INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONE: 2240-1
CABLE: CENTRATOM - TELEX 460392-1

SECOND SCHOOL ON ADVANCED TECHNIQUES
IN COMPUTATIONAL PHYSICS
(18 January - 12 February 1988)

SMR.282/8

VMS EXERCISES

M.NATIELLO
Quantum Chemistry Group, University of Uppsala, Sweden

Ex. 1: Perform the demo commands, by invoking `@[demo]demo1` (and demo2, demo3, etc.). If you want to modify these commands, copy them first to your default directory with copy *name.com* `sys$login:*`, so that you perform the modifications on your local copy only.

Ex. 2: Change the name of your current process from a FORTRAN program.

1. Read a new process name (lib$get_input).
2. Change the process name (sys$setprn).
3. Write the error message in a nice fashion:
   a. Get the error message (lib$sys_getmsg)
   b. Uppercase the first letter in message (str$upcase).
   c. Write the message.

Ex. 3: Read a string from the terminal with "no echo" and write it to the terminal, using a FORTRAN program.

1. Assign a channel to the terminal (sys$assign).
2. Issue an IO service to read from the channel with "no echo" (sys$qiow, with some value in the "func" argument. Check error status with `IOSB(1)`).
3. Issue an IO service to write to the channel (sys$qio).

Ex. 4: Associate a `common flag cluster` to two processes to demonstrate communication.

A. Process 1:

   1. Associate a `flag cluster` (`sys$ascefc(`*nn*`,`*name*`,,)`. The name is crucial for communication.).
   2. Clear the flag (sys$clref).
   3. Do something until the flag is set (sys$readef).
   4. End when the flag is set.

B. Process 2:

   1. Associate a flag cluster with the same name as Process 1.
   2. Wait some time (lib$wait).
   3. Set the flag (sys$setef).
   4. End.

Ex. 5: Read input for a registration file (some items to be filled in each card) from the upper half of the terminal. Show the complete input in the lower half of the terminal. Erase the display and print a greeting when leaving. The upper half of the terminal should scroll when filled.

1. Create a pasteboard (smg$create_pasteboard; a channel to the screen).
2. Create a display (smg$create_virtual_ display) and paste it to the pasteboard (smg$paste_virtual_display).
3. Create a virtual keyboard (smg$create_virtual _keyboard; reads from the screen).
4. Make the display look nice (smg$set_display _scroll_region, smg$scroll_display _area, smg$label_border, smg$put_line).
5. Create and paste a second display to the lower half of the screen.
6. Prompt for reading form the display (smg$read_string).
7. Write the input in the second display (smg$erase_display, smg$put_line).
8. Exit nicely on `CTRL/Z` (Check for `end-of-file` while reading with smg$_eof).

```
include '($smgdef)'
include '($prcdef)'
include '($ssdef)'
include '($lnmdef)'
include '($iodef)'

lib$get_ef
lib$get_input
lib$sys_getmsg
lib$wait

sys$setprn
sys$trnlnm
sys$creprc
sys$wake
sys$schdwk
sys$hiber
sys$delprc
sys$assign
sys$qiow
sys$qio
sys$clref
sys$asctim
sys$bintim
sys$setimr
sys$readef
sys$setast

smg$create_pasteboard
smg$create_virtual_display
smg$paste_virtual_display
smg$create_virtual_keyboard
smg$label_border
smg$set_display_scroll_region
smg$scroll_display_area
smg$put_line
smg$read_string
smg$erase_display

str$upcase

if(.not.status) call lib$stop(%val(status))
f1=io$_readprompt.or.io$m_noecho
if(.not.iosb(1)) lib$stop(%val(iosb(1)))
f2=io$_writevblk
parameter       smg$_eof=1213442
outstyle=smg$m_blink.or.smg$m_reverse
if(status.eq.smg$_eof)
do while (status.ne.ss$_wasset)
```

$QIO Request service queues an I/O request to a channel associated with a device. The $QIO service completes asynchronously; that is, it returns to the caller immediately after queuing the I/O request, without waiting for the I/O operation to complete. For synchronous completion, use the $QIOW service. The $QIOW service is identical to the $QIO service in every way except that $QIOW returns to the caller after the I/O operation has completed. Format:

SYS$QIO([efn] ,chan ,func [,iosb] [,astadr] [,astprm]

[,p1] [,p2] [,p3] [,p4] [,p5] [,p6])

efn (mechanism: by value)

Event flag that $QIO is to set when the I/O operation actually completes. The efn argument is a longword value containing the number of the event flag.

chan (mechanism: by value)

I/O channel that is assigned to the device to which the request is directed. The chan argument is a word value containing the number of the I/O channel; however, $QIO uses only the low-order word.

func (mechanism: by value)

Device-specific function codes and function modifiers specifying the operation to be performed. The func is a longword value containing the function code.

iosb (mechanism: by reference)

I/O status block to receive the final completion status of the I/O operation. The iosb is the address of the quadword I/O status block.

astadr (mechanism: by reference)

AST service routine to be executed when the I/O completes. The astadr argument is the address of a longword value that is the entry mask to the AST routine.

astprm (mechanism: by value)

AST parameter to be passed to the AST service routine. The astprm argument is a longword value containing the AST parameter.

p1 to p6 (mechanism: by reference)

Optional device- and function-specific I/O request parameters.

$ASSIGN (1) provides a process with an I/O channel so that input/output operations can be performed on a device or (2) establishes a logical link with a remote node on a network. Format:

SYS$ASSIGN (devnam ,chan ,,)

devnam (mechanism: by descriptor--fixed length string descriptor)

Name of the device to which $ASSIGN is to assign a channel. The devnam argument is the address of a character string descriptor pointing to the device name string.

chan (mechanism: by reference)

Number of the channel that is assigned. The chan argument is the address of

a word into which $ASSIGN writes the channel number.

$SETPRN allows a process to establish or change its own process name. Format:

SYS$SETPRN ([prcnam])

prcnam (mechanism: by descriptor--fixed length string descriptor)

Process name to be given to the calling process. The prcnam argument is the address of a character string descriptor pointing to a 1- to 15-character process name string. If prcnam is not specified, the calling process is given no name.

LIB$GET_INPUT gets one record of ASCII text from the current controlling input device, specified by SYS$INPUT. Format:

LIB$GET_INPUT (get-str [,prompt-str] [,out-len])

get-str (mechanism: by descriptor)

String which LIB$GET_INPUT gets from the input device. The get-str argument is the address of a descriptor pointing to the character string into which LIB$GET_INPUT writes the text received from the current input device.

prompt-str (mechanism: by descriptor)

Prompt message that is displayed on the controlling terminal. The prompt-str argument is the address of a descriptor containing the prompt.A valid prompt consists of text followed by a colon (:), a space, and no carriage return /line feed combination. The maximum size of the prompt message is 255 characters. If the controlling input device is not a terminal, this argument is ignored.

out-len (mechanism: by reference)

Number of bytes written into get-str by LIB$GET_INPUT, not counting padding in the case of a fixed string. The out-len argument is the address of an unsigned word containing this number. If the input string is truncated to the size specified in the get-str descriptor, out-len is set to this size. Therefore, out-len can always be used by the calling program to access a valid substring of get-str.

SMG$CREATE_VIRTUAL_DISPLAY creates a virtual display and returns its assigned display id. Format:

SMG$CREATE_VIRTUAL_DISPLAY (num-rows, num-columns, display-id,,,)

num-rows (mechanism: by reference)

Specifies the number of rows in the newly created virtual display. The num-rows argument is the address of a signed longword integer that contains the desired number of rows.

num-columns (mechanism: by reference)

Specifies the number of columns in the newly created virtual display. The num-columns argument is the address of a signed longword integer that contains the desired number of columns.

display-id (mechanism: by reference)

Receives the display-id of the newly created virtual display. The display-id argument is the address of an unsigned longword into which is written the display identifier.

SMG$SET_DISPLAY_SCROLL_REGION creates a scrolling region in a virtual display. Format:

SMG$SET_DISPLAY_SCROLL_REGION (display-id,,)

SMG$SCROLL_DISPLAY_AREA scrolls a rectangular region of a virtual display. Format:

SMG$SCROLL_DISPLAY_AREA (display-id,,)

SMG$LABEL_BORDER supplies a label for a virtual display's border. Format:

SMG$LABEL_BORDER (display-id ,text,,,,,)

text: (mechanism: by descriptor)

String into which the input line is written. The text argument is the address of a descriptor pointing to the storage into which the text is written.

SMG$CREATE_VIRTUAL_KEYBOARD creates a virtual keyboard and returns its assigned keyboard-id. Format:

SMG$CREATE_VIRTUAL_KEYBOARD (new-keyboard-id,,,)

new-keyboard-id (mechanism: by reference)

Receives the keyboard identifier of the newly created virtual keyboard. The new-keyboard-id argument is the address of an unsigned longword into which is written the keyboard identifier.

SMG$CREATE_PASTEBOARD creates a pasteboard and returns its assigned pasteboard-id. Format:

SMG$CREATE_PASTEBOARD (pasteboard-id,,,)

pasteboard-id (mechanism: by reference)

Receives the identifier of the newly created pasteboard. The pasteboard-id

argument is the address of an unsigned longword into which is written the new pasteboard identifier.

SMG$PASTE_VIRTUAL_DISPLAY pastes a virtual display to a pasteboard. Format:

```
SMG$PASTE_VIRTUAL_DISPLAY (display-id ,pasteboard-id
                           ,pasteboard-row ,pasteboard-column)
```

display-id, pasteboard-id: Same as above.

pasteboard-row (mechanism: by reference)

Specifies the row of the pasteboard that is to contain row 1 of the specified virtual display. The pasteboard-row argument is the address of a signed longword integer that contains the row number.

pasteboard-column (mechanism: by reference)

Specifies the column of the pasteboard that is to contain column 1 of the specified virtual display. The pasteboard-column argument is the address of a signed longword integer that contains the column number.

SMG$ERASE_DISPLAY erases all or part of a virtual display by replacing text characters with blanks. Format:

```
SMG$ERASE_DISPLAY (display-id)
```

SMG$READ_STRING reads a string from a virtual keyboard. Format:

```
SMG$READ_STRING (keyboard-id ,received-text [,prompt-string]
                 [,,,,] [,received-string-length]
                 [,][,display-id] )
```

keyboard-id (mechanism: by reference)

Specifies the virtual keyboard from which input is to be read. The keyboard-id argument is the address of an unsigned longword that contains the keyboard identifier.

received-text (mechanism: by descriptor)

String into which the input line is written. The received-text argument is the address of a descriptor pointing to the storage into which the text is written.

prompt-string (mechanism: by descriptor)

String used to prompt for the read operation. The prompt argument is the address of a descriptor pointing to the prompt string.

received-string-length (mechanism: by reference)

Receives the number of characters read or the maximum size of received-text, whichever is less. The received-string-length argument is the address of an unsigned word into which is written the number of characters or the maximum size.

display-id: Same as above.

SMG$PUT_LINE writes a line of text to a virtual display. Format:

```
SMG$PUT_LINE (display-id ,text ,,rendition-set,0,,)
```

display-id, text: Same as above.

rendition-set (mechanism: by reference)

A mask that denotes video attribute fo the draw line. The rendition-set argument is the address of an unsigned longword that contains an attribute mask. Each bit attribute in this argument causes the corresponding attribute to be set in the display. The possible attributes are:

SMG$M_BLINK	Displays characters blinking.
SMG$M_BOLD	Displays characters in higher-than-normal intensity (bolded).
SMG$M_REVERSE	Displays characters in reverse video --- that is, using the opposite default rendition of the virtual display.
SMG$M_UNDERLINE	Displays characters underlined.

STR$UPCASE converts a string to uppercase and writes the converted string into the destination string. When you need to compare characters without regard to case, you can first use STR$UPCASE to convert both characters to uppercase. STR$UPCASE converts all characters in the multinational character set. Format:

```
STR$UPCASE dst-str ,src-str
```

dst-str (mechanism: by descriptor)

Destination string into which STR$UPCASE writes the string it has converted to uppercase. The dst-str argument is the address of a descriptor pointing to the destination string.

src-str (mechanism: by descriptor)

Source string that STR$UPCASE converts to uppercase. The src-str argument is the address of a descriptor pointing to the source string.

LIB$SYS_GETMSG calls the System Service $GETMSG and returns a message string into dst-str using the semantics of the caller's string. Format:

LIB$SYS_GETMSG (msg-id ,[msg-len] ,dst-str [,flags,])

msg-id (mechanism: by reference)

Message identification to be retrieved by LIB$SYS_GETMSG. The msg-id argument contains the address of a signed longword integer that is this message identification.

msg-len (mechanism: by reference)

Number of characters written into dst-str, not counting padding in the case of a fixed-length string. The msg-len argument contains the address of a signed word integer that is this number.

dst-str (mechanism: by descriptor)

Destination string. The dst-str argument contains the address of a descriptor pointing to this destination string. LIB$SYS_GETMSG writes the message that has been returned by $GETMSG into dst-str.

flags (mechanism: by reference)

Four flag bits for message content. The flags argument is the address of an unsigned longword that contains these flag bits. The default value is a longword with bits zero through 3 set to 1. The flags argument is passed to LIB$SYS_GETMSG by reference and changed to value for use by $GETMSG.

LIB$GET_EF allocates one local event flag from a process-wide pool and returns the number of the allocated flag to the caller. If no flags are available, LIB$GET_EF returns an error as its function value. Format:

LIB$GET_EF event-flag-num

event-flag-num (mechanism: by reference)

Number of the local event flag that LIB$GET_EF allocated, or --1 if no local event flag was available. The event-flag-num argument is the address of a signed longword integer into which LIB$GET_EF writes the number of the local event flag that it allocates.

$SETAST enables or disables the delivery of ASTs for the access mode from which the service call was issued. Format:

SYS$SETAST enbflg

enbflg (mechanism: by value)

Value specifying whether ASTs are to be enabled. The enbflg argument is a byte containing this value. A value of 1 enables AST delivery for the calling access mode; a value of 0 disables AST delivery.

System_Services

$READEF returns the current status of all 32 event flags in a local or common event flag cluster. In addition, the condition value returned indicates whether the specified event flag is set or clear. Format:

SYS$READEF efn ,state

efn (mechanism: by value)

Number of any event flag in the cluster whose status is to be returned. The efn argument is a longword containing this number. Specifying an event flag within a cluster requests that $READEF return the status of all event flags in that cluster.

state (mechanism: by reference)

State of all event flags in the specified cluster. The state argument is the address of a longword into which $READEF writes the state (set or clear) of the 32 event flags in the cluster.

$SETIMR sets the timer to expire at a specified time. When the timer expires, an event flag is set and (optionally) an AST routine executes. Format:

SYS$SETIMR [efn] ,daytim ,[astadr] ,[reqidt]

efn (mechanism: by value)

Event flag to be set when the timer expires. The efn argument is a longword value containing the number of the event flag. If efn is not specified, event flag 0 is set.

daytim (mechanism: by reference)

Time at which the timer expires. The daytim argument is the address of a quadword time value. A positive time value specifies an absolute time at which the timer expires; a negative time value specifies an offset (delta time) from the current time.

astadr (mechanism: by reference)

AST service routine that is to execute when the timer expires. The astadr is the address of the entry mask of this routine. If astadr is not specified or is specified as 0 (the default), no AST routine executes.

reqidt (mechanism: by value)

Identification of the timer request. The reqidt is a longword value containing a number that uniquely identifies the timer request. If reqidt is not specified, the value 0 is used.

$ASCTIM converts an absolute or delta time from 64-bit system time format to an ASCII string. Format:

SYS$ASCTIM (,timbuf,,)

timbuf (mechanism: by descriptor--fixed length string descriptor)

Buffer into which $ASCTIM writes the ASCII string. The timbuf argument is the address of a character string descriptor pointing to the buffer.

$CLREF clears (sets to 0) an event flag in a local or common event flag cluster. Format:

SYS$CLREF efn

$CREPRC creates a subprocess or detached process on behalf of the calling process. Format:

```
SYS$CREPRC  ([pidadr],[image],,[output],[error]
            ,,,[prcnam],[baspri] ,,,[stsflg])
```

pidadr (mechanism: by reference)

Process identification (PID) of the newly created process. The pidadr argument is the address of a longword into which $CREPRC writes the PID.

image (mechanism: by descriptor--fixed length string descriptor)

Name of the image to be activated in the newly created process. The image argument is the address of a character string descriptor pointing to the file specification of the image.

output (mechanism: by descriptor--fixed length string descriptor)

Equivalence name to be associated with the logical name SYS$OUTPUT in the logical name table of the created process. The output argument is the address of a character string descriptor pointing to the equivalence name string.

error (mechanism: by descriptor--fixed length string descriptor)

Equivalence name to be associated with the logical name SYS$ERROR in the logical name table of the created process. The error argument is the address of a character string descriptor pointing to the equivalence name string.

prcnam (mechanism: by descriptor--fixed length string descriptor)

Process name to be assigned to the created process. The prcnam is the address of a character string descriptor pointing to a 1- to 15-character process name string.

baspri (mechanism: by value)

Base priority to be assigned to the created process. The baspri argument is a longword value in the range 0 to 31 (Use 3).

stsflg (mechanism: by value)

Options selected for the created process. The stsflg argument is a longword bit vector wherein a bit corresponds to an option. Only bits 0 to 10 are used; bits 11 to 31 are reserved and must be 0.

$TRNLNM returns information about a logical name. Format:

```
SYS$TRNLNM  (,tabnam,lognam,,itmlst)
```

tabnam (mechanism: by descriptor--fixed length string descriptor)

Name of the table or name of a list of table names in which to search for the logical name. The tabnam argument is the address of a descriptor pointing to this name. This argument is required.

lognam (mechanism: by descriptor--fixed length string descriptor)

Logical name about which information is to be returned. The lognam argument is the address of a descriptor pointing to the logical name string.

itmlst (mechanism: by reference)

Item list describing the information that $TRNLNM is to return. The itmlst argument is the address of a list of item descriptors, each of which specifies or controls an item of information to be returned. The list of item descriptors is terminated by a longword of 0.