

DATE: 9-7-84
FROM: Tim
TO: All
SUBJECT: Calling Modules

I have designed a simple scheme to allow modules to call other modules in any overlay to the depth of the return stack.

THE SCHEME: The general idea is to save the pfa of the current overlay caller on the return stack, execute the called module, get the previous overlay pfa from the return stack and restore the overlay. Control then passes to word following the module call.

EXAMPLE MODULE CALL: The following example shows how the functions in starport can be made into modules that can be called from any overlay.

```
: STARPORT
BEGIN
  GETROOMCHOICE ( query user for choice of room)
  ROOMCHOICE @ 0 = IF 'SHIPCONFIG @ ELSE
  ROOMCHOICE @ 1 = IF 'TRADEDEPOT @ ELSE
  ROOMCHOICE @ 2 = IF 'CREWASSIGN @ ELSE
  ROOMCHOICE @ 3 = IF 'TRAINCHAR @ ELSE
  THEN THEN THEN THEN
  MODULE
AGAIN ;
```

WHERE: 'SHIPCONFIG is a variable that contains the pfa of the SHIPCONFIG module caller.

'STARPORT-OV is a variable that contains the pfa of the STARPORT overlay loader.

```
: MODULE ( module-caller-pfa -- \ call a module)
  CURRENT-OV @ >R ( save current overlay loader pfa)
  EXECUTE ( call the module)
  R> CALL-OV ; ( restore the overlay)

: CALL-OV ( overlay-loader-pfa -- \ load an overlay)
  DUP CURRENT-OV ! ( set this as the current overlay)
  EXECUTE ; ( load the overlay)

: SHIPCONFIG ( -- \ module caller for shipconfig)
  'STARPORT-OV @ CALL-OV ( get overlay loader & load)
  STARPORT ( set vocabulary to overlay)
  DOSHIPCONFIG ; ( execute word in vocab)

' SHIPCONFIG 'SHIPCONFIG ! ( set vector)
' STARPORT-OV 'STARPORT-OV ! ( set vector)
```

DESIGN CONSTRAINTS: If you call modules in other overlays you should be aware of the following:

1. Since overlays are read-only, any variables within the current overlay will be reset upon return from a called module in another overlay. To prepare for this, load any variable that you want to preserve onto the stack, call the module, and restore the variables.
2. Each module and overlay you define must have a corresponding vector variable which you must load. This may be done at compile time or, in the case of overlays that vary from the PC to the PCjr, at run time.
3. For maximum flexibility, module callers and overlay loaders should reside in the kernel. This, however, takes space in the kernel which reduces the maximum size of all overlays.
4. The word MODULE must reside in the kernel.
5. The word CURRENT-OV must reside in the kernel.
6. The return stack, parameter stack and processor state ^{AND} must be saved when the game is saved and restored ^{DISPATCH IMAGE} when the game is resumed.

ADDITIONAL PROPOSAL:

1. The SAVEGAME module should be called in response to an 'ESC' key press in any module. The message, "Press 'S' to save the game (any other key to continue)." is then presented to the user his choice. If 'S' is pressed, save the game, else end module.

* Precede instance constant addresses.