUALIFY_REFERENCE command. If a PP had not been previously qualified, then this cannot be specified as a keyword. This parameter is only valid if the count parameter is also specified.

Default. If this parameter (keyword) is not specified then no PP instruction execution results are displayed.

Examples:

```
run_pp count=30000
run_pp c=10 display=3
rp 10 display
rp 5000
```

3.0 DESCRIPTION
3.7.32 RUN_PP ! RP

```
rp 15 5
```

Sample output from RUN_PP with display=5 (some lines truncated due to ERS line length)

```
05D  4214   LDN  07  (A)=000000/000007
05D  4215   PSN  00
05D  4216J  UJN  20
05D  4236   ZJN  13  (A)=000007
05D  4237   LPML 2020,73 (73)=0017 (2037)=111023 (A)=000007/000
05D  4241   SHN  76  (A)=000003/000001
```

3.7.33 TRACE ¦ T

The purpose of the TRACE command is to set up the address limits (pva's) for the CPU trace. Instructions executed within the range of the specified limits are interpreted with the results written to the Simulator trace file 'SESSMTF'. The trace remains in effect until nullified by entering the REMOVE_TRACE command or until a subsequent TRACE command is entered. Since only one CPU trace range may be active at any one time, successive TRACE commands override any previous one.

CPU trace information may be interspersed with PP trace information if the PP trace is also active.

```
trace limits=<start_pva..end_pva>
      [exchange=<exchange_designator>]
```

limits ¦ l: This parameter specifies the limits or range of pva's of CPU instructions to be traced. Ring numbers need not be entered as part of the PVA. They are ignored if entered and ignored when testing if the P-register is within the limits.

exchange ¦ exc: This parameter specifies an exchange package which is to be used to "qualify" the trace limits. Values accepted for this parameter are:

1. exc= JOB This value indicates that the trace limits apply only to any JOB exchange package.

2. exc= MONITOR ¦ MON This value indicates that the trace limits apply only to any MONITOR exchange package.

3. exc= UNQUAL ¦ UNQ This value is only used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is specified, any previous exchange package parameter qualification is overridden, and the trace limits will apply to any exchange package - JOB or MONITOR.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to

designate a particular exchange package to be used to qualify the trace limits.

Default. If the exchange parameter is not entered for the command but there is a current exchange parameter qualification due to a previous qualify_reference command, then that value is used and interpreted exactly as described above. Otherwise, there is no qualification and the trace limits will apply to any exchange package - JOB or MONITOR.

Examples:

```
trace limits=8000043f8(16)..8000044b2(16) exc=job
trace l=10000000(16)..10000ffff(16) exchange=unq
t 0..0ff(16) job
t 0b101000090000(16)..0b1010000a000(16)
```

Sample output from trace file SESSMTF:

```
 M 0000020A 0E0F     CPYSX          XF=000000006FFFFFEC
 M 0000020C A9FF0008 SHFX           XF=0000006FFFFFEC00
 M 00000210 B11F0020 KEYPOINT
 M 00000214 0200     EXCHANGE
                                 RMA=00011400 0000100600000216
 J 00001FB0 8E100078 ADDAQ          A0=200800000078
 J 00001FB4 84350002 LA             A5=200A000001D0
 J 00001FB8 82520005 LX             X2=00000000FFFFFFFF
 J 00001FBC 8D5F03CA ENTE           XF=00000000000003CA
 J 00001FC0 9602FF8C BRXGT
```

3.7.34 TRACE_BACK ; TB

The purpose of the TRACE_BACK command is to display information relevant to stack frames associated with an interrupted procedure and its predecessor procedures. 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

A value of zero is displayed in place of the stack frame save area address for the exchange package.

```
trace_back [stack_frame=<frame number>]
           [repeat_count=<count>]
           [address=<pva>]
           [exchange=<exchange_designator>]                    ;
           [dbug_file=<local file name>]                       ;

     stack_frame ; sf: This 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 (the exchange        ;
          package) or with the stack frame associated with the;
          address parameter, if specified. Stack frame two is ;
          associated with the first stack frame procedure's   ;
          predecessor, etc. The valid range for this          ;
          parameter is 1 .. 999.                              ;

          Default. If the stack frame parameter is not        ;
          specified, then the first stack frame for which     ;
          information is displayed will be the exchange       ;
          package or the stack frame associated with the      ;
          address parameter (if specified).                   ;

     repeat_count ; rc: This parameter specifies the total    ;
          number of stack frames for which information is to  ;
          be displayed. The valid range for this parameter is ;
          1 .. 999. If this value exceedes the number of      ;
          stack frames, output will terminate with the last   ;
          frame in the stack.                                 ;

          Default. Information for one stack frame is         ;
```

displayed.

address ! a: This parameter specifies the process virtual address (PVA) of a stack frame save area at which the trace back begins (stack frame number one). Ring numbers are optional and ignored if entered as part of a PVA.

Default. If the address parameter is not specified, stack frame number one is the interrupted procedure (the exchange package).

exchange ! exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ! MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ! UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

dbug_file ; df: This parameter specifies the name of the local file which contains the debug module address table produced by VELINK. Values accepted for this parameter are:

1. df= <local file name> Any valid NOS file name is accepted. This file name temporarily overrides any qualified dbug_fileestablished previously by a qualify_reference command.

2. df= UNQUAL ; UNQ This value is used to negate any dbug_file qualification in effect due to a previous QUALIFY_REFERENCE command. No module names and offsets appear in the displayed information.

Default. If the dbug_file parameter is not entered for the command, the dbug_file name is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the dbug_file for subsequent commands, then that value is used as the dbug_file name for this command.

(2) If the dbug_file name was not previously qualified by QUALIFY_REFERENCE, the dbug_file remains unqualified.

Examples:

```
trace_back sf=1 rc=999
trace_back rc=6
```

```
tb a=300000078(16) sf=2 rc=3
tb exc=mon
```

3.7.35 TRANSLATE_PVA ; TP

The purpose of the TRANSLATE_PVA command is to translate a process virtual address (PVA) into a CPU real memory address.

translate_pva address=<pva> [exchange=<exchange_designator>]

address ; a: This parameter specifies the process virtual address that is to be translated. Ring numbers are optional and ignored if entered as part of a PVA.

exchange ; exc: This parameter specifies the 'frame of reference' for the command, ie. it specifies to which exchange package this command applies or which exchange package is to be used for virtual address translation. Values accepted for this parameter are:

1. exc= JOB This value sets the frame of reference for this command to be the current job exchange package as located by the contents of the Job Process State (JPS) register.

2. exc= MONITOR ; MON This value sets the frame of reference for this command to be the current monitor exchange package as located by the contents of the Monitor Process State (MPS) register.

3. exc= UNQUAL ; UNQ This value is used to override any exchange parameter qualification in effect due to a previous QUALIFY_REFERENCE command. When UNQUAL is given, the frame of reference for this command is the exchange package of the current mode as determined by the mode bit in the Status Summary (SS) register. If the current CPU mode is monitor, then the frame of reference is the monitor exchange package pointed to by the MPS. If the current mode is job , the exchange package pointed to by the JPS is the frame of reference.

4. exc= <address> The real memory address (RMA) of an exchange package may be specified to designate that exchange package as the frame of reference for this command.

Default. If the exchange parameter is not entered for the command, then the frame of reference for the command is determined in one of two ways:

(1) If a previous qualify_reference command had qualified the exchange parameter for subsequent commands, then that value is used as the frame of reference for this command.

(2) If the exchange parameter was not previously qualified by QUALIFY_REFERENCE, then the frame of reference is determined from the current mode in the Status Summary (SS) register and its corresponding process state register - MPS or JPS.

Examples:

```
translate_pva address=100004fff(16)
translate_pva a=0acdb(16)
tp 81010000fddd(16)
```

Sample output:

```
translate_pva a=0e00001528(16)
RMA= 0002 3528
```

3.8 SPECIAL INSTRUCTION PROCESSING

This section explains any special instruction processing done by the Simulator and any instruction deviations from the hardware description presented in the CYBER 180 MIGDS.

3.8.1 I/O INSTRUCTION

Input or Output a message at [Aj] plus Q + 1, mode per k. This is a special CP instruction (op code FF) that is defined as an unimplemented instruction for the hardware, but is processed by the Simulator as an I/O instruction (format j k Q). The instruction gives the user a means of communicating with a simulated 180 program from a time-sharing terminal or to and from files if running in batch mode.

This instruction transfers a field of bytes between the terminal and central memory, with the direction of the transfer determined by the k field, and the length of the transfer determined by the binary contents of the first byte in the field - the Variable Field Indicator (VLI). The VLI is addressed be expanding the Q field to 32 bits by means of sign extension and then adding the results to the rightmost 32 bits of the PVA contained in the Aj register. The remaining bytes comprise the ASCII characters of the message.

k=0. Output. The message in the byte field is displayed at the users terminal.

k=1. Output_and_halt. The message is displayed at the terminal as above, but once completed the CP is halted pending further Simulator commands.

k=2. Input. The input message is solicited from the user. First the Simulator displays the three character prompt ENT followed by a question mark. The input message may now be entered and will be placed starting at the byte immediately following the VLI byte. The actual length of the message entered is placed into the VLI byte.

k=3. Output_prompt_and_Input The prompt message in the byte field is displayed at the terminal followed by a question mark; then input is solicited from the user. This option allows the user to display his own prompt for input. The 'ENT' prompt is not displayed as it is for the k=2 option. The message

string used for the prompt is terminated by a period (.). The VLI for this option determines the length of the input message which is placed in the byte field in the same manner as the k=2 option.

Abnormal conditions. If the VLI byte is encountered by the Simulator as zero (0), then the diagnostic message 'VLI INPUT ERROR' or 'VLI OUTPUT ERROR' will be displayed at the terminal. In addition, if the error occurs for input (k=2) then the external bit in the MCR will be set.

3.8.2 KEYPOINT INSTRUCTION

The CP Keypoint instruction (op code B1) is simulated as described in the CYBER 180 MIGDS with the exception that no transmission is done to the Performance Monitoring Facility (PMF). This facility currently is not simulated.

In place of this, the Simulator writes keypoint information to the SESSMKF file. Each time the appropriate keypoint conditions are met for recording to the PMF, the following CYBIL record is written to SESSMKF. (The PMF keypoint request flag is assumed as always set.)

```
TYPE
  zsmtprt_pmf_record_type = (keypoint),
  zsmtpmf_pmf_record = RECORD
    CASE record_type: zsmtprt_pmf_record_type OF
    =keypoint=
      keypoint_time: 0 ..  3FFFFFF(16),
      keypoint_class: 0 ..  0f(16),
      keypoint_code: 0 ..  0FFFFFFFF(16),
    CASEND,
  RECEND;
```

3.8.3 EXECUTE ALGORITHM INSTRUCTION

The CP Execute Algorithm instruction (op codes C0 - C7) is utilized by the Simulator to "escape" from normal CPU instruction simulation. Four types of escapes are defined for the Simulator which correspond to the four instruction op codes C4 - C7. The remaining op codes for this instruction (C0 - C3) cause an unimplemented instruction condition as specified in the MIGDS.

The four types of escapes are:

C4 This will escape CPU instruction simulation and initiate PP (IOU) instruction simulation for all PPs in an on state. Control will return only after a similar PP "escape" instruction (see following section on the PP Escape Instruction) is encountered. If all PP's are off, then this instruction will execute as a NO-OP.

C5 This escapes to the Simulated NOS/VE Program Manager Interface package. Control returns upon completion of the simulated function.

C6 This escapes to the Simulated NOS/VE I/O interfaces package. Control returns upon completion of the simulated function.

C7 Escapes to a Hardware Checkout System (HCS) physical I/O simulation module. Control returns when complete.

3.8.4 PP ESCAPE INSTRUCTION

The PP escape instruction is used to escape from PP instruction simulation and initiate CP instruction simulation. Control will return only after a corresponding CP escape instruction has been executed (see the above section on the Execute Algorithm Instruction). If the CP is off, then this instruction executes as a NO-OP instruction. The instruction used for the escape is defined as a PASS instruction in the MIGDS, as follows:

107707(8)

3.9 DIAGNOSTIC MESSAGES

The following diagnostic messages may appear in response to Simulator commands. Some messages such as the Simulator banner are purely informative. Others denote error conditions caused by incorrect command syntax.

The messages described here are messages originating from the Simulator and are identified by the prefix ** X SM followed by a number in the range of 6000 - 6999. The only messages within this range that are not described in this ERS fall in the 6350 - 6399 range. These are simulated NOS/VE procedures messages and are described in the ERS for Simulated NOS/VE Program Interfaces. The X in the prefix is either an I, W, E or F referring to a severity level of Informational, Warning, Error or Fatal.

Other messages outside the 6000 - 6999 range may also be encountered during a simulation run. These messages originate from supporting utilities which the Simulator calls such as the System Command Language procedures, Message Generator, Command Processor, etc. No attempt is made to define these messages, as most should be self-explanatory and usually result from from syntactical errors. Users may refer to the the appropriate ERS for a definition of these messages.

Other questions or problems concerning error conditions should be directed to the SES analyst, Curt Rupert 482-2583.

6001 xxx COMMAND UNKNOWN
The command indicated by xxx is unknown to the Simulator.

6002 INSUFFICIENT HEAP SPACE FOR xxx
This message indicates that some unexpected condition has occurred which could not be processed due to lack of heap space. The particular condition is specified by xxx. The user should contact the SES analyst if this error is encountered.

6003 REGISTER ID AND VALUE LISTS NOT EQUAL IN LENGTH
This message occurs when the number of register values does not equal the number of register names in the register name list for the change_register or change_pp_register commands.

6004 CHECKPOINT FILE BAD, ENTER -BYE-
This message follows SES.SIM180 if the restart parameter specifies a checkpoint file that is bad. The file is more than likely not a valid checkpoint file, and the user must enter bye to exit from the Simulator, as all commands will fail.

6005 SIM CONTROL INFO REINITIALIZED, REGS AND MEM LOADED
This message is displayed after a RESTART command or after SES.SIM180 is entered where the restart parameter specifies a checkpoint file that has been outdated by new requirements on the current version of the Simulator. This is usually caused by new requirements for control information and tables that are saved at checkpoint time. Memory and processor registers are loaded and unaffected, only internal simulator control information such as run counts, breakpoints, etc. are lost.

6006 TERMINATED
A terminal interrupt condition has terminated the command.

6007 ERROR IN LOADING OVERLAY
An operating system or hardware problem has prevented successful loading of a Simulator overlay. Notify development center support.

6022 INVALID NAME FOR xxx PARAMETER
The name quoted for the xxx parameter is not an acceptable value. The user should double check the syntax of the command in question.

6035 CPU INTERRUPTED AT P=xxx, STATE=yyy
This informative message occurs when a terminal interrupt condition is encountered during instruction simulation initiated by the run or run_pp commands. The user may enter any Simulator command at this point. The xxx is the address in the P register at the time of interruption, and yyy specifies the CP state at the time - MONITOR or JOB.

6036 PPxx INTERRUPTED AT P=yyy
This informative message occurs when a terminal interrupt condition is encountered during instruction simulation initiated by the run or run_pp commands. The user may enter any Simulator command at this point. The xx is the number of the PP (decimal) interrupted and yyy is the address at the time of interruption.

6048 INSUFFICIENT OR INCONSISTENT PARAMETERS SPECIFIED
This message indicates that parameters or combination of parameters specified for the preceding command are in conflict with the correct syntax of the command. The user should double check the syntax of the command in question.

6052 CYBER 180 SIMULATOR Vxx LEV yyy (MIGDS REV z)
This is the Simulator banner message which indicates that the simulated environment has been established, and any Simulator command may now be entered. The xx indicates the current version of the Simulator and is increased by 1 with each SES Release. The yyy is the current level which is increased by 1 with every new prerelease and SES Release of the Simulator. The z indicates the current hardware MIGDS Revision level with which the Simulator fully conforms.

6101 BAD LOAD FILE DIRECTORY ON xxx
The file indicated by xxx is not the correct format for the load_memory command.

6102 NO BINARY TEXT ON xxx
The load_memory command or load_pp_memory command did not encounter any binary text on the file xxx.

6103 CAN NOT LOAD MEMORY FROM xxx
The load_memory command was unable to write information from file xxx into simulated central memory.

6105 BAD PP LOAD FILE DIRECTORY ON xxx

The file indicated by xxx is not the correct format for the **load_pp_memory** command.

6106 PP BINARY TEXT ON xxx UNKNOWN
The **load_pp_memory** command encountered an unknown format on file xxx.

6107 xxx IS NOT A PP LOAD FILE
The file xxx is not a valid format to be loaded by the **load_pp_memory** command.

6108 PP MEMORY OVERFLOW, xxx
File xxx contains an excess of binary text for loading into simulated PP memory.

6200 EXPECTING HEX VALUE FOR xxx
This message indicates that an incorrect hexadecimal value was specified for the xxx parameter, eg. an alpha other than a thru f was found in the value.

6201 INVALID REGISTER NUMBER
An illegal register number has been specified on the **display_register**, **change_register** or **display_memory_indirect** command. User should recheck syntax of command in question.

6204 INVALID REGISTER NAME -xxx-
The illegal register name xxx has been specified on the last command. The user should recheck the syntax of the command in question.

6205 REGISTER VALUE TOO LARGE
A register value specified for the **change_register** or **change_pp_register** command is too large to fit into the quoted register. The user should check the hardware description of the register to determine its size.

6206 INVALID NUMERIC SIZE FOR xxx PARAMETER
The value entered for the xxx parameter for the last

command is too large for the parameter. The user should recheck the syntax of the command in question.

6207 INVALID ASCII PARAMETER SIZE
A value entered as an ASCII string for a register or memory field for the last command is too large. The user should recheck the syntax of the command in question.

6209 CYBER 180 ADDRESS TOO LARGE
A value entered for a central memory address parameter for the last command was too large. User should recheck the current simulated memory configuration.

6212 INVALID CYBER 180 ADDRESS
The central memory process virtual address (PVA) specified on the last command was not translatable to a real memory address. User must verify that the virtual address translation tables are set up correctly, ie. page and segment tables.

6213 PAGE NOT IN REAL MEMORY
The last command was submitted which required access to a CYBER 180 memory page that could not be found in real memory. The user should recheck the entry of the command in question and validate any PVA parameter value. The Simulator does not perform any implicit memory paging, and all PVA must translate to real memory.

6214 ADDRESS SPECIFICATION ERROR
The PVA specified on the previous command was illegal because the sign bit (bit 32) in the byte number (BN) field within the PVA was found to be set.

6217 CPU BREAKPOINT ENCOUNTERED AT P=xxx, STATE=yyy
This informative message indicates that the CP breakpoint address previously entered by the breakpoint command, has been encountered. The P

register contents xxx for the state yyy at the time of the halt is included in the message. Any valid Simulator command may be entered at this point.

6218 CPU HALT AT P=xxx, STATE=yyy
This informative message indicates that the CP instruction simulation has encountered a halt condition at the P register address specified by xxx, in the state indicated by yyy (MONITOR or JOB). The user may enter any valid Simulator command at this point.

6220 PPxx BREAKPOINT ENCOUNTERED AT P=yyy
This informative message occurs when a previously entered PP breakpoint address has been encountered. The PP number in which the breakpoint occurred is indicated by xx and the p-register address by yyy. Any valid Simulator command may be entered at this point.

6221 PPxx HALT AT P=yyy
This informative message indicates that the PP instruction simulation in the PP numbered xx has encountered a halt condition at the P register address specified by yyy. Any valid Simulator command may be entered at this point.

6222 INVALID VALUE FOR xxx PARAMETER
The user quoted a value for the parameter indicated by xxx that was unacceptable to the command. The user should review the command syntax.

6224 xxx - FILE NOT LOCAL
The file indicated by xxx specified on the previous command must exist as a local file. The file should be ACQUIREd and the command reentered.

6228 EXPECTING POSITIVE VALUE FOR xxx PARAMETER
The value entered for the parameter indicated by xxx must be entered with a positive value.

6229 EXPECTING INTEGER VALUE FOR xxx PARAMETER
The value entered for the parameter indicated by xxx for the last command was invalid and must be an integer, and not a name or string.

6230 EXPECTING INTEGER OR STRING VALUE FOR xxx PARAMETER
The value entered for the parameter indicated by xxx for the last command was not an integer or character string.

6231 DESCENDING RANGE QUOTED FOR xxx PARAMETER
The range entered for the parameter indicated by xxx for the last command was entered in the reverse order, and must be entered in as an ascending range.

6232 <STRING> VALUE NOT ALLOWED FOR xxx PARAMETER
The value entered for the parameter indicated xxx for the last command cannot be a string value.

6233 DUPLICATE VALUE FOR -xxx-
The parameter indicated by xxx was specified at least twice on the previous command.

6234 ADDRESS RANGE CANNOT SPAN SEGMENTS
The range of PVA's specified for the SPY option of the ON command must be within the same segment number.

6301 PP NOT IN CONFIGURATION
The **pp_number** parameter value for the last command specified a PP number that is not included in the simulated CYBER 180 hardware system. The user should use the **display_info** command to determine the numbers of the PPs in the configuration.

6302 PP NUMBER NOT QUALIFIED - VALUE REQUIRED
No value was quoted for the **pp_number** parameter for the last command. Also, there is no current **pp_number** parameter qualification. This parameter is only optional if it has been previously qualified

by the **qualify_reference** command. The user must either qualify the **pp_number** or enter it with each command.

6303 ALL PPS ARE OFF
The **run_pp** command was entered with all the PPs in an off state. The user should use the **on** command to turn on the PPs which are to be simulated. The **display_info** command gives the on/off status of all the PPs.

6304 CPU IS OFF
The **run** command was entered with the CPU in an off state. The **display_info** command can be used to inspect the on/off status of the CP and the **on** command can be used to turn it on for instruction simulation.

6350 - 6399
See the Simulated NOS/VE Program Interfaces ERS.

4.0 APPENDIX A - SAMPLE SIMULATOR SESSIONS

4.0 APPENDIX A - SAMPLE SIMULATOR SESSIONS

The following sample Simulator sessions give some representative examples of Simulator command usage. Some command output lines may be truncated due to line length limitations of this document.

4.1 SAMPLE SESSION # 1

This sample session is an example of a user interested only in CPU simulation. Most of the CPU commands are included at least once in this example. The format of the trace output written to file SESSMTF as a result of this session is shown under the TRACE command description.

```
/ses.sim180 rs=mtrck
  ** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)
? display_memory fba=300004030(16) rc=4 am=pva exc=job
  SEGMENT= 0003
  00004030  00241022 22200000   00003333 55550000   $ " "      33 UU
  00004040  00000012 00120000   00000000 02424200                BB
? help change_memory
  CHANGE_MEMORY  FIRST_BYTE_ADDRESS= P1  BYTE_COUNT= P2
                 REPEAT_COUNT= P3,  MEMORY_VALUE= P4
                 EXCHANGE= P5  ADDRESS_MODE= P6
  CM  FBA= P1,  BC= P2,  RC= P3,  MV= P4,  EXC= P5,  AM= P6
? qualify_reference am=pva exc=job
? change_memory fba=30000403a(16) mv=3456(16) bc=2
? display_memory fba=30000403a(16) bc=2
  SEGMENT= 0003
  0000403A  3456                                    4V
? display_register x=8
       X8 = 0000 0000 0000 0000
? change_register x=8 xv=33(16)
? 'define,tempckx'
? checkpoint fn=tempckx
? run 6 display
  M 000011F8 8D070011 ENTE        X7=0000000000000011
  M 000011FC 8D080022 ENTE        X8=0000000000000022
  M 00001200 0200     EXCHANGE
                                RMA=00000580 0000B00000001202 ..
  J 00004000 9400006C BRXEQ
```

4.0 APPENDIX A - SAMPLE SIMULATOR SESSIONS

4.1 SAMPLE SESSION # 1

```
  J 000040D8 94000062 BRXEQ
  J 0000419C 3053     ENTP          X3=0000000000000005
? trace limits=10003000(16)..10004fff(16)
? breakpoint address=33c8(16)
? display_info
  QUALIFIERS:  ADDRESS_MODE= PVA  EXCHANGE= JOB  PP= UNQUAL                ;
               FILE_NAME= UNQUAL  DBUG_FILE= UNQUAL                        ;
  LAST_CP_RUN_COUNT=            6  TOTAL=         9006
  LAST_PP_RUN_COUNT=            0  TOTAL=            0
  CP_TIMING  OFF

  CONFIGURATION:      CPU      PP00 PP01 PP02 PP03 PP04 PP05 PP06 PP07
  STATUS:             ON        OFF  OFF  OFF  OFF  OFF  OFF  OFF  OFF
  BRKPT:REF XP        JOB
        ADDRSS  000 0000 33C8
        REMAIN      9999
  TRACE:REF XP        JOB
        LIMITS  001 0000 3000
                001 0000 4FFF
? run
  ** I SM 6217: CPU BREAKPOINT AT P=0000 B000 0000 33C8, STATE=JOB
? display_exchange_package
      P=0000 B000 0000 33C8   MCR=0000                    UCR=0000
    MDW=0000 0000 0000 0000    MM=FFFF                     UM=FFFF
    UTP=0000 0000 0000       VMID=   0                  UVMID=   0
     TP=B000 0000 3E10         TE=   0                   LPID=00
    DLP=0000 0000 0000         DM=00                       DI=00
    STA=0000 3338             STL=0000                    LRN=   F
     KC=0000 0000              KM=0000                    KCN=   0
    PIT=FFFF FFF5             CFF=   0                    OCF=   0
    KEF=   0                  PND=   0                     BC=0000 0000
    MDF=0000
? display_register s=(ss,jps) a=0..3 exc=mon
          SS=0000 0000 0000 0008                JPS=0000 3E60
       A0 = B002 0000 2718                   A1 = B002 0000 2718
       A2 = B002 0000 2718                   A3 = B002 0000 2678
? display_memory_indirect s=jps am=rma
  ABSOLUTE 0000
  00003E60  0000 B000 0000 33C8                                  3
? translate_pva a=20002718(16) exc=mon
  RMA= 0001 5418
? remove_trace
? remove_breakpoint
? run 750
? display_register (p,ss)
          P=0000 B000 0000 5280                SS=0000 0000 0000 0008
? bye                                                                    ;
  REVERT.    END SIM180                                                  ;
```

4.0 APPENDIX A - SAMPLE SIMULATOR SESSIONS
4.2 SAMPLE SESSION # 2

4.2 SAMPLE_SESSION_#_2

This example shows a user interested only in two concurrent PP simulations and no CPU simulation. Most PP oriented commands are used in the example. The format of the trace output written to SESSMTF is shown under the PP_TRACE command description.

```
/ses.sim180                                                              ;
  ** I SM 6052: CYBER 180 SIMULATOR V6.5 LEV 127 (MIGDS REV S)           ;
? 'acquire,ppbina,ppbinb'
? load_pp_memory fn=ppbina pp=3 fwa=101(8)
? change_pp_register pp=3 regs=p vals=260(8)
? display_pp_register pp=3
  03D  P= 0260  A= 000000
? qualify_reference pp=3
? change_pp_memory ,,fwa=3300(8) mv=5555(8) rc=10
? display_pp_memory ,,fwa=3300(8) rc=13
  PP03
  3300  005555 005555 005555 005555
  3304  005555 005555 005555 005555
  3310  005555 005555 000000 000000
  3314  002203  RC
? on pp=3
? qualify_reference pp=4
? load_pp_memory,,fn=ppbinb
? pp_breakpoint pp=3 address=7710(8) f=1
? pp_trace,,limits=1000(8)..2000(8)
? on pp=4
? display_info
  QUALIFIERS:  ADDRESS_MODE= RMA  EXCHANGE= UNQUAL  PP= 04D               ;
               FILE_NAME= UNQUAL  DBUG_FILE= UNQUAL                       ;
  LAST_CP_RUN_COUNT=            0  TOTAL=           0
  LAST_PP_RUN_COUNT=            0  TOTAL=           0
  CP_TIMING  OFF

  CONFIGURATION:      CPU      PP00 PP01 PP02 PP03 PP04 PP05 PP06 PP07 P
  STATUS:             ON        OFF  OFF  OFF  ON   ON   OFF  OFF  OFF
  BRKPT:ADDRSS                                7710
        REMAIN                                   1
  TRACE:LIMITS                                      1000
                                                    2000
? help off
  OFF  PP_NUMBER= P1  CPU  CP_TIMING SPY
  OFF  PP= P1  CP  CT SPY
? off cpu
? qualify_reference pp=unqual
```

4.0 APPENDIX A - SAMPLE SIMULATOR SESSIONS
4.2 SAMPLE SESSION # 2

```
? run_pp 100
? remove_pp_trace
? run_pp
  ** I SM 6220: PP03 BREAKPOINT ENCOUNTERED AT P=7710
? remove_pp_breakpoint pp=3
? checkpoint fn=saveit
? bye
  REVERT.    END SIM180
```

5.0 APPENDIX B - SAMPLE SPY OUTPUT

5.0 APPENDIX B - SAMPLE SPY OUTPUT

The following is a sample "SPY" output (see on and off commands) for a CPU run of 10000 instructions with SPY limits =0a000000001000(16) .. 0b000000002000(16) and an exchange designator exc=JOB.

Counts for the various range "bins" are only incremented when the P-register matches the exchange designator - in this case JOB. This explains the difference between the total run count and the total sample count, i.e., 2774 instructions in MONitor mode.

The actual SPY output is formatted for a wide printer, so columns 50 - 114 were deleted to fit it into this ERS. Also, approximately 90 lines were deleted to keep it somewhat short.

```
  P REGISTER SAMPLES FOR SEGMENT  00A, BY:
  SAMPLES BELOW RANGE          00000160
  SAMPLES IN RANGE             00001850
  SAMPLES ABOVE RANGE          00005216

  TOTAL SAMPLES                00007226

BYTE OFFSET.
                      0.00      10.00      20.00          100.00
                        +          +          +              +
                      ---------------------------|-----------
BELOW       00001000  I**        .          .          .I 00000160
00001000 - 0000101B   I          .          .          .I 00000000
0000101C - 00001037   I          .          .          .I 00000000
00001038 - 00001053   I          .          .          .I 00000000
00001054 - 0000106F   I          .          .          .I 00000000
00001070 - 0000108B   I          .          .          .I 00000000
0000108C - 000010A7   I          .          .          .I 00000000
000010A8 - 000010C3   I          .          .          .I 00000000
000010C4 - 000010DF   I          .          .          .I 00000000
000010E0 - 000010FB   I          .          .          .I 00000000
000010FC - 00001117   I          .          .          .I 00000002
```

5.0 APPENDIX B - SAMPLE SPY OUTPUT

```
00001118 - 00001133  I          .          .          :  :          .I 00000008
00001134 - 0000114F  I          .          .          :  :          .I 00000007
00001150 - 0000116B  I          .          .          :  :          .I 00000008
0000116C - 00001187  I          .          .          :  :          .I 00000007
00001188 - 000011A3  I          .          .          :  :          .I 00000008
000011A4 - 000011BF  I          .          .          :  :          .I 00000007
000011C0 - 000011DB  I          .          .          :  :          .I 00000007
000011DC - 000011F7  I          .          .          :  :          .I 00000007
000011F8 - 00001213  I          .          .          :  :          .I 00000008
00001214 - 0000122F  I          .          .          :  :          .I 00000005
00001230 - 0000124B  I          .          .          :  :          .I 00000005
0000124C - 00001267  I          .          .          :  :          .I 00000007
00001268 - 00001283  I          .          .          :  :          .I 00000009
00001284 - 0000129F  I***       .          .          :  :          .I 00000272
000012A0 - 000012BB  I****      .          .          :  :          .I 00000360
000012BC - 000012D7  I***       .          .          :  :          .I 00000258
000012D8 - 000012F3  I***       .          .          :  :          .I 00000234
000012F4 - 0000130F  I          .          .          :  :          .I 00000056
00001310 - 0000132B  I*         .          .          :  :          .I 00000096
0000132C - 00001347  I          .          .          :  :          .I 00000008
00001348 - 00001363  I          .          .          :  :          .I 00000008
00001364 - 0000137F  I          .          .          :  :          .I 00000010
```

```
00001E00 - 00001E1B  I          .          .          :  :          .I 00000000
00001E1C - 00001E37  I          .          .          :  :          .I 00000000
00001E38 - 00001E53  I          .          .          :  :          .I 00000000
00001E54 - 00001E6F  I          .          .          :  :          .I 00000000
00001E70 - 00001E8B  I          .          .          :  :          .I 00000000
00001E8C - 00001EA7  I          .          .          :  :          .I 00000000
00001EA8 - 00001EC3  I          .          .          :  :          .I 00000000
00001EC4 - 00001EDF  I          .          .          :  :          .I 00000000
00001EE0 - 00001EFB  I          .          .          :  :          .I 00000000
00001EFC - 00001F17  I          .          .          :  :          .I 00000000
00001F18 - 00001F33  I          .          .          :  :          .I 00000000
00001F34 - 00001F4F  I          .          .          :  :          .I 00000000
00001F50 - 00001F6B  I          .          .          :  :          .I 00000000
00001F6C - 00001F87  I          .          .          :  :          .I 00000000
00001F88 - 00001FA3  I          .          .          :  :          .I 00000000
00001FA4 - 00001FBF  I          .          .          :  :          .I 00000000
00001FC0 - 00001FDB  I          .          .          :  :          .I 00000000
00001FDC - 00001FF7  I          .          .          :  :          .I 00000000
00001FF8 - 00002000  I          .          .          :  :          .I 00000000
ABOVE      00002000  I**************************     :  :          .I 00005216
                     ---------------------------    :  :----------
                     +          +          +        :  :          +
                   0.00       10.00      20.00      :  :       100.00
```

Table of Contents