Inter-Office Memorandum

To	Mesa Users	Date	October 27, 1980
From	Jim Sandman, John Wick	Location	Palo Alto
Subject	**Integrated Mesa Environment**	Organization	SDD/SS/Mesa

XEROX

Filed on: [Iris]<Mesa>Doc>CommandCentral.bravo (and .press)

This memo documents a small executive called Command Central; this Tool is intended to be installed with the Debugger and can be used to invoke the Compiler, the Binder, and client programs, all of which upon completion are directed to return to Command Central rather than to the Alto Executive. The idea is that, while programming in Mesa, you enter Command Central's control only once, and you rarely have to leave it; this is made possible by the editor that is now included in the Debugger, as well as by the context switching facilities provided by Command Central.

Installation

To include Command Central in the Debugger, type the following Alto Executive command when installing, after retrieving `<Mesa>Fetch.bcd` (which contains **TinyPup**, **Stps**, and the **FileTool**) and `<Mesa>Utilities>CommandCentral.bcd`. (If you have more than 64K of memory, be sure to consult the Installation section of the Debugger documentation before proceeding.)

```
>XDebug Fetch/l CommandCentral/l
```

While it is possible to use Command Central without also installing the **FileTool**, including it will help minimize the number of times you have to leave the Mesa environment. If you have enough memory on your machine, you might consider installing other Tools with your Debugger as well (e.g., **ChatTool**, **SendMessageTool**).

Note: Tools loaded via the command line are initially inactive (i.e., no window is showing); move the cursor into the gray area outside all windows and use the menu found there to activate them.

Entering Command Central

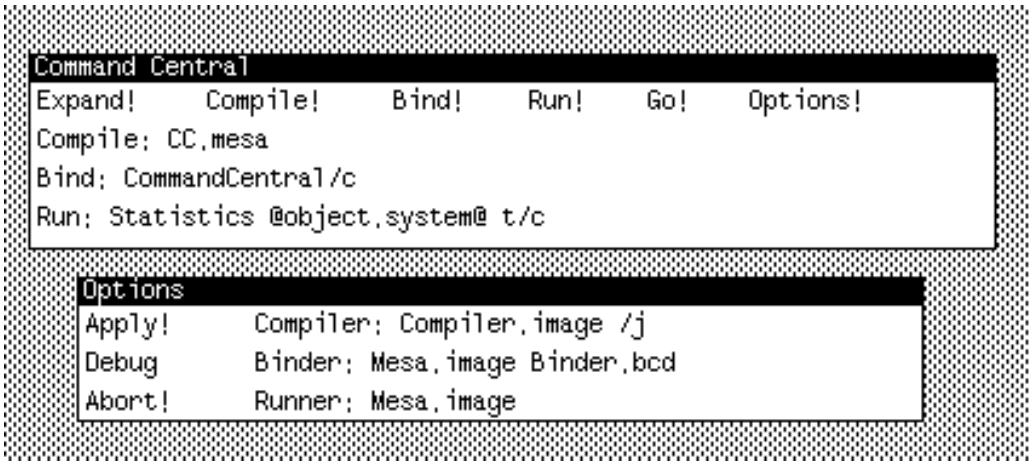
When using Command Central, the Debugger becomes the executive from which all programs are invoked. To first enter this environment, type

```
>Mesa/d
```

to the Alto Executive. You can now use the **FileTool** to retrieve the modules you wish to work on and the Tools editor to modify them. When you have finished your changes, turn your attention to the Command Central window.

Command Central Window

This window provides command lines for compiling, binding, and running your program, the context switching commands, and an option sheet; it also supports the standard window operations (scrolling, growing, etc.).



The three fields contain command lines for the Compiler, the Binder, and the System; their contents are written to `Com.cm` when the commands are invoked. (A special global switch /q is added so that control is returned to the Debugger rather than to the Alto Executive.)

The `Compile!`, `Bind!`, and `Run!` commands invoke the appropriate programs using the command lines constucted from the `Compile:`, `Bind:`, and `Run:` fields, respectively. The `Run!` command can be used to invoke `.bcd`, `.image`, and `.run` files (see below). The `Go!` command constructs a combined command line using all non-null fields and executes the appropriate programs in order. Note that the Debugger will complain if you issue any of these commands while a file is being edited (since the edits might be lost as a result of executing them).

The parameter fields also recognize command files preceeded by the traditional at-sign (e.g., `@file.cm`); the `Expand!` command will expand all such references into their contents and write the result back into the window. Indirect references are also automatically expanded when any of the other commands are invoked.

The `Options!` command produces the options window which allows you to specify the names of the compiler, binder and system you are using. The names are parsed to allow default switches to be included. The `Apply!` command will save the new names and the `Abort!` command will restore the names to their previous state (the default names are shown above). Both `Apply!` and `Abort!` will remove the options window.

If the Compiler or the Binder detect errors (and the pause switch is in effect), they will invoke the Debugger with an appropriate message instead of pausing. You can then load the appropriate error log into a window and step through it and your source file together. Because the file index of the error is included in each message, the `position` menu command can be used to find the source of the error quickly.

Invoking Other Programs

Any Mesa `.bcd` which expects to be loaded into `Mesa.image` and obtains its commands from the command line (`Com.cm`) can be invoked by Command Central using the `Run:` field and the `Run!` command. (As above, the global /q switch is added to the command line so that control will return to the Debugger.) Some obvious programs which you might include on your disk are **Access** and **Print**.

You can also run arbitrary `.image` and `.run` files using Command Central, but unless they have made provision to return control to the Debugger, they will exit to the Alto Executive upon completion. Use the `Mesa/d` command to reenter Command Central.

Limitations

If you use the `Compile!`, `Bind!`, `Run!`, or `Go!` commands when you are in the middle of a debugging session (at a breakpoint or an uncaught signal, for example), the state of the client will be lost. In particular, normal termination processing of the client will not take place (e.g., open files will be left dangling).

Distribution:
- Mesa Users
- Mesa Group
- SDSupport