Inter-Office Memorandum

To	Mesa Users	Date	October 23, 1980
From	Dick Sweet	Location	Palo Alto
Subject	**Mesa 6.0f Change Summary**	Organization	SDD/SS/Mesa

XEROX

Filed on: [Igor]<AlphaMesa>Doc>Summary60f.bravo

This memo outlines changes made in Mesa since the last alpha update (Mesa 6.0m, September 26, 1980). These changes will shortly be incorporated into the Mesa 6.0 Change Summary to correctly reflect the differences between Mesa 5.0 and Mesa 6.0. This document also includes a list of ARs fixed since 6.0m. Since these versions of the programs are the final release ones, they say 6.0 instead of 6.0f.

Language

The language has been extended slightly.

Default Return Values

In Mesa 6, one can specify a default value for a type or a field of a record. Version 6.0f has some small changes in the interaction of defaults and procedure declarations. The most notable is the ability to specify defaults on *result records* as well as parameter records. The other difference is that default values in parameter (result) records are not inherited from types with default values. This is best explained by considering some examples, followed by some rules that help explain them.

```
. . .
Ptr: TYPE = POINTER _ NIL;
Proc: PROCEDURE RETURNS [a: BOOLEAN _ TRUE, b: CARDINAL _ 0, c: Ptr] =
    BEGIN
    -- a initialized to TRUE, b initialized to 0, c initialized to NIL
    . . .
    RETURN[a: FALSE, c: exp];  -- returns [FALSE, 0, exp];.
    . . .
    a _ FALSE;
    RETURN[c: exp];          -- returns [TRUE, 0, exp];.
    . . .
    RETURN[a:, b: ];         -- illegal, c has no default (even though Ptr has a default)
    . . .
    RETURN[a:, b:, c: ];     -- also illegal.  Although c has no default, its need for a
                             --  non-NULL value, inherited from type Ptr, is retained
    . . .
    RETURN;                  -- returns current values of a, b, and c.
    END;
```

Now for some rules:

> There are two uses for default return values: to initialize values of local variables, and as defaults in constructors for result records. The first use is probably the most valuable.
>
> If the type of a field in the result record has a default, the default applies for the initialization of the corresponding local variable, but does not apply in a constructor for the result record. This protects you from assigning a value to a return variable and then forgetting to mention it in a RETURN statement, causing the default for its type to be returned.
>
> If the type of a field has a default, and it is not of the form "_ *exp* | NULL," then the field can be thought of as having a default of the form "_". That is, some non-NULL value must be given in all constructors for that field.
>
> The same non-inheritence of type defaults applies for parameter records as well as for result records.
>
> In Mesa 6, you cannot use the values of the parameters in defaults for return values; you must use constants or expressions involving variables global to the procedure. This restriction will probably be lifted in future releases.

Binder

There have been several changes

Symbol Compressor Reappears

The symbol compressor is available again. It can be found on the `>Binder` subdirectory of the release directory as `SymbolCompressor.bcd`. To use it, load it ahead of the Biinder and specify the `/x` switch. Thus

```
>Mesa.image SymbolCompressor Binder Foo/x
```

will create `Foo.bcd` and `Foo.symbols` (compressed). Compressed symbols are more compressed than they were in Mesa 5. They contain just the (public and private) procedures and signals declared at the top level of the module.

New Global Switches

The switches to copy code (`/c`), to copy symbols (`/s`), and to compress symbols (`/x`) may now be given as global switches, and hence apply to all source files thereafter.

Copying (*some*) *Symbols*

In earlier versions, copying symbols was an error if not all of the symbol files were available at bind time. The binder now copies all symbols that it can find, leaves the symbol table references for the other modules in the original (unavailable) files, and issues a warning.

Flashier Cursors

The cursor now changes as soon as warnings or errors are logged by the binder, in case you aren't suspecting them and want to stop things and look around.

Better Error Messages

The new Binder tries harder to give you the name of missing interface items.

System

The file `Mesa.symbols` is again available as compressed symbols for the file `Mesa.image`.

Debugger

In addition to many bug fixes, the following changes and additions have been made:

Sequence Indexing

The Debugger now knows how to index sequences. Thus, if *seq* is a pointer to a record containing a sequence part, say *items*, the interpreter commands

seq[*i*]

and

seq.items[*i*]

are equivalent, and display the *i*th element of the sequence.

Break Exit Source

When you set an exit break on a procedure, you take the break on any return (they jump to a common instruction to make this easy on the debugger). The debugger has no way of telling you which return was taken if there is more than one. When asked for source, the debugger now shows the first line of the procedure declaration instead of the last return.

Debugger Looks for Symbols

Even if a module has compressed symbols, the debugger will first look around for the file `modulename.bcd` and see if it is the original compiler output for that module (by checking the version stamp). If so, it will use those symbols. Thus, there is no need to `Attach Symbols` if the proper file is on the disk. It makes sense to compress symbols for large systems and have around the complete symbols for specific modules undergoing detailed debugging.

Representation of Literals in Interpreter

The debugger does a better job of determining signed/unsigned representation of values given to the interpreter. For example, any octal number is assumed to be unsigned.

Fetch Tool

Instead of the File Tool, one should use `Fetch.bcd`, available on the release directory, to have a file transfer facility within `XDebug`. The necessary Pup machinery is part of `Fetch.bcd`.

Editor Changes

The command for moving the insert point in the editor has been changed to `Control-Red` (from `Shift-Red`) partly to benefit the sanity of Laurel 6 modeless editor users. The debugger will now complain if you try to `Kill session` when you are editing a file (you can, however, `Proceed`).

Lister

The command scanner for the lister now allows optional quotes on string parameters. Anything up to the next comma or right bracket is considered the parameter. Of course you can still type them, and must if the parameter contains a comma or right bracket (I can't think of any lister commands where this makes sense, though). In addition, there are several new lister commands of general interest.

`Implementors["FileName"]`

> `FileName` is a list of BCDs, separated by spaces (such as `object.system`). The lister produces a file *root*`[FileName].iml` (e.g. `object.iml`) that contains a listing of exported interfaces, together with the name of the program module implementing each interface item. If the given interface is also named in the input file, the output will also include those interface items not implemented by any programs in the collection. In order to run this command, you have to have symbols for all of the exported interfaces as well as all of the items named in the input file.

`Stamps["FileName"]`

> `FileName` is a Compiler or Binder or Packager output BCD. The lister produces the file `Filename.bl` that contains the version stamp of any modules bound in the BCD, and the version stamps for all imported and exported interfaces. This command should replace most uses of the `Bcd["FileName"]` command and produces output much easier to read.

`UnboundExports["FileName"]`

> `FileName` is a Compiler or Binder or Packager output BCD. The lister produces the file `FileName.xl` that shows, for each exported interface, which items are not implemented. In order to run this command, you need the symbols for any exported interface that is not completely implemented. The lister will complain about missing ones.

`Version["FileName"]`

> `FileName` is a Compiler or Binder or Packager output BCD, or an image file. The lister shows, on `Mesa.typescript`, the version stamps for the file, its creator, and its source.

Command Central

The debugger will now complain if you try to exit the debugger (`compile!`, `bind!`, `run!`, or `go!`) while editing a file.

Distribution:
Mesa Users
Mesa Group
SDSupport

Number	Originator	Subsystem	Subject
305	Ayers	Binder	Binder/cx failed when included was missing symbols
3615	Hamilton	Binder	Abolish or explain .code
4326	Karlton	Binder	Procedure instead of Module in IMPORTS list bombs
4671	mbrown	Binder	Doc: new semantics of "foo/c foo/s"
4925	Newman	Binder	Did not look up unbound procedures in Defs on disk
4994	rovner	Binder	Doc: Change in the meaning of /s
5283	Newman	Binder	Doc: "old" vs "new" command lines
6038	Wyatt	Binder	Illegal command line syntax causes uncaught InvalidObject
6067	Levin	Binder	Spurious warning messages for required codelinks
1248	Sheil	Compiler	Compiler Confuses Circularity
3713	Satterthwaite	Compiler	Procedures declared within discriminations
5033	Newman	Compiler	Semantics of "Foo: TYPE = CARDINAL _ ;" declaration
5288	Newman	Compiler	Allowed invalid initialization expr for CONDITION
5703	Atkinson	Compiler	ERROR bugs
5851	Inouye	Compiler	Fatal Compiler Error (another)
5985	mbrown	Compiler	examples of poor code produced
6020	Levin	Compiler	Generated code loses long return record in IF clause
6028	AWells	Compiler	Compiler dies in Pass 5 sending me to debugger with no displ
6043	Kiser	Compiler	Bad code for adding cardinal to long relative pointer.
6058	Newman	Compiler	Type/Length confusion on RELATIVE LONG POINTERs
6172	Newman	Compiler	ANY doesn’t catch unnamed ERROR in same frame
6173	Newman	Compiler	ENABLE UNWIND fails to execute when unwinding unnamed ERROR
6186	Swinehart	Compiler	Fatal Pass 4 bug, using strange var. rec. arg.
6229	israel	Compiler	Compiler dies on RETURN [] in pass 4
2013	Murray	Debugger	Coremap After BootFrom Doesn’t Help Much
2111	Levin	Debugger	Front.run Produced Files
3401	Murray	Debugger	Lost breakpoints
4531	McJones	Debugger	Printing "msg=..." on parameterless uncaught signal
4796	Levin	Debugger	Bootloading vs. memory configuration
4872	Malasky	Debugger	Prohibit conditional breaks on pointers and arrays
4884	Cattell	Debugger	What to do about edited windows when exit debugger
4983	Newman	Debugger	Set process context to self => "nnn is not a valid frame!"
5026	Newman	Debugger	Sometimes you don’t get "nnnnnn is not started!" message
5110	morris	Debugger	Debugger printed out a wrong variant
5222	Murray	Debugger	CoreMap: SerialNumbers: print out high half also
5340	JEllis	Debugger	Can’t print a module variable w/o setting Module context
5386	Murray	Debugger	SIGNAL declared in a catch phrase
5470	Daniels	Debugger	Proc[]^ gives ERROR
5750	Cattell	Debugger	ATtach Condition goes into infinite loop in several cases
5926	kolling	Debugger	Doc: What is "wory" mode?
5960	Levin	Debugger	Attach Image or Loadstate of Run file fails
5962	Levin	Debugger	Can’t carry Swatee to another machine and bootload
5965	Newman	Debugger	Doc: Interpreter can print out a TYPE
6016	Levin	Debugger	Reading code after bootloading
6017	Levin	Debugger	Debugger shouldn’t read machine number from client
6061	mbrown	Debugger	Debugger displays imported variables incorrectly
6062		Debugger	Certain TYPEs print out as "PUBLIC TYPE = <blank space>"
6069	Sweet	Debugger	Base RELATIVE LONG POINTER TO Foo
6071	McJones	Debugger	Incorrect "! Incorrect tag " message
6241	israel	Debugger	Degugger doesn’t understand relativa array descriptors.
6242	israel	Debugger	Can’t FTP debug log
6269	Malasky	Debugger	printing interval of array[0..0)
4767	Cattell	Ether	FileTool: filename "." doesn’t work
5132	MBrown	Ether	FileTool: does not set creation date properly
5207	Birrell	Ether	FileTool: retrieve NIL file causes StringBoundsFault
5214	Hamilton	Ether	FileTool: Clear message window
5266	Hamilton	Ether	ChatTool: needs to Yield
5433	Lauer	Ether	CoPilot leaves me ’logged in’ in new session
5939	Murray	Ether	Pup: EFTPAbortReceiving sends extra characters
5942	Murray	Ether	FTP: Can’t retrieve mail from connected directory.
6010	Malasky	Ether	Brownie: Enumerate with !h or ;0
6011	Malasky	Ether	Brownie: Validate passwords immediately
6012	Malasky	Ether	Brownie: Rename update in addition to rename everything
6013	Malasky	Ether	Brownie: Use PupChecksums microcode
6076	Murray	Ether	Pup+FTP: a few constructors smashing LOCKs or CONDITIONs
6108	Birrell.pa	Ether	FTP "no" code of 0: ProtocolError vs. UnidentifiedError
6110	Birrell.pa	Ether	BSP "interrupt" sending doesn’t handle lost packets
1994	Murray	Other	RunMesa: Addition To XMesa Microcode