XEROX

BUSINESS SYSTEMS

Systems Development Department

To:	Gateway Administrators	Date:	March 23, 1981 12:04 AM
From:	Hal Murray	Org:	SDD/Comm
Subject:	Alto Gateway Operation	Filed:	[IVY]<Portola>Mesa6>AltoGateway.press

This memo documents the operation of the Mesa 6 version of the Alto Gateway program. You are assumed to be reasonably familiar with Altos. Familiarity with Mesa and the Tools environment will be very helpful.

I have avoided any details of the current network topology since it is changing frequently. There are two documents that are very helpful when trying to track down network troubles. They are `[MAXC2]<AltoDocs>NetTopology.press` and `[IVY]<Portola>Pup-Network.txt`. I suggest that you keep a reasonably up to date copy handy.

0. Changes since the Mesa 5 version

There have been several major changes in the Gateway program since Mesa 5.

The user interface has changed drastically. The Gateway program now uses the Tools interface. You will need SysFont.strike on your disk.

The name of the typescript file has changed from Mesa.typescript to Gate.typescript.

The name of the parameter file has changed. User.cm now contains a pointer to the real parameter file, which is now named *hostname*.txt, where *hostname* is the name of the Gateway. The format of the parameter file has changed drastically. The details are described in `GateParameter.press`.

The Gateway program now includes an FTP Server TimeChecker, LineWatcher, NetWatcher, and HostWatcher.

The boot server now make occasional long range probes. This will propagate new versions of boot files past Gateways that don't have enough room for them on their disk.

Password protection is now available for dialup lines and the FTP server.

The EFTP portion of the GateControl Server has been removed. Use FTP to store and/or retrieve files.

1. Normal Operation

Power Up and such

There is nothing special about turning the power on or off for an Alto that is being used as a Gateway.

Starting the Gateway Program

Type the command `Gateway` followed by carriage return. The program will read the parameter file, turn off the display, pause for a while, and finally print the message "Alto Gateway of *date-time* in operation". Since the display is normally off, you won't be able to see the last message. Control-Shift-D should turn the screen back on. At this point the Gateway program is running. It takes a few seconds for new routing information to be propagated throughout the inter-network, more if a packet gets lost, so it may not be possible for connections to be established through the Gateway immediately.

After the Gateway proper is running, other servers, primarily as the Boot Server and Time Server, are started sequentially.

The Alto Display

The Gateway program uses the Tools environment to control the display, mouse and keyboard. (The Tools environment is also used by the Mesa Debugger, so most Mesa programmers will be able to give you a quick introduction.) If you are not familiar with the Tools environment, you should probably read the Tools User's Guide from `[Iris]<Tools>Doc>TUG-main.press`.

The display is normally turned off while the Gateway program is starting. This conserves CPU cycles. Normally the screen will be all black. Control-Shift-D will complement the ON/OFF state of the display. Control-Shift-I will invert the black/white mode of the display. If you type anything on the keyboard, even while the display is off, the screen will blink to let you know that the program is still alive.

There is a background process that wakes up every 1/4 second. If the display is off, it smashes the cursor to a big G, moves the cursor down one step, and moves the cursor right one step for each packet that was forwarded since the last time the process got to run. Actually, that description is a bit misleading. The background process really moves the mouse. The Tools keyboard watching process then moves the cursor to track the mouse. That's why the cursor will momentarily switch to other strange shapes as it wanders across the screen. If the cursor overflows the bottom or right edge, it wraps around and starts over again. If the display is on, cursor is not moved. When you turn the display on, the cursor will probably be left as a big G. It will get fixed up it you move it across a window boundary.

Most messages printed on the screen are also saved in Gate.typescript.

If the Gateway has an SLA line driven by an EIA board, there are troubles keeping the clock accurate when the display is on. To avoid sending out bogus information, the Pup Time Server is disabled whenever the display is turned on. It is automatically reset when the display goes off.

Gateway Program Commands

The Gateway program has a simple menu driven command interpreter. Again, it uses the Tools interface, so if you have problems figuring out how to do things, find a Mesa programmer to help you (it won't take long) and/or look at the Mesa Debugger documentation. *Remember, if you are using an EIA board, the time server is disabled whenever the display is on.* This discussion does not mention many of the obscure operations that are possible. If you discover an interesting one that I left out, please tell me so that I can update this memo.

Gate

This menu is available when the mouse is within the main Alto Gateway window, or when the mouse in not inside any window.

Gate

This will print out a summary of various operating statistics, including the

length of time Gateway program has been running (hours:minutes:seconds), the number of times each server has been invoked (explained later in this memo), and a matrix showing number of packets forwarded from one directly-connected network to another. The number of discarded packets is also listed. Discard of packets is not cause for alarm: it is a normal consequence of the great disparity in speed between the Ethernet and the leased lines, and does not give rise to loss of information in file transfers or terminal connections.

Route

This will print the routing table used by the Pup Software to determine where to send a pup to get it to its final destination. 0 hops means that this gateway is directly connected to that network by the net and host number indicated. If the hop count is greater than 0, then the pup will be forwarded to another gateway at the indicated address for further processing.

Recent and **Total**

These commands print out reams of statistics. The numbers printed by **Total** include everything that has happened since the Gateway program was last restarted. The numbers printed by **Recent** include only what has happened since the last time the Recent Statistics were printed.

Quit

This is how you stop the Gateway program. You must confirm this command by poking the left mouse button. Any other mouse button aborts the command. It may take a while for the Gateway program to unwind if, for example, the boot server is retrieving a new boot file.

DeviceInfo

This menu is available only when the mouse in not inside any window.

Ether1-xx

This simply prints out the statistics for the Ethernet interface (there may be more than one) connected to net xx.

SLA-7

This prints out operating summaries for the Synchronous Line Adaptor (SLA) i.e. the EIA board or the CommProc. For each line, the number of packets successfully sent and received on that line is printed, followed by the number of instances of three types of errors: CRC (Cyclical Redundancy Check) errors, Sync errors (bit synchronization was lost), and Control sequence errors. The total number of errors should be much less than one percent of the packets successfully received. If a high error rate occurs on a single in-use line (whose state is "Up"), the line or modem is suspect, whereas if frequent errors occur on all lines (or particularly on "Looped back" lines), the SLA interface is suspect.

This command also prints out the SLA routing table. Under normal circumstances, the routing table for a particular Gateway should show that it can reach all other Gateways through one or more of the connected lines.

Booter - Table

This prints out the table used by the boot server. If you are low on disk space, this may help you find boot files that are not used very often.

Names - Cache

This simply prints out the cache used by the Name Lookup Server.

Inactive - PupEcho

There is an Echo user program built into the Gateway. There are actually three copies of the same program. To run it, fill in the desired **Target**, and poke **Start**. If you turn on **Verbose**, you will see a ! for each successful packet. **Stop** is, of course, used to terminate a run.

ToolMgr - CreateSource

There is an Editor built into the Gateway. This may be a convient way to make a minor adjustment to the parameter file. Look in the Mesa Debugger Documentation for more info. There is also one in the debugger if you have the debugger installed on your disk.

Bugs

If the Gateway program gets into trouble, it will probably get to Swat (or the Mesa Debugger if there is one on the disk) print a message on the display and wait for a person to do something. At this point, there are two options. If you have the Mesa Debugger on your Gateway disk, you can poke around. Normally there won't be room for it. In that case, you can only poke around from Swat. I think I can do a reasonable job of diagnosing bugs if you are careful to save `Swatee` at the right time. The trick is to be sure the the right information is there, and to avoid clobbering it. If the Gateway program is hung (not forwarding packets), try to get to Swat via Control-Shift-Swat. If that doesn't work try Dumper.boot. (It should be under the DU keys.) If you have to use Dumper, please save the contents of R registers 1, 2, 10, 15, 16, 51, 52, 53, and 54. (^R to Swat will print them.) If Swat (or the Mesa Debugger) has printed out an error message, please write it down. Next, reboot the Alto. Control-K or Shift-Swat will probably overwrite `Swatee`. (Empress uses `Swatee` as a scratch file, so be sure to save things before making a listing of Gate.typescript.) Then FTP `Swatee` and `Gate.typescript` away to a safe place. Remember that FTP only knows how to talk to machines that are connected to the normal Ethernet board. If you don't have an IFS handy, you can store it on another Alto until you get the Gateway back up again. Please send me a message telling me what went wrong and where you put `Swatee` and `Gate.typescript`.

There are three general categories of troubles. The first is a problem during initialization. Hopefully, the text will be sufficient to solve the problem. The most likely cause is an error in your parameter file. If you run into a message that is too cryptic, please let me know, and I will fix it. The second is an uncaught SIGNAL. It will be printed in octal, and a listing of `Gateway.signals` (from `[IVY]<Portola>Mesa6>Gateway.signals`) will probably help translate it into English. The third catagory of trouble is an inconsistency discovered by the Pup Software. Again an octal number will be printed, and you will need a listing of Gateway.signals to translate it into English.

One common uncaught SIGNAL is InsufficientVM (1015B). This normally means that the first 64k of memory didn't have enough room to swap in a large code segment. The arg is the number of pages that were needed. The normal cause for this problem is allocating too many buffers. Try adjusting the `Buffers` parameter to leave more space, but be careful not to go too far.

Another common SIGNAL is DiskFull (2201B). This normally happens while it is trying to extend the typescript file during retrieval of a new copy of a boot file that won't fit. The

simple case of retrieving a new boof file that doesn't fit normally recovers without crashing. The easy way to avoid this problem is to leave enough space for a new copy of the biggest boot file in your list. Othello is over 400 pages, and the system needs another 100.

Another SIGNAL that happens occasionally is UnrecoverableDiskError (605B). This probably means that your disk is sick. Try running the Scavenger. If it has troubles with a boot file, you should delete the damaged file to be sure that garbage doesn't propagate around the net. If the boot file number is less than 100000, the boot server will retrieve a new one automatically.

A *Code Trap from Interrupt Routine* is either a software bug, or the result of a hardware failure. Please contact me if your hardware is healthy.

2. Functions Performed by the Alto Gateway

The primary purpose of the Gateway is to forward Pups (Parc Universal Packets) from one network to another. However, the fact that the Alto is somewhat underutilized by this task and that it is in operation all the time makes it a desirable source of other services as well. These services are necessarily limited to those that are relatively easy to implement and whose resource requirements don't significantly impact the primary role as a Gateway, but they make life considerably more pleasant for Alto users than would be the case were these services not available.

The Gateway program provides the following services:

Gateway Information

This service is actually related intimitely with the Alto's role as a Gateway. It provides routing information to all hosts on directly-connected networks. It is by means of this routing information that Pup software such as FTP is able to communicate with hosts on other networks.

Date and Time

The Gateway program maintains the date and time in a form usable by Altos. The SetTime EXEC command obtains the date and time by this means.

Name Lookup

This service translates inter-network name/address expressions into addresses usable for communication. When an Alto subsystem is requested to establish contact with a host addressed symbolically (e.g., "Maxc"), it generates a name lookup request, to which the gateway responds by supplying the corresponding inter-network address (e.g., "3#200#").

Boot

Many machines are capable of boot-loading over the Ethernet if a suitable server is available to provide the boot files. The Gateway provides this service for a limited set of boot files that are useful on a machine with no disk loaded or with a disk that won't boot (e.g., DMT, Scavenger, FTP, CopyDisk, and a NetExec providing more convenient access to the other boot files). There is no intention, however, of making available a complete set of subsystems by this means.

Echo

This service consists simply of a process that echoes all received packets back to their source. It is useful for hardware and software diagnosis.

In addition to these services, the Gateway program also implements several internal support

protocols, such as the one that automatically updates the network directory data base.

3. Contents of the Gateway Disk

The Alto Gateway disk contains all of the files needed for normal operation of the Alto Gateway. Since space is tight, it probably does not contain much else. Be sure to keep at least 400 free pages on your Gateway disk. If you don't have enough there won't be any way to update big files. If you are trying to clean up your disk, you can delete *$ and Gate.typescript, they will be created again when needed.

Gateway.image

> This is the Gateway program itself. This file is updated when a new version of the Gateway program is released.

User.cm, *hostname*.txt

> These are the parameter files needed by the Gateway program. See GateParameter.press for a description of the required contents.

MesaGateEIAChain.br or MesaGateCPChain.br or ....

> This is the microcode loaded into the RAM.

Pup-Network.Directory

> The local copy of the inter-network name data base. The master source is [Ivy]<Portola>Pup-Network.Directory. Pup-Network.Directory is updated automatically by the Gateway program (and other name lookup servers).

DMT.Boot, Chat.boot, NewOs.Boot, FTP.Boot, Scavenger.Boot, CopyDisk.Boot, NetExec.boot, and a number of others

> These are Alto bootstrap files, given out by the Gateway program when an Alto is boot-loaded over the Ethernet. The master copies are stored on [Maxc]<BootFiles>. They are updated automatically by the Gateway program from neighboring Gateways. They may have been "reformatted" so that they are NOT useable by the BootFrom command to the Exec.

*.Scratch$, NewGateway.image

> *.Scratch$ are temporary files used while getting a new version of Pup-Network.Directory, new boot files, or during remote updating of files from GateControl. There must be enough room on the disk for it to hold a copy of the biggest file that will ever be remotely updated. NewGateway.image is a temporary copy of the Gateway program used when remotely distributing a new version of the Gateway software. It is needed for remote updating and automatic restarting because Mesa uses Gateway.image for code swapping, and is automatically deleted by the Restart command from GateControl.

SYS.boot, Swat, Swatee, Executive.run, FTP.run (and whatever)

> These are just the normal files/programs needed to get any Alto off the ground. FTP isn't really needed, but it is the simplest way to put things back together and/or save information needed to chase bugs.

SYSFont.strike

> The Tools environment needs a strike format font. I use Gacha10. If you use something different, some printout may overflow a line or something equally

harmless.

EDP.run, PupTest, Scavenger.run (and whatever)

These are just the normal files/programs needed to get an Alto running again if something goes wrong. You don't really need them on your Gateway disk, but you need to keep a copy someplace handy since you can't boot them over the Ethernet if the Gateway program/disk/Alto is busted.

Gate.typescript

Gate.typescript is a log of everything that gets printed on the Alto display.

RunMesa.run

RunMesa.run is the bootstrap program that loads the normal Mesa microcode into the RAM, and then loads Gateway.image into memory and starts running it.

XDebug.image, MesaDebugger, Debug.log

These are the Mesa Debugger. They will be on your disk only if it is setup for debugging -- there isn't room for them normally. NB: Do not move or delete any of them unless you know what you are doing. They contain FPs.

PupTypes.bcd, BufferDefs.bcd, DriverDefs.bcd, ...

These files contain the symbol tables for the Mesa Debugger. Again, they will be on your disk only if you are chasing a nasty bug.

4. Gateway Disk Maintenance

The Gateway software includes the capability for remote updating of files on the Gateway disk. It is also possible to remotely command the Gateway to restart, to reset its date and time, and to perform other operations. We use this remote control capability at Parc to release new versions of the Gateway program and to update the boot files. Therefore the following procedures should not be needed in the ordinary course of events.

Note that you are in deep trouble if your last Gateway disk gets clobbered. It is a very good idea to keep a backup copy. You can easily make one using CopyDisk.

A Gateway disk can also be configured to run on any XM Alto. It won't be able to forward packets to any other network, but it will provide luxuries like a Time server and a Boot server which make working on Altos much more pleasant.

Obtaining Updated Files

This section describes procedures for obtaining new Gateway software or other files from Parc and installing them on the Gateway disk. This involves stopping the Gateway program briefly, so one should first make certain that there are no users who might be affected.

The following procedure assumes that the file being updated is Gateway.image, but it applies to any other file. [IVY]<Portola> is the normal location for released Gateway software. A few of the boot files come from [MAXC]<Alto>. The procedure requires use of a second Alto connected to the same Ethernet as the Gateway Alto.

1. While the Gateway is still in operation, run FTP on an second Alto, connect to IVY, and retrieve the file `<Portola>Mesa6>Gateway.image`. After closing the connection to IVY, leave the Alto running FTP.

2. Type the **Quit** command on the Gateway keyboard to return control to the Alto Executive.

3. Run the FTP program on the Gateway Alto, connect to the second Alto, and retrieve Gateway.image. Note that since no name server is running at this point, it is necessary to refer to the second Alto by number rather than name (e.g., if its Ethernet address is 123, you Open a connection to "123#"). After retrieving the file, quit out of FTP to the Alto Executive.

4. If desired, back up the Gateway disk.

5. Restart the Gateway program by issuing the **Gateway** command.

One might wonder why it is not possible simply to connect directly to IVY from the Gateway in order to retrieve files. In order to do this, FTP would have to communicate over the SLA line to Parc. However, FTP (and other standard subsystems) does not contain the software for driving the SLA interface, but only contains a driver for the Ethernet. Hence, FTP running on the Gateway can communicate only with machines on the directly connected Ethernet.

5. PupTest Operation

PupTest is a program developed primarily for checkout of the Pup software; however, some of its capabilities are helpful in locating and troubleshooting network problems. *Be sure to keep a few copies of PupTest.run on various disks because you can't boot it from the Gateway when you really need it.*

We describe here only those PupTest commands useful for troubleshooting network problems.

The usual symptom of failure is likely to be that a user at a remote location is unable to access Maxc or some other machine not on the local Ethernet. There are a multitude of problems that can cause this, and one should not jump to the conclusion that the leased line or the Gateway is down. The following procedure should pinpoint the failing link.

The simplest test is to try to connect to some other server that is known to be on the same remote network. If you can connect to either MAXC2 or IVY, but cannot connect to MAXC, then MAXC is probably down.

The PupTest command used in this procedure is **Echo**. It requests the name or address of a host running an Echo Server, and it then sends Echo requests to that server. All Gateways have Echo servers, and any machine running PupTest itself has an Echo server. For each reply PupTest receives, it prints "!", and if no reply is received, it times out after 1.5 seconds, prints "?", and tries again. It prints "#" whenever it gets an packet with an incorrect sequence number. Normally this happens just after it printed a "?", which simply means that it didn't wait long enough. The test terminates when any key (e.g., space) is pressed.

1. Run PupTest on two Altos connected to the same Ethernet, and direct one of them to Echo to the other. One should address the other Alto by number rather than name; e.g., if the other Alto's Ethernet address is 123, one should say "Echo to: 123#". If this doesn't work, then the local Ethernet is broken. It is not necessary for the Gateway to be up for this test to work.

2. Echo to the local Gateway; that is, say "Echo to: *n*#", where *n* is the Gateway's address on the local Ethernet. If this doesn't work, then the Gateway program is down or the Gateway or its Ethernet interface is broken.

3. Echo, in turn, to each of the other Gateways along the path from the local Gateway to the unreachable destination. The topology is geting pretty complicated now, so you will have to be careful doing this. (It should be possible to reference these machines by name since the name lookup is performed by the local Gateway, which is known to be up by step

2.) If this doesn't work, then there are a number of possibilities, some of which can be checked out by running PupTest on the Gateway (see step 4). If it is possible to echo to the last Gateway along the path to some desired host, then the most likely reason for inability to reach that host is that the host itself is down. If it is possible to echo to the second Gateway along the path to the unreachable destination (the local Gateway is the first), but it is not possible to echo to the last Gateway, then the failure is in the first unreachable Gateway or the modems or leased line between it and the next nearer Gateway. Determining which component has gone down requires cooperation with personnel at other sites.

4. Unplug the line to the modem and put in an echo plug. Be sure to unplug the echo plugs parked in any spare slots. Activate `PupEcho` and tell it to echo to itself through the looped back line; that is set Target to: 7#n# where n is the assigned SLA network address for your machine, and poke `Start!`. The normal behavior in this case is to flip the indicator box reasonably rapidly. Try it sometime so you will know how fast it should go. A "?" will be printed each time a packet is not returned correctly. If this doesn't work, the SLA interface is probably broken. Be sure to plug the modem back in.

5. After plugging the modem back in, press the AL (Analog Loopback) button on the modem so that it stays in its "in" position. Again, direct `PupEcho` to echo to itself through the SLA driver. If this doesn't work, the modem is probably broken. When you are finished with this test, be sure the modem's AL button is in its "out" position.

6. If the previous test worked, but it was not possible to echo to the Gateway at the other end of the link, then the failure is in the other Gateway or the modem at the far end of the link or in the leased line itself. Determining which component has gone down requires cooperation with personnel at the far end.

To force Echoing to go through a particular line, unplug all of the others.

The Bell 209 modems that are used for most of the Xerox-Pup network have several diagnostic aids. If you push in the AL button, the modem should loop things back locally. The SLA statistics should show a state of Looped-back within 5 seconds. If you push the DL button at the remote modem, it will send back whatever it receives. Again, the SLA line should show a state of Looped-back. If a line gets looped back when the local modem is in AL mode, but not when the remote modem is in DL mode, the line or one of the modems is probably sick. You might learn something by trying the tests in the other direction.

There is one nasty problem that we noticed once. If the DL and AL tests work from both ends, yet the Gateways can't talk to eachother, it may be because the baud rate switch on one of the modems has been bumped.

6. Notes to Gateway Maintainers

Be sure to keep at least 400 free pages on your Gateway disk after *.scratch$ has been deleted. If you don't have enough there won't be any way to update big files. If your Gateway is also a boot server providing Othello to D0s, you should have at least 600 free pages.

When receiving a new copy of Gateway.image via FTP, the Gateway program automatically changes the name of the incoming file on the Gateway disk to be NewGateway.image. The GateControl Restart command looks for this filename. The running program is swapping code out of Gateway.image, and it will probably die horribly if the code were changed out from underneath it.

If a file already exists when an update starts, the incoming data is stored in a temporary file until it is all collected to protect against clobbering things if the EFTP transfer fails. If a file does not exist when the update starts, the incoming data is written directly into the target file to avoid filling up

the disk with two copies. If the transfer fails, the partial file is deleted.

The Restart command appends a few commands into Rem.cm, so you can do almost anything you want to a Gateway disk remotely by storing your own commands in there ahead of its restart sequence. If NewGateway.image exists when the Gateway is being restarted, Gateway.image will be deleted and NewGateway.image will be renamed into Gateway.image. At the end is a command to run Gateway.image.

Whenever the Gateway program starts up, it will reformat any boot files in its list that have not been previously reformatted. That takes a while if it actually does any reformatting. This should not happen very often.

If the Gateway program is forwarding a great deal of traffic, there will not be any CPU cycles left over for other things. This will probably only be noticable if you are trying to get a boot file while running PupTest between two machines connected to different Ethernets via the same Alto Gateway. The Gateway will support about 400 kilobits of throughput, but it may take over a minute to get a big boot file while that is happening.

7. Hardware Requirements

To run the Alto Gateway software requires appropiate hardware. Here is a quick checklist:

First, you need an Alto with at least 128K of memory. You need a 3K RAM or, XMesa 5.0, or Mesa 6 must be blown into ROM1. Beware, there are some proms with strange versions of XMesa that don't work with the Gateway microcode. A D0 can be used if you don't need to use any phone lines.

If you are sending a set of proms to another site, I suggest that you test them first. It may save a lot of time. Similarly, if you are having troubles bringing up the Gateway software on a new machine, try testing your disk on a different Alto. The Gateway program uses several features of the hardware that are not normally used, and they may never have been debugged.

A single Alto disk is quite full if you have the normal collection of boot files. If you want a large collection of boot files, or the Mesa Debugger and some symbols, a second disk drive will be necessary.

If your Gateway is going to talk to a phone line, you will need either an EIA board(s) or a CommProc. An EIA board costs about $1200 per line. It does not seem to run faster than about 20KB, but I don't know why it doesn't. A CommProc costs about $4500 plus $500 per line. It works ok at 56KB. 6 lines should work ok at 9600 buad.

If you are using an EIA board, MRT must be made Ram-Related. Connect MRTACTIVE (slot 11, pin 109) to either TASKA' (slot 10, pin 13) or TASKB' (pin 14) whichever isn't being used by other devices. The CommProc uses TASKA'. SmallTalk won't work on a machine with this modification.

The CommProc needs another backplane jumper from J11 pin 108 (for 9600 baud, pin 111 for 57KB) to pin 105 of the slot containing the LIM driving the line. That brings an internal clock out to pin 18, which is normally unused, on the modem connector. The LoopBack Plug (described below) expects the Alto to provide a clock on pin 18.

If you are going to connect to more than one Ethernet, you will have to aquire and install extra Ethernet boards. See `[IVY]<Portola>ExtraEther.press`. An Alto can handle up to three Ethernets. The Gateway software expects the first extra Ethernet to use task 2 and SIO bits 12 and 13. It also expects the second extra Ethernet to use task 1 and SIO bits 10 and 11. CopyDisk won't work if you have added a third Ethernet board.

If you have a CommProc, and you are adding a third Ethernet, you will have to undo the

CommProc timer. It is not used by the Gateway software. Remove the wires connecting pin 119 on slot 16 to pin 119 on slot 11 (1ACT'), and pin 114 on slot 16 to pin 114 on slot 11 (WAKE1'), or simply omit them when you install the CommProc.

Modem Cable:

The Alto-to-modem cable has a female DB-25 connector at the Alto end and a male DB-25 at the modem end. *Do not connect pin 18 straight through.* The Bell 209 modems get confused by the loopback clock. The following signals should be transmitted through the cable:

Pin	Signal
1	Protective ground
2	Transmit data
3	Receive data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal ground
15	Transmit clock
17	Receive clock
20	Data Terminal Ready

In practice, you can save 4 wires by jumpering pins 4 to 5 and pins 6 to 20 at *both* the modem and the Alto end of the cable.

Modem Options:

The cable described above and a CommProc do not seem to work with some Bell 209 modems. I have not tracked down the problem, but the following set of jumpers works at the Bayhill building. They are located inside the front cover of the modem. You have to bend the plastic a bit to unhook the 4 hooks that hold the cover on.

WN, WK, WS
WF, WH,
WP, WB, WD
YX, ***XI***
YC, WJ, YI, ***YM***

The factory defaults are XG instead of XI, and YN instead of YM. XI controls the generation of carrier, so if it is wrong, you will discover it promptly. YM is only interesting when using the AL test mode. If your modem has XG, move it to XI and things will probably work. Similarly, move YN to YM if necessary. You will probably need a small pair of pliers since YN and YM are hidden between the switches and the frame. It is probably a bad idea to change any of the other jumpers even if they are different from this list.

LoopBack Plug:

A loop-back-plug is very useful for debugging. It is just a female DB-25 connector with the following jumpers:

Pins	Signals
2-3	Transmit data to Receive data
4-5	Request to Send to Clear to Send
6-20	Data Set Ready to Data Terminal Ready
15-17-18	Transmit clock and Receive clock to Internal clock

8. Building a Gateway Disk

One way is simply to copy a disk that you like and modify it as necessary. In this case, you will need to:

rename *HostName*.txt to be *YourHostName*.txt
update the Indirect section of User.cm to point to *YourHostName*.txt
change the hardware configuration in *YourHostName*.txt, and
update the [BootServer] and [TimeServer] sections as appropriate.

Alternatively, you can start from scratch by using the following recipe:
get a new disk (or two) and install NewOS.boot with the Erase option,
retrieve [Ivy]<Portola>Mesa6>NewGatewayDisk.cm and run it,
copy and/or merge NewGateway-user.cm into User.cm,
run the Gateway by typing Gateway and RETURN.
This will start up the Gateway program using a default parameter file. After it finishes retrieving new boot files and things (this takes several minutes), you can update the Indirect section of User.cm and Edit/Rename NewGateway.txt to *YourHostName*.txt.

See [Ivy]<Portola>Mesa6>GateParameter.press for the details of parameter files.

In either case, there is a slight problem editing the parameter file because there is not normally room enough on a Gateway disk for Bravo. If you don't make any errors, you can use the Tajo editor. If you keep copies of the initial files, you can even recover from errors that kill the Gateway by copying them over again. Another approach is to delete enough boot files (or do the edits before they arrive) to make room for Bravo. You can also use FTP.run to move the files to a disk that is already setup with an editor.

Remember, if you smash your last copy of the Gateway disk, you may become cutoff from the world. A backup copy is good insurance.

Revision history:
October 13, 1978, initial release
December 7, 1978, XMesa, 209 testing hints, LineWatcher, various
December 22, 1978, 209 options, SysFont
June 18, 1979, Update for Mesa 5
November 21, 1979, Minor updates
June 11, 1980, YM/YN option
December 2, 1980, Mesa 6 update
December 8, 1980, Dialup passwords
January 12, 1981, FTP Server and HostWatcher
March 11, 1981, Split out parameter file info