Inter-Office Memorandum

To	Distribution	Date	October 1, 1979
From	Roy Levin	Location	Palo Alto
Subject	Mesa for Extended Memory Altos (version 5)	Organization	PARC/CSL

XEROX

Filed on: [Ivy]<XMesa>Doc>XMesa.bravo (and .press)

XMesa is a version of the Mesa runtime system that can exploit in certain ways the extended memory option of the Alto II. This document describes the facilities provided by this system, its limitations, and its compatibility with other current Mesa systems.

Overview of XMesa

XMesa uses the extended memory of an Alto II XM as additional swapping space for code. This means that code and data need not co-exist in the primary 64K of memory. XMesa takes advantage of any available extra space automatically; standard Alto programs do not need to be modified to run with XMesa. XMesa does not simplify the task of managing data structures that are too large to fit in 64K, but it does free (most of) the 64K primary address space for use by data.

Because XMesa uses extra memory for code segments, it includes a page-level storage allocator for the additional banks. Client programs may request storage in the additional banks by using this allocator; the interface is public. XMesa provides primitive mechanisms to read and write words in extended memory and to copy blocks of data between banks of memory, but gives no other assistance in accessing information in the extended memory. In particular, arbitrary use of **LONG POINTER**s is *not* supported.

Clients should understand that, while XMesa is interface-compatible with Mesa 5.0 the implementation of certain system components (e.g., the page-level memory allocator and segment swapper) has changed substantially. Client programs that assume undocumented properties of these components do so at their peril. Information about incompatibilities in the implementation appears later in this document.

Compatibility with Mesa 5.0

Considerable efforts have been made to ensure that XMesa is compatible with Mesa 5.0. Exceptions are minor and typically affect only those programs that depend upon the format of global frames (details appear below). To enhance the value of compatibility and because wide-bodied Altos are not universally available, XMesa will execute properly (though with some performance degradation) on a normal Alto (I or II) without recompilation or rebinding. Indeed, XMesa requires no special compiler or binder. Both BCDs and image files are compatible across machines. Thus a Mesa disk containing the XMesa versions of `XMesa.image` (or `BasicXMesa.image`), and the debugger can be used without alteration on any Alto, XM or otherwise. Specifically,

1) Mesa 5.0 BCDs will load and execute under XMesa,
2) Mesa 5.0 image files will run correctly under XMesa,
3) Image files created with XMesa will run (under XMesa) on any Alto,
4) Files created with Mesa 5.0 and XMesa may be debugged (under XMesa) on any Alto.

Of course, these properties can be guaranteed only for programs that do not assume the existence of extended memory.

Note! Compatibility is assured only for Mesa 5.0. BCDs compiled or bound by previous releases of the Mesa system will NOT work properly under XMesa.

The Public Interface to XMesa

[Previous releases of XMesa provided their services exclusively through the **XMesaDefs** interface. In Mesa 5.0, some of the facilities previously available only in XMesa have been incorporated into standard system interfaces. To aid users of XMesa 4.1 in converting to XMesa 5.0, these facilities are still documented here. In some cases, however, the interfaces have changed slightly.]

XMesa provides most of its services through the **XMesaDefs** interface. However, the facilities supplied logically belong in other existing interfaces, notably **SegmentDefs** and **AltoDefs**. Thus, **XMesaDefs** actually includes several semi-independent interfaces. Unless otherwise stated, everything defined in this section comes from **XMesaDefs**.

Configuration Information

The facilities described in this section are defined in the **MemoryOps** interface, and therefore are available in both standard Mesa 5.0 and XMesa.

The Mesa runtime system has an internal data structure that contains information about the hardware configuration of the Alto on which it is running. Clients may obtain a copy of this data structure by writing

```
memoryConfig: MemoryConfig _ GetMemoryConfig[];
```

and should normally test for the existence of extended memory by examining **memoryConfig.useXM**. The extant banks of memory are indicated by **memoryConfig.banks**, which is a bit mask (*e.g.*, **memoryConfig.banks=14B** => banks 0 and 1 exist).

```
BankIndex: TYPE = [0..3];

MachineType: TYPE = {unknown0, AltoI, AltoII, AltoIIXM, . . . };

MemoryConfig: TYPE = MACHINE DEPENDENT RECORD[
    reserved: [0..37B],
    AltoType: MachineType,
    useXM: BOOLEAN,
    mdsBank: BankIndex,
    secondROM: BOOLEAN,
    banks: [0..17B],
    mesaMicrocodeVersion: [0..377B],
    XMMicrocodeVersion: [0..377B]];
```

GetMemoryConfig: **PROCEDURE RETURNS**[**MemoryConfig**];

useXM is true if and only if the following conditions hold:

1) the machine is an Alto II with XM modifications (**AltoType = AltoIIXM**),
2) the Alto has more than one memory bank installed (**banks ~= 10B**),
3) the Alto has a second ROM (**secondROM** is true), and
4) the second ROM contains the current version of the XMesa microcode.

The microcode version fields tell only the *microcode* version, *not* the Mesa release number. (For example, for Mesa 5.0, **mesaMicrocodeVersion** is 39.) **XMMicrocodeVersion** will be non-zero if and only if XMesa microcode is installed in the second ROM.

The procedure **IsXMesa** in **MemoryOps** provides a convenient way for a client BCD to determine if it has been loaded into an XMesa runtime system. The return value **TRUE** from **IsXMesa** does *not* guarantee that XMesa is running on a wide-bodied Alto; the client should use **GetMemoryConfig** to verify the existence of extended memory (see below).

IsXMesa: **PROCEDURE RETURNS [BOOLEAN];**

Unless otherwise noted, the facilities described in subsequent sections are usable only when the following expression, henceforth called "the XM condition", is **TRUE**:

IsXMesa[] AND GetMemoryConfig[].useXM

Extended Memory Management

The facilities described in this section are defined in the **XMesaDefs** interface and should be used only when **MemoryOps.IsXMesa[]** is **TRUE**.

Segments in extended memory are created with the usual primitives in **SegmentDefs**. However, XMesa recognizes additional "default" parameter values for those procedures that expect a VM base page number. **DefaultBase0**, **DefaultBase1**, **DefaultBase2**, and **DefaultBase3** request allocation in a specific memory bank. **DefaultXMBase** requests allocation anywhere in the extended memory banks (banks 1-3, except for Cedar). **DefaultBase** (defined in **SegmentDefs**) requests allocation anywhere in memory (banks 0-3) *if the segment is a code segment*, otherwise, it is equivalent to **DefaultBase0**. **DefaultMDSBase** is equivalent (and preferred to) **DefaultBase0** (except for Cedar).

The following procedures logically belong in **SegmentDefs**. They convert between segment handles and long pointers, and work for segments anywhere in the 18-bit address space.

XVMtoSegment: **PROCEDURE** [**a**: **LONG POINTER**] **RETURNS** [**SegmentHandle**];

XSegmentAddress: **PROCEDURE** [**seg**: **SegmentHandle**] **RETURNS** [**LONG POINTER**];

XVMtoDataSegment: **PROCEDURE** [**a**: **LONG POINTER**] **RETURNS** [**DataSegmentHandle**];

```
XDataSegmentAddress: PROCEDURE [seg: DataSegmentHandle]
    RETURNS [LONG POINTER];

XVMtoFileSegment: PROCEDURE [a: LONG POINTER] RETURNS [FileSegmentHandle];

XFileSegmentAddress: PROCEDURE [seg: FileSegmentHandle]
    RETURNS [LONG POINTER];
```

The following definitions logically belong in **AltoDefs**. They define parameters of the extended memory system.

```
PagesPerBank: PageCount = . . . ;
MaxXPage: PageNumber = 1777B;                    -- analogous to MaxVMPage
```

The following signal logically belongs in **ImageDefs**, and is raised when MakeImage (or CheckPoint) discovers a segment in the extended memory banks that cannot be swapped out. (See the section on restrictions, below, for more information about image files). If this signal is **RESUME**d, MakeImage assumes that the segment in question has been removed from high memory by the client; failure to do so may result in an image file that cannot be restarted.

```
ImmovableSegmentInHighBank: SIGNAL [SegmentDefs.SegmentHandle];
```

Long Pointer Support

The facilities described in this section are defined in the **XMesaDefs** interface and should be used only when the XM condition (see above) is **TRUE**.

XCOPY is analogous to **COPY** in **InlineDefs**, but accepts long pointers instead of pointers and moves data across memory bank boundaries. The restrictions imposed on XCOPY in XMesa 4.1 have been eliminated; all combinations of source and destination banks are now legitimate. **XCOPY** makes no attempt to validate the long pointers; if they exceed 18 bits or reference non-existent memory, **XCOPY** will have unpredictable results.

```
XCOPY: PROCEDURE[from: LONG POINTER, nwords: CARDINAL, to: LONG POINTER];
```

The implementation of XMesa prohibits client programs from manipulating the emulator bank register directly (see section on restrictions, below). Clients wishing to perform a BitBlt across bank boundaries must therefore use the **XBitBlt** procedure, which verifies that the requested extended memory bank is available (raising **XMNotAvailable** if it isn't), then loads the emulator's alternate bank register with **bank** and performs a BitBlt. Because the BitBlt microcode does not support arbitrary 18-bit addresses, XBitBlt can only be used to move data within a single bank or between the MDS bank (bank 0) and some other bank.

```
XBitBlt: PROCEDURE[bbt: BitBltDefs.BBptr, bank: MemoryOps.BankIndex];

XMNotAvailable: ERROR;
```

The following procedures convert between page numbers and long pointers, and are analogous to **AddressFromPage** and **PageFromAddress** in **SegmentDefs**. Illegal parameters are detected and cause the indicated signals to be raised.

```
LongAddressFromPage: PROCEDURE[page: AltoDefs.PageNumber]
    RETURNS[lp: LONG POINTER];

InvalidXMPage: ERROR [page: AltoDefs.PageNumber];

PageFromLongAddress: PROCEDURE[lp: LONG POINTER]
    RETURNS[page: AltoDefs.PageNumber];

InvalidLongPointer: ERROR [lp: LONG POINTER];
```

How to Obtain and Use XMesa

The files for XMesa reside on [Ivy]<XMesa>. The subdirectory structure parallels that used on [Iris]<Mesa>, but only those files that supersede or supplement Mesa 5.0 are stored on <XMesa>. The files on <XMesa> are organized as follows:

<XMesa>
: `XMesa.image`, `.symbols`, and `.signals`
`BasicXMesa.image`, `.symbols`, and `.signals`
`XMesa.cm`, `BasicXMesa.cm`, and `MakeXMDebug.cm`
miscellaneous command files

<XMesa>System>
: system definitions and program modules (`.mesa` and `.bcd`)
miscellaneous command files

<XMesa>Doc>
: `XMesa.press` and `.bravo`
<XMesa>XDebug>
modules to be installed in the Mesa 5.0 debugger (`.mesa` and `.bcd`)

The standard Mesa 5.0 version of `RunMesa.run` will run XMesa programs; no special version of RunMesa is required.

`XMesa.cm` and `BasicXMesa.cm` retrieve the corresponding image files, the debugger, and related supporting files. `MakeXMDebug.cm` installs the debugger, including the modules required for debugging the XMesa environment. (`XMesa.cm` and `BasicXMesa.cm` implicitly execute `MakeXMDebug.cm` as well.)

Note to BasicXMesa users: The standard Mesa 5.0 version of `ImageMaker.bcd` may be used with XMesa programs. (XMesa 4.1 supplied its own.)

Restrictions, Limitations, and "Features"

Images and Checkpoints. MakeImage cannot preserve the contents of extended memory in the image file it constructs. If MakeImage is invoked when the XM condition is **TRUE**, it will swap out all unlocked file segements in extended memory. (It will also move any locked code segments to primary memory.) If any segments then remain in extended memory, MakeImage will refuse to build the image file. Analogous comments apply to CheckPoint.

Bank Registers. XMesa assumes it has exclusive control of the emulator bank register. Client programs must not attempt to alter the bank register, but rather must use the public interfaces for moving data to and from extended memory (see **XCOPY** and **XBitBlt**, above).

Global Frame Format. XMesa exploits the full generality of the global frame format defined in **ControlDefs**. Client programs should not assume that the code segment associated with a global frame can be obtained by writing

frame.code.handle .

The construct

FrameOps.CodeHandle[frame]

should be used instead.

Segment Object Format. The format of segment objects has changed slightly, although (for compatibility) **SegmentDefs** has not been recompiled. The format assumed by XMesa appears in **XMesaOps**. Clients should be unaffected by this change.

Segment Alignment. Segments may not cross bank boundaries.

Resident Code. The amount of resident code in XMesa is 4 pages greater than in Mesa 5.0. XMesa (and BasicXMesa) also have a larger process limit (75) than the standard Mesa system, and therefore consume an additional page of resident data. Accordingly, XMesa may exhibit slightly poorer swapping performance on non-XM Altos than Mesa 5.0 does.

Debugger Incompatibilities. The Mesa 5.0 version of XDebug cannot be used with XMesa unless the module `XMDebug.bcd` is loaded when the debugger is installed. In addition, the debugger's CoreMap command does not work at all under XMesa. The UserProc `XCoreMap` should be installed in the debugger and used instead. The command file `MakeXMDebug.cm` will install the debugger properly for use with XMesa.

Swapper Algorithms. The XMesa swapper loads a segment into extended memory by first swapping it into primary memory, then copying it to extended memory and releasing the primary memory space. Thus, if primary memory is so full that the requested segment cannot be swapped in, **InsufficientVM** will be raised, even though sufficient space for the segment may exist in other banks. (Analogous comments apply when swapping out segments that must be written to disk.)

Distribution:
XMesa Users
Cedar Group
Mesa Group