Genera 8.0 ECO #2 Notes

Overview of Genera 8.0 ECO #2

This is the second ECO to Genera 8.0, and is also called Genera 8.0.2. Genera 8.0.2 includes Genera 8.0 ECO #1, so this documentation includes the documentation of ECO #1 changes.

Genera 8.0.2 offers two new areas of functionality:

- Software support for the UX1200S.

  The UX1200S is a more powerful, higher-performance version of the UX400S. The UX1200S has the same processor architecture as the XL1200, whereas the UX400S has the same processor architecture as the XL400. See the section "Overview of the Symbolics UX1200S".

- Software support for XL1200 Single-Monitor Color Stations.

  XL1200 single-monitor color stations include an XL1200 with a FrameThrower color board and a color console. This station requires only a color console, unlike other Symbolics Ivory-based color stations, which require both a color and a black-and-white console. See the section "XL1200 Single-Monitor Color Stations".

Installing Genera 8.0 ECO #2

Genera 8.0 ECO #2 is distributed with XL1200 single-monitor color stations and UX1200S systems.

To install the software, refer to the appropriate sections in *Genera 8.1 Software Installation Guide*. This section describes some important changes to the installation procedure for the ECO #2 release.

Installing a Symbolics UX1200S System

For installation instructions, see the section "Installing the Genera Software on a UX". Follow those instructions, but note this update to "Restoring the Genera Distribution Worlds From CD-ROM":

The name of the world load file to use is:

```
Genera-8-0-2-Network.ilod.1
```

If you are a user of the UX1200S delivery system, the name of the world load file to use is:

```
Genera-8-0-2-UX-Delivery.ilod.1
```

Be sure that you use the correct name of the world load file in step 3 (when restoring the world load file), and in step 4 (when editing the `boot.boot` file).

After you have restored the world load file and edited the `boot.boot` file (after step 4), you need to evaluate this form in a Lisp Listener:

```
(si:install-fep-kernel "I322-kernel")
```

Next, you need to edit the `hello.boot` file to scan the I322 FEP files, instead of the older FEP flods. Currently, you might be using I321, or I318, or I317 FEP files. You need to replace the old FEP version number with I322 wherever it appears in the `hello.boot` file.

Proceed as directed in "Restoring the Genera Distribution Worlds From CD-ROM".

Installing an XL1200 Single-Monitor Color Station

For installation instructions, see the section "Tape Installation Steps for XL-family Machines". Follow those instructions, but note this update to "Restoring the Genera Distribution Worlds From CD-ROM":

The name of the world load file to use is:

```
Genera-8-0-2-Color.ilod.1
```

Be sure that you use the correct name of the world load file in step 3 (when restoring the world load file), and in step 4 (when editing the `boot.boot` file).

After you have restored the world load file and edited the `boot.boot` file (after step 4), you need to evaluate this form in a Lisp Listener:

```
(si:install-fep-kernel "I322-kernel")
```

Next, you need to edit the `hello.boot` file to scan the I322 FEP files, instead of the older FEP flods. Currently, you might be using I321, or I318, or I317 FEP files. You need to replace the old FEP version number with I322 wherever it appears in the `hello.boot` file.

If this is a new XL1200 single-monitor color station, you can proceed as directed in "Restoring the Genera Distribution Worlds From CD-ROM".

Extra Steps for XL1200 Single-Monitor Station Upgrades

For users who are upgrading from an XL1200 to an XL1200 single-monitor color station, you must perform these extra steps, after you install the FEP kernel by calling **si:install-fep-kernel** as shown above.

1. Give this command at a Lisp Listener:

   ```
   Halt Machine
   ```

2. Push the RESET button on the XL1200 to cold-boot the FEP.

3. Give the following FEP command:

   ```
   Set Disk Label 0 :Color System Startup File FEP0:>I322-FrameThrower.sync
   ```

4. Perform the hardware upgrade (install the color console and remove the black-and-white console).

5. Reboot, and proceed as directed in "Restoring the Genera Distribution Worlds From CD-ROM".

Overview of the Symbolics UX1200S

The UX1200S is a more powerful, higher-performance version of the UX400S. The UX1200S has the same processor architecture as the XL1200, whereas the UX400S has the same processor architecture as the XL400.

UX1200S Software Requirements

The UX1200S requires Genera 8.0.2. Note that the UNIX software shipped with Genera 8.0.2 to UX1200S customers is compatible with SunOS 4.1 only.

In a configuration where multiple UX boards are installed in a single Sun, each UX board runs its own copy of Genera. In a configuration that includes mixed UX400S and UX1200S boards in a single Sun, you can run the Genera 8.0.2 software on the UX1200S boards and run the Genera 8.0 or 8.0.1 software on the UX400S boards; the Genera versions are compatible.

Note that Genera 8.0.2 can also be run on UX400S boards, but Genera 8.0.1 and Genera 8.0 cannot be run on UX1200S boards.

UX1200S Documentation Update

The documentation on the UX400S pertains to the UX1200S, except as noted in this section.

On the UNIX side, `/dev/ivory0` is the device for the first UX400S board. Any additional UX400S boards are in `/dev/ivory1`, `/dev/ivory2`, and so on. Similarly, `/dev/ivory16` is the device for the first UX1200S board. Any additional UX1200S boards are in `/dev/ivory17`, `/dev/ivory18`, and so on.

Changes to the FEP in Genera 8.0.2

On Ivory machines with more than one console attached, the FEP now prints a greeting on each console. (This was already done for 3600-family machines.) In Genera 8.0.2, the FEP on Ivory machines prints the greeting, all delayed errors, and all warnings on all consoles. On each non-selected console, the FEP prints the command you can type to select that particular console.

In Genera 8.0.2, some FEP commands which were previously available only for 3600-family machines are now available for Ivory machines. These commands are the Set Console FEP command and the Set Monitor Type FEP command.

Genera 8.0.2 includes some new FEP commands: Set Disk Label and Set FEP Options. See the section "Set Disk Label FEP Command". See the section "Set FEP Options FEP Command".

In Genera 8.0.2, you can use a serial terminal to communicate with the FEP. See the section "Using a Serial Terminal to Communicate with the FEP".

XL1200 Single-Monitor Color Stations

Overview of XL1200 Single-Monitor Color Stations

XL1200 single-monitor color stations include an XL1200 with a FrameThrower color board and a color console. This station requires a color console only, unlike other Symbolics Ivory-based color stations, which require both a color and a black-and-white console. The color console includes a Sony monitor and a color console unit which typically is placed under the monitor. Three cables attach the monitor to the FrameThrower color board in the XL1200. The color console unit is connected via the normal black-and-white cable and supports the mouse, keyboard, and console serial port.

The color screen is used for all displays: device PROM, FEP, Genera, and Color. Because the FrameThrower is completely programmable, there are additional files and commands that support configuring the FrameThrower to a particular color monitor.

The device PROM and FEP use the FrameThrower in a "reduced capability" mode to simulate a black-and-white monitor. Once Genera is running, the full power of the FrameThrower is available.

Notes About Using an XL1200 Single-Monitor Color Station

This section mentions explains some things you will notice when first using an XL1200 single-monitor color station.

- Booting is slower than on non-color stations.

  It takes longer to boot a color world because it is a larger world, and because it takes time for Genera to fully initialize the FrameThrower.

- Memory is used up more quickly than in a non-color station.

  Think about what happens when you switch windows, by using the `SELECT` key. For example, you are in the Lisp Listener and you enter `SELECT E` to go into the Zmacs editor. Genera saves a copy of the Lisp Listener window; this is called a bit-save array. Genera draws a saved copy (or creates and draws a new copy) of the Zmacs window, another bit-save array. In a color station, a bit-save array is eight times larger than in non-color stations, because one of the dimensions (call it depth) of the array is 1 in non-color and 8 in a color station. Therefore, switching windows uses up a lot of virtual memory.

Some suggestions for coping with this problem are to keep the Dynamic Garbage Collector turned on, and to run a nightly GC with GC cleanups.

Note that the color software makes more intelligent use of the FrameThrower than does Genera, so using the color software features does not use up memory as much as using Genera does.

- Background greying is disabled.

  In a non-color station, when you switch from the Lisp Listener window to a Zmacs window, the Zmacs window appears and the portion of the screen which is not covered by the Zmacs window (the right-hand margin of the screen) is greyed. This background greying does not happen on an XL1200 single-monitor color station, because it would be expensive in terms of using up memory. Genera performs background greying by saving a copy of the window and then saving another copy which is a greyed version of the former copy. For reasons discussed above, these bit-save arrays would use up a lot of memory on a color station.

- Flashing the screen takes a long time the first time.

  When a **beep** causes the screen to flash, for example when you receive a Notification or a Converse message, you will notice a very long beep and the screen going completely blank for a noticeable period of time. Just be patient and wait, and the screen will be repainted normally. Subsequent beeps and screen flashes do not last as long.

Editing the Disk Label

Edit Disk Label Command

Edit Disk Label *unit-number*

Edits the disk label of the disk identified by *unit-number*. Brings up a menu of choices, which you can click on to change aspects of the disk label, such as the FEP kernel, backup FEP kernel, and the color system startup file (used for color systems). You can also click on choices to deinstall the current FEP kernel, deinstall the FEP backup kernel, and deinstall the color system startup file.

Just as the device PROM finds the FEP kernel by reading the disk label, it also needs to find the file where the FrameThrower color system startup programs are stored. Note that the device PROM does not understand the FEP file system; it just reads the disk label, which points to the correct file.

unit-number{*integer*, All} The number identifying the FEP disk whose disk label you want to edit. All allows you to edit the labels of all the disks attached to the machine.

Normally, the FEP kernel and FEP backup kernels are installed automatically, when you use the Copy Flod Files command. Copy Flod Files installs the new FEP kernel and installs the previous FEP kernel as the backup FEP kernel. Only in unusual debugging circumstances do you need to deinstall the FEP kernel or FEP backup kernel by using Edit Disk Label. When you deinstall the FEP kernel, the FEP backup kernel is installed as the FEP kernel. When you deinstall the FEP backup kernel, the FEP kernel is installed as the FEP backup kernel. (Note that the FEP kernel and FEP backup kernel must always be present, in order for this to work properly.) Thus, when you deinstall either the FEP kernel or FEP backup kernel, the result is that a single FEP kernel is installed as the FEP kernel and the FEP backup kernel; in other words, there is no separate FEP backup kernel. When you next use Copy Flod Files, the new kernel will be installed as the FEP kernel, which means you will again have a different FEP kernel and FEP backup kernel.

FEP Commands Useful for XL1200 Single-Monitor Color Stations

The following FEP commands are particularly useful for XL1200 single-monitor color stations:

- Set Disk Label
- Show Disk Label
- Set Console
- Set Monitor Type
- Set FEP Options

Set Disk Label FEP Command

Set Disk Label *unit-number keywords*

unit-number	A disk unit (must be a number in base 10).
keywords	:Color System Startup File, :FEP Kernel, :Query

:Color System Startup File	Pathname of the file where the color system startup programs are stored.
:FEP Kernel	Pathname of the file where the FEP kernel is stored.
:Query	{Yes, No} Whether the system should ask for confirmation before setting the disk label. The default is No.

The Set Disk Label command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show Disk Label FEP Command

Show Disk Label *unit-number*

Displays the information in the disk label.

unit-number	A disk unit (must be a number in base 10).

The Show Disk Label FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set Console FEP Command

Set Console *console keywords*

Allows you to select between the available consoles. For 3600-family machines with CadBuffer2 hardware, this command also switches input from the console's display hardware to the CadBuffer2 keyboard. To make Lisp notice that the console has been changed, you need to reload microcode (on 3600-family machines) and warm or cold boot (on both 3600-family and Ivory machines) after using the Set Console FEP command.

console	On 3600-family machines, the *console* is Color or Monochrome. On Ivory-based machines, the *console* can be any of the available and enabled consoles; press HELP for a list of choices.
keywords	:Clear Screen, :Cold Load Too
:Clear Screen	{Yes, No} Whether to clear the screen before selecting it. The default is No.
:Cold Load Too	{Yes, No} Whether Lisp's cold-load stream (and Main console if you warm boot) should be redirected there also. The default is Yes.

The Set Console FEP command is resident in G208 and greater versions of the FEP EPROM, and in I322 or greater versions of the IFEP kernel, so it needn't be loaded from an overlay (flod) file.

Set Monitor Type FEP Command

Set Monitor Type *console type* (for 3600-family machines with G208
or greater FEP EPROMs, and for Ivory machines with I322 or greater
FEP kernels)

Set Monitor-Type *console-name* (for machines with V127 and G206
FEP EPROMs, and for Ivory machines with I321 or less FEP kernels)

Users of 3600-family machines should use the Set Monitor Type FEP command when installing a monitor that differs from the one specified by the machine's hardware.

On 3600-family machines with G208 or greater FEP EPROMs, sets the console type and name. On 3600-family machines with V127 and G206 FEP EPROMs, sets the console name. This command also loads the appropriate sync program into the machine's display controller, CadBuffer, or CadBuffer2 hardware.

Users of Ivory-based machines should use the Set Monitor Type FEP command if the FEP options installed by the Set FEP Options FEP command are incorrect. At your first opportunity after using the Set Monitor Type command, you should use Set FEP Options to correct the FEP options, and reset the FEP to make the new FEP options take effect.

console	A particular console; this argument defaults to the current console. Use HELP to find out which consoles are applicable. On Ivory machines, a console is specified by three elements: the console type, the sync program, and the unit number. On 3600-family machines, a console is specifed as being color or monochrome
console-name	The list of choices depends on the *console*. Use HELP to find out which choices are applicable.

The Set Monitor Type FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set FEP Options FEP Command

Set FEP Options *keywords*

Sets options which are used by the FEP.

keywords	:Color System Number, :Color System Startup Program, :Color System Type, :Serial Console Type

The Color System keywords are used only when you are using a custom color monitor (other than the default Sony monitor).

:Color System Type
: {None, FrameThrower} FrameThrower enables the FEP to use the color monitor. None disables the use of the color monitor.

:Color System Number
: The number of the color system, a decimal integer between 0 and 255. (You can have more than one FrameThrower, and the color system number enables you to distinguish among them.)

:Color System Startup Program
: Which color system startup program in the color system startup file for the color system to use.

:Serial Console Type
: {None, ASCII, or X3.64} None prevents the FEP from ever us-

ing the serial console; this is appropriate if you have a serial device other than a console connected to the serial port, and know you won't use the serial device as a console. ASCII indicates that the serial console is a dumb terminal. X3.64 indicates that the serial console is an ANSI-standard X3.64 terminal, such as a VT100.

The new options you specify in Set FEP Options take effect when you next reset the FEP with the Reset FEP FEP command.

The Set FEP Options FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Troubleshooting an XL1200 Single-Monitor Color Station

Because the XL1200 single-monitor color station uses the FrameThrower as its only display, and because the FrameThrower is both very flexible and complex, there is a greater possibility of problems in using these stations than in monochrome systems or color stations with two monitors. This section describes some common problems, and steps you can take to solve them before calling Customer Service.

Troubleshooting the Power-up and Initialization of an XL1200 Single-Monitor Color Station

Below, we describe the expected steps that would result in successfully powering on and initializing an XL2100 single-monitor color station to familiarize you with what could go wrong at each step:

1. When power is applied by depressing the on/off button on the front panel, the green POWER light in the on/off button should illuminate and the yellow FAULT light in the reset button should illuminate.

   You should hear the disks spin up; usually there is a "click" (about two seconds after power is applied) followed by a "whirr" that increases in pitch (lasting for about 10 seconds), and finally a "clunk" as the heads are loaded. The FAULT light indicates that the processor is uninitialized at this point. (If the FAULT light comes on later in the process, it indicates that there has been a fatal error. The processor will attempt to re-initialize itself and restart.)

   What can go wrong: If neither the POWER nor FAULT light is illuminated, or you don't hear the disks spin up, you should suspect your power connections.

   What to do: Turn off the system, recheck your power connection, and retry. Note that there is a master power switch on the back of the chassis. This switch must be in the 1 position and should be illuminated (indicating power is available).

2. The booting process is a three-stage process. First, the "Boot" program (in ROM on the processor board) initializes the processor and searches for a "Device" program (in ROM on the I/O board) that can load the "FEP" program from one of the attached peripherals. Second, the Device program examines the attached peripherals looking for a console and a device from which to load the color startup program for the FrameThrower (if the console is a FrameThrower console). Finally, the Device program loads the FEP program from disk or tape and the FEP program will start.

   In the second stage, the Device program must go through a number of steps to determine how to initialize the console:

   a. First it reads the "Switchpack" from the color console unit. This switchpack has two relevant settings (these should be checked by your Customer Service Engineer on installation and rechecked if the system appears to operate incorrectly).

Switch	Function
1	If on, use the color console. If off, the color console is ignored. Normally, it should be on; off is used for diagnostic purposes only
5	If on, use the default monitor type and number: Sony (1024x1280 CADBuffer) 0 If off, a custom monitor type and number is determined by the setting of the FEP Options (as registered by the Set FEP Options FEP command)

      (The remaining switches control the operation of the console unit, setting various diagnostic modes; these should be left as they are.)

      What can go wrong: If the console-unit is not plugged in properly or the switchpack is set improperly, the Device program will not attempt to use the color console.

      What you should do: In early models of the XL1200 single-monitor color station, the console switches are occasionally mis-read. You can try pressing the RESET button or power-cycling the machine, which will cause the Device program to be restarted and the switches to be re-read. You should also check that your console unit is properly plugged in. If it is not plugged in, the Device program will not be able to read the switches; by default it assumes that it should *not* use the color console if it cannot read the switches (this is because using a console with an incorrectly guessed monitor type can physically damage the monitor).

   b. If the switchpack setting indicates that the color console is to be used, the Device program will probe the VMEbus looking for the FrameThrower hardware. (It does not automatically scan the VMEbus, as there may be some other peripheral board in the address space normally occupied by the FrameThrower that should not be randomly written or read.)

What can go wrong: The FrameThrower system may not be found at the expected VME address. The Device program can't use the FrameThrower if it can't find it. The XL1200 processor may not be in VME slot 1. The processor may hang during VME operations if it is not in slot 1.

What you should do: You can power down your system and attempt to re-seat the FrameThrower, XL1200 Processor, and other boards. Note that the XL1200 processor board *must* be in VME slot 1 (the left-most slot as you face the back of the chassis) for it to be able to detect the FrameThrower board. Your Customer Service Engineer can verify the settings of all jumpers and switches on the FrameThower, XL1200 processor, and other boards.

c. If the switchpack setting indicates that the color console is to be used and the FrameThrower hardware is located by the peripherals search, the Device program then looks for the startup program file in the disk label. (Shortly after the "clunk" of the disk heads loading, the Device program is able to use the disk. You should hear a "rattle" or "buzz" as it accesses the disk looking for and loading the startup program. This occurs from 12 to 20 seconds after power-on. Note that if you press the RESET button rather than power-cycling, the disks do not have to spin up or load, hence you may not hear any disk noises.)

If the switchpack indicates the default monitor type is to be used, any startup program found on disk will be accepted and the default monitor type parameters extracted from it and loaded. If the switchpack indicates a custom monitor type, the FEP Options are read from NVRAM and the Device program searches the startup program for the matching type. When an appropriate program is found, default or otherwise, the Framethrower console is initialized (you may see it flash as the sync is loaded) and the Device program greeting is displayed:

```
"Autoloading the IFEP Kernel -- type any character to abort"
```

It takes approximately 21 seconds from the time power is applied until the display is initialized, under normal conditions. If you press the RESET button, the display should initialize in approximately 8 seconds, since the disks are already spun up.

What can go wrong: If the disk label is unreadable, does not have a startup program, or (for custom monitor types) does not have a matching entry, the Device program will not be able to initialize the console.

What you should do: The Device program will look for and load a start-up file from the tape drive if there is a tape in the unit. If you suspect the disk label or a corrupted disk, you can try inserting your IFS tape and pressing the RESET button on the front panel. The tape will spin because of the reset, but it should spin a second time when the Device program searches it for the startup program.

NOTE: In a correctly configured station, the Device program does not need to use the console and will automatically load the FEP from the default disk unit. Thus, you need not consider it a problem if the console appears uninitialized at this point, unless you need to interact with the Device program to request loading of a different FEP or loading from tape. See the section "Using a Spare Monitor to Troubleshoot an XL1200 Single-Monitor Color Station".

You can verify the settings in the disk label by using the Show Disk Label FEP command, and change them by using the Set Disk Label FEP command.

3. If the console has been properly initialized by the Device program, you will see the standard Device program dialog as it initializes the disks and looks for the FEP kernel on one of the disks to load. If the Device program was unable to initialize the console, it will still look for and load the FEP kernel. In either case, you should hear the "rattle" or "buzz" of the disk again as the FEP kernel is loaded. The FEP kernel will go through approximately the same procedures as the Device program in attempting to initialize the console, with the exception that it will attempt to initialize all consoles it can find and then choose one console as the default console.

   When the FEP succeeds in initializing any console it will print out its standard greeting on that console:

   ```
   Type "Hello" to initialize the FEP's command databases, etc.
   ```

   If the console is not the default FEP console, you will see a message to the effect:

   ```
   Type "Set Console FrameThrower Sony console 0" to select this console
   ```

   What can go wrong: As with the Device program, the switches on the console unit may be wrong or mis-read. In this case the FEP may not choose it as the default console, however, unlike the Device program, the FEP initializes all the consoles it can find, so you do have a chance to manually tell it to use a particular console, even if the switches are wrong or mis-read.

   What you should do: If you see the message about "Set Console" on your color monitor, which indicates the FEP decided not to use the color monitor as the default console, you can simply type the suggested command on the color monitor to get the FEP to use it. Note that because the FEP supports command completion, unless you have more than one of a particular type of console it is usually sufficient to only type the first few letters of a console type to select it. For instance, typing "Se C F" and pressing Return is usually sufficient to select the FrameThrower console.

4. Following the standard FEP greeting will be any errors that occurred during initialization and any warnings about boards or peripherals that are not up to date.

   What can go wrong: If you see a message starting "** WARNING **" the FEP has determined that some component of your system is out of date and must be upgraded to work properly. This is unlikely to occur if you have a new system, but if you have had your system upgraded, it may be that some component was missed.

   What you should do: It is important that you have all components updated to the proper revision level for reliable operation. You should contact Symbolics Customer Service to have any components that are out of date upgraded as soon as possible. You run the risk of faulty operation of the system if you proceed with out-of-date components.

5. When the FEP program has been successfully loaded and started and is ready to accept commands from the keyboard, the yellow FAULT light will go out. This takes approximately 37 seconds from the time the system was powered on, or about 24 seconds from the time the system was reset.

   What can go wrong: You might see the yellow FAULT light go off but still have no display on the color monitor. This can be for the same reasons that the Device program could not initialize the console.

   What you should do: If neither the Device program nor the FEP can initialize the console, you should go back and check the suggestions under #2. If none of those suggestions are helpful, it might be that the FEP is unable to determine the type of monitor that is connected to the FrameThrower, either because the switches could not be read or because the switches are set incorrectly. In this case, you can type "blind" to the FEP to tell it the type of monitor.

   NOTE: the Set Monitor Type FEP command is different in the I322 FEP from previous Ivory FEP versions. It takes two arguments now, a console whose type to set and a type. The first argument defaults to the current console, so if you are typing blind you need to type the following commands.

   Whenever you are typing "blind", it is useful to press CLEAR-INPUT and RETURN before you type any commands. Since you cannot see whether there was already some input, this clears any that might be there, so the commands that follow are interpreted from scratch.

```
Press CLEAR-INPUT
Press RETURN
```

Now give a command to ensure that the FrameThrower is selected as the current console. To give the FEP command "Set Console FrameThrower", type the following characters, including pressing the SPACE key and RETURN key where indicated:

```
se SPACE c SPACE f RETURN
```

To give the FEP command "Set Monitor Type *this-console* Sony", type the following characters, including pressing the SPACE key and RETURN key where indicated:

```
s SPACE m SPACE t SPACE SPACE sony RETURN
```

In the command above, note the two SPACES after the letter "T": the first SPACE causes T to be completed to "Type" while giving the Set Monitor Type command. The second SPACE causes the default to be used for the first argument, which is the current console.

What can go wrong: The Device Program makes some simplifying assumptions that may allow it to choose and initialize the color console, but when the FEP is loaded you may find the color console is not initialized. This is due to a mis-setting of NVRAM options on the processor board.

What you should do: The command to enable the color console is:

```
Set FEP Options :Color System Type FrameThrower
```

The minimum you can type (for typing blind) is:

```
s SPACE f SPACE o SPACE :c SPACE s SPACE t SPACE f RETURN
```

After typing this command, either power-cycle or reset the system and the FEP should now initialize the color monitor.

Boot Procedure for XL1200 Single-Monitor Color Stations

When an XL1200 single-monitor color station boots, the following steps happen:

1. Determining which console should be active

   The device PROM queries the hardware to determine which consoles are attached. The possibilities include: a color console, serial console, and a black-and-white console. If more than one console is attached, the device PROM makes a decision about which console should be active. The active console will have the `FEP Command:` prompt. Other attached consoles will display a message about what to type (the Set Console command) to make that console active.

The color console is active if the FEP notices that the FrameThrower board and the color console unit are present. If the color console is not active, then the black-and-white console is active. If the black-and-white console is not attached, then you won't see the `FEP Command:` prompt on any of the attached consoles.

2. Loading the color system startup file for the color console

   The color system startup file contains a set of color system startup programs. A color system startup program is necessary for initializing the color system. If a color console is attached, the device PROM looks at the disk label to find out where the color system startup file is stored. (The information in the disk label is updated by the Edit Disk Label command, and is also updated by the Copy Flod Files command.)

   2A. If the color system startup file is found, it is loaded, and the monitor will be activated. In this case, the next step is 3A.

   2B. If the color system startup file is not found, the XL1200 can still boot, but the monitor will not show any display at this point. In this case, the next step is 3B.

3. Loading the FEP

   3A. The FEP is loaded and it comes up on the color monitor. In this case, the next step is 4A.

   3B. The FEP is loaded, but because the color system startup file was not found, the monitor cannot show any display. In this case, the next step is 4B.

4. Booting Genera

   4A. If the XL1200 is set up to autoboot, it will boot itself. Otherwise, you can give the FEP commands to boot the machine.

   4B. If the XL1200 is set up to autoboot, the machine will boot even though you cannot see anything on the monitor. You can also give FEP commands from the keyboard, and they will work, even though you cannot see them on the monitor.

Using a Spare Monitor to Troubleshoot an XL1200 Single-Monitor Color Station

If you are able to plug a spare black-and-white monitor into your system in place of the color console unit, you may be able to discover what is preventing the color console from being used. Both the device program and the FEP will print out diagnostic information on the black-and-white monitor when searching for and attempting to initialize the color monitor (if the black-and-white monitor is connected in place of the color console unit).

You can then use the FEP to fix the disk label to point to a valid color startup file, you can set the NVRAM options to enable the color console, and you can experiment switching back and forth between the black-and-white and color consoles. Note that both consoles will use the keyboard attached to the black-and-white monitor.

If you are not able to plug in a spare black-and-white monitor, it is still possible you can type "blind" to the FEP to enable the color console, by typing carefully and using command completion. You will know that the FEP is ready to accept commands when the yellow FAULT light on the front panel goes out.

After verifying that the color monitor is enabled (Set FEP Options :Color System Type FrameThrower) and that the proper color startup file is installed in the disk label, you can RESET or power-cycle the system. Note that if you connected a black-and-white monitor, and leave it connected, the FEP will still not initialize the color monitor, because it cannot read the switches from the color console unit. You can cause the FEP to initialize the color monitor anyway, even with the black-and-white monitor by using another NVRAM setting:

```
Set FEP Options :Color System Startup Program Sony :Color System Number 0
```

The FEP will still not choose the color console as the default console, as long as the black-and-white monitor is connected, but you can use the black-and-white monitor to debug your setup and then use the Set Console FEP command to switch to the color console before you start Lisp.

When you are satisfied that everything is working, disconnect the black-and-white monitor, reconnect the color monitor, and reset the system. (If Lisp is running, you can Halt Machine and then use the RESET button to re-initialize the FEP. You can then warm-boot Lisp and it will use the color console).

Using a Serial Terminal to Communicate with the FEP

You can use a serial terminal to communicate with the FEP. One case in which this can be particularly useful is in troubleshooting an XL1200 single-monitor color station. For example, if the color monitor cannot come up, you can connect a serial terminal to the serial port on the color console unit, and use that terminal to give FEP commands such as Show Disk Label, or Set FEP Options.

In addition to connecting the serial terminal physically, you need to give the Set Console FEP command to tell the FEP to use the serial console. You might need to also use the Set Monitor Type FEP command to tell the FEP whether the serial console is ASCII (a dumb terminal) or X3.64 (such as a VT100).

You need to set the serial terminal's parameters for 9600 baud, 8 bits, and no parity.

Keep in mind that serial terminals don't have all the special keys of the Symbolics keyboard. If you need to transmit special Symbolics characters, such as Meta or

Super characters, you need to understand how serial terminal keys are mapped to the Symbolics keys.

Mapping of Serial Terminal Keys to Symbolics Keys

The keyboard of a serial terminal does not have the same set of keys as does the Symbolics keyboard. For example, such a keyboard typically lacks a Meta key, Super key, Hyper key, and Symbol key. These keyboards do, however, have a Control key, however, most such keyboards can handle only Control-A, Control-Z, and a few other characters, most of which are reserved for escapes.

Accessing the Symbolics Character Set

The following characters may be used to access the Symbolics character set:

c-^	= Toggles the Control bit	c-]	= Toggles the Super bit
ESC	= Toggles the Meta bit	c-\	= Toggles the Hyper bit
c-@	= Toggles the Shift bit		

For example, to enter c-m-C, you need to set both the Control bit and the Meta bit, by entering c-^ and ESC; you can then press C to enter c-m-C.

Similarly, you might need to enter c-sh-C. The serial keyboard has both a Control key and a Shift key, but you cannot press them both at once to enter c-sh-C. You can enter c-^ to set the Control bit, then press the Shift key while typing C. Or, you can enter c-@ to set the Shift bit, and then press the Control key while typing C.

Entering Special Symbolics Keys

The character c-_ (that is, the Control key and the underscore _ key) is used as a prefix to enter special characters as follows:

H	= <Help>	L	= <Line>
E	= <End>	P	= <Page>
A	= <Abort>	F	= <Refresh>
S	= <Suspend>	B	= <Back-Space>
R	= <Resume>	N	= <Network>
C	= <Complete>	1	= <Square>
I	= <Clear-Input>	2	= <Circle>
X	= <Escape>	3	= <Triangle>

For example, if you press c-_ followed by H (that is, two keystrokes) on the keyboard of a serial terminal, you get the effect of the HELP key on a Symbolics keyboard.

Entering Symbol Characters

`c-_ _` is the prefix for Symbol characters. (That is, Control Underscore followed by Underscore, two keystrokes.)

For example, you can enter `c-_ _ P` (that is, three keystrokes) to get the effect of `SYMBOL-P`.

`c-_ ?` displays the `c-_` dispatch table.

Overview of Genera 8.0 ECO #1

This is the first ECO to Genera 8.0 and Genera 8.0XL. The purpose of this ECO is to make the following improvements to Genera:

- Provide patches which lay the groundwork for supporting CLIM.
- Improve the integration of CLOS with Flavors.
- Include patches which support the Symbolics XL1200. These changes affect the VME interface on the XL1200 and the XL400. The revised VME documentation is included in this document. See the section "Genera 8.0 XL Documentation Update".
- Provide support for those UX400S customers who want to upgrade to SunOS 4.1. The UNIX software provided with ECO #1 supports SunOS 4.1 and does *not* run with SunOS 4.0. If you have a UX400S and are still running SunOS 4.0 on the Sun, do not load the UNIX software provided with this ECO on your UX400S. Genera and the UNIX software are completely intercompatible between 8.0 and 8.0 ECO #1: you may run Genera 8.0 ECO#1 with the Genera 8.0 UNIX software.
- Fix the problem with MacIvory Ethernet packet transmission that was corrupting packets, causing problems copying worlds and loading files via CHAOSNET.
- Fix the problem with 3600-family cart tape that caused tapes which were not rewound to give hard tape errors.
- Speed up access to the Macintosh file system from MacIvory.
- Fix an IFEP problem with the XL1200 in Genera 8.0XL where certain errors that should have printed in the cold load stream were causing the processor to reset, forcing a warm boot.
- Fix a bug in Statice that could cause databases to become corrupted.
- Fix some other significant bugs in Genera 8.0.

Improvements and Bug Fixes in Genera 8.0 ECO #1

Improved Integration Between CLOS and Flavors

ECO #1 provides a new capability, in which CLOS generic functions can be invoked on Flavors instances. A parameter specializer name in a CLOS method can be the name of a flavor as well as the name of a class. In fact every flavor is now also a class, of metaclass **clos-internals::flavor-class**. This capability is required for CLIM. It also lays the groundwork for better integration between Statice and CLOS, in that CLOS methods can specialize on Statice entity handles, which are Flavors instances.

A number of small improvements in the performance and integration of CLOS have been made:

- The Inspector now works on CLOS instances.
- DW and CLOS interact better.
- Zmacs m-. and CLOS interact better.
- **trace**, **breakon**, and **advise** now work on CLOS generic functions and methods.
- The Find Symbol command can find symbols that are defined as CLOS classes. Show Callers, List Callers (m-X), **who-calls** and **what-files-call** can locate callers that instantiate CLOS classes and that are methods.
- There are many improvements to CLOS performance and correctness, especially for relatively obscure corners of the language used by CLIM.
- Method combination has been completely reimplemented and now supports **:arguments** and gives names to the functions that it generates.
- A function name or function spec for a method is now a method object. Old-style **(method ...)** function specs still work.

Users who wish to take advantage of all of these performance improvements should recompile their code.

New FEP for ECO #1

Changes to the FEP flods include:

- A bug in the the NFEP disk flod has been fixed. The bug was in the disk type specification for the XT8760 disk. If a disk was formatted twice, it usually would fail within a few weeks. The XT8760 disk type has been removed, and two new disk types have been added (XT8760-24 and XT8760-25), to support the old and new configurations of this drive.

- The Set Network-Address command has been moved from the Rel-7 flod to the Lisp flod, and the Rel-7 flod has been deleted.

- Some improvements have been made in the FEP debugger.

- The default world load is now found by searching for the world with latest timestamp.

- The default microcode is that required by the default world.

- The Load World command will print the contents of **sys:*lisp-release-string*** if it is of type array. If not, the release is calculated as was done previously.

- The IFU decode rams are loaded from data in the microcode file if the associated microcode block type is present. If not, they are loaded from FEP generated data as before.

Miscellaneous Improvements and Bug Fixes in ECO #1

- The :Before keyword argument to the Show System Modifications command now assumes that the value you specify is a time in the past. This means that a specification like :Before "Tuesday" will now be correctly interpreted as last Tuesday rather than next Tuesday. The new behavior is consistent with the :Since keyword.

- The undo functions for **si:delete-ie-commands** and **si:add-ie-command** have been fixed.

- In Genera 8.0, **package-name** would sometimes return one of the package nicknames rather than the primary name of the package. This bug has been fixed.

- A patch to the Serial system has improved the performance over sync-link gateways. The performance of interactive traffic over a busy gateway has been greatly improved. Larger packets of information are automatically queued to a pending queue enabling smaller packets to be queued immediately.

- MacIvory serial I/O works much more reliably.

- A bug in the UX400S serial interface has been fixed. The bug caused UNIX authentication credentials to be held across logins.

- RPC Authentication now includes checking UNIX passwords.

- Support for SunOS 4.1 NFS automount features has been added to NFS Client.

- TCP performance has been improved by enabling adaptive TCP retransmission. The default for **tcp::*enable-tcp-adaptive-retransmission*** is now **t**.

- Dialnet has been made more robust in cases where two systems get out of synchronization while exchanging characters. This situation is now detected and the connection is aborted. Previously, the two systems exchanged useless data forever, never timing out because there were always characters available, but never making any progress because because they were out of synchronization.

- Some peculiarities with incorrectly interning certain Dialnet host objects in the wrong namespace have been corrected.

- Graphics line drawing has been improved for XL machines.

- Some storage system bugs that manifest themselves as garbage collection bugs have been fixed.

- Another bug in the storage system has also been fixed. This bug could cause the system to waste some paging space. The amount of lost paging space without this ECO varies, but is larger when Dynamic or In-Place Garbage collection is used, particularly on Ivory systems. On Ivory systems which heavily use Dynamic GC, loading this ECO before the first Dynamic GC avoids wasting as much as 3000 pages of paging space.

- The use of Laserwriter functionality via Appletalk has been enhanced somewhat.

- A new function, **si:fix-fep-dpn** for fixing ECC errors on Ivory FEPs has been provided. Its behavior is exactly like **si:fix-fep-block** except that its arguments are *unit* and *page* (Disk Page Number).

- The mailer no longer blows out while trying to issue a warning if dialnet registries (obsolete in 8.0) exist. Instead, it successfully issues the warning.

- Dumping a LMFS to a cart tape on a UX400S now works correctly. Tapes written before loading this ECO, while they may show correct output when using the [List Backup Tape], were written incorrectly and cannot be restored. Note that this was only a problem if a UX400S cart tape was being used to back up the LMFS partition.

- Bugs involving unreliable operation of cart tapes on NBS machines, particularly timeouts during rewind, have been fixed.

- Support has been added for the MacinStor version 2.1 (Storage Dimensions) disk driver. Ivory disk partitions are now discovered when the Ivory is started, not when the Macintosh is started. This means that you can use the partition editor or reconfigure your disks while the Ivory is shut down and have the changes take effect without restarting the Macintosh.

- The bug that permitted you to delete a MacIvory disk partition that was in active use by the Ivory has been fixed.
- The ethernet performance problem with MacIvory IIci/IIfx has been fixed.
- The bug that caused the partition editor to give the wrong information about the amount of free space on a Macintosh volume has been fixed.
- The bug that caused **:draw-multiple-lines** on MacIvory to report errors has been fixed.

Documentation Updates for Genera 8.0 ECO #1

Clarification on Installing NFS

In Genera 8.1, the NFS system is obsolete and should not be loaded.

Instead, users should load NFS Client, NFS Documentation, and/or NFS Server as needed.

Clarifications on Printers

If you are using a 3600-family machine, you must have the systems RPC, Embedding-Support, and UX-Support loaded before you can use a printer spooled from a UNIX machine.

When installing an LGP2 or LGP3, the `Other Options` field in `Interface Options` should include the following:

```
NUMBER-OF-DATA-BITS 8 PARITY :NONE XON-XOFF-PROTOCOL YES
```

This is particularly important for printers connected to XL-family machines. This setting must be overridden if you are *not* using an Apple LaserWriter or LaserWriter II.

Genera 8.0 XL Documentation Update

This section pertains to XL-family machines only.

Changes to the VMEbus Interface in Genera 8.0 XL

- There are two new slave buffer variables, **sys:*vme-slave-buffer-base*** and **sys:*vme-slave-buffer-end***. They are the starting and ending addresses of the slave buffer in words, not bytes.
- The slave buffer base address is now settable in the FEP via the :Slave-Buffer Base keyword to the Set Boot Options command. This address is in bytes. The

system defaults are correct unless the jumpers on the processor board have been moved.

- To let an application allocate a portion of the slave buffer, use **(sys:allocate-slave-buffer-memory** *:name :words* &key *from-end*). The function returns a starting and ending address in words. There is no enforcement, but this a simple check-out scheme for slave buffer memory so you do not accidentally use memory allocated by the system (as on the UX400) or by another application (like FrameThrower). Specific allocations can be added to an initialization list. System allocations take place on the system initialization list.

- Flonum data tagging is not supported on the XL1200.

- There are two new functions to enable and disable bus interrupts: **sys:enable-bus-interrupt** and **sys:disable-bus-interrupt**. They take a bus interrupt level from 1 to 7. They are somewhat easier to use than **sys:logior-bus-interrupt-mask** and **sys:logand-bus-interrupt-mask**.

- **sys:install-bus-interrupt** has been changed. It now takes a status ID rather than a level. Interrupt functions are dispatched by the status ID rather than the interrupt level. This change is incompatible with Genera 8.0.

- The default slave buffer address for the XL1200 is #xFAC00000.

- **sys:make-bus-address** of an address within the range of the slave buffer addresses will create the appropriate Ivory address to access the slave buffer. This is based on the variables **sys:*vme-slave-buffer-base*** and **sys:*vme-slave-buffer-end***.

- You can initiate VME SYSReset by calling the function **cli::merlin-ii-sysreset**. **Warning:** The effect of using this function is the same as pressing the RESET button on the machine.

- **sys:bus-read** and **sys:bus-write** now update the system bus parameters such as address-modifier and release-mode.

- The slave buffer on the XL1200 processor is 256K words long. Reservations of slave buffer memory are handled by **sys:allocate-slave-buffer-memory**.

- VMEbus errors on the XL1200 are not signalled on writes. Reads receive errors, as do polled reads using **bus-read**.

- The VMEbus interrupt handler does not disable the interrupt level for an incoming interrupt. The user process must disable and reenable the interrupt, if necessary.

- VMEbus interrupt status-ID #x5F on the XL1200 is reserved for Symbolics use. The Slave Trigger Interrupt uses this vector.

VMEbus Interface

Introduction to the XL-Family VMEbus Interface

The XL-family system is based on a 7-slot, 9U form factor VMEbus backplane and card cage. The VMEbus is a versatile, standard 32-bit bus which provides power and clock distribution, asynchronous data transfer, and interrupt delivery and acknowledgment. Although the performance requirements of modern processor/memory interconnects have exceeded the design range of the VMEbus, it remains popular as a peripheral I/O bus, and is often used in tandem with a separate high-speed memory bus. This is the strategy used in the XL: a private 48-bit bus connects the processor with its memory and I/O board, and the VMEbus interface is used to communicate with optional I/O peripherals and other 32-bit wide devices.

The XL processor is a VMEbus master, meaning that it can issue requests to read and write locations in other VMEbus cards, and can deliver interrupts. The interface provides a flexible *polled access* facility with which nearly all possible data transfer operations may be performed. It also provides a *direct-access* facility with which a portion of the VMEbus address space may be mapped into the XL's physical address space, for high-speed access to 32-bit slaves.

The XL processor is also a VMEbus slave, meaning that other bus masters can issue requests to read and write locations in it, and that it can receive interrupts issued by other bus masters. However, other bus masters may not directly access the XL processor's main memory; they may access only a dedicated memory in the XL VMEbus interface called the *slave buffer*. This design eliminates the hardware complication of arbitration deadlock between the VMEbus and the XL private bus, and eliminates the software complication of negotiating with the Genera virtual memory system to reserve contiguous portions of main memory.

The VMEbus is a flexible bus with many options and modes. In the design of the XL VMEbus interface, particular attention was paid to optimizing both the master and slave for high speed 32-bit data transfer, but the interface supports nearly all possible modes and can accommodate virtually any VMEbus device. The interface hardware includes the following features:

- On the XL400, the slave appears on the VMEbus as a 32K by 32-bit memory. On the XL1200, the slave is one megabyte in size.

- The master can transparently map up to 267 megawords of VMEbus address space into the Ivory physical address space (for 32-bit transfers only).

- Both the master and slave implement 8, 16, 24, and 32-bit data transfers, including transfers not aligned on an address boundary.

- Both the master and slave may specify that data be shuffled to compensate for differences in system bit, nibble, or byte ordering.

- Both the master and slave may request that data returned to Ivory be tagged as either integers or IEEE 32-bit floating-point numbers on the XL400. (On the XL1200, data returned can only be tagged as integers.)

- The master may specify an arbitrary address modifier for a data transfer.

- The arbitration parameters (arbitration level, bus release behavior) of the master are programmable.

- The interface can issue and receive all seven interrupt levels, and supports 8-bit interrupters.

- The slave implements the VMEbus block transfer protocol.

- The interface includes a system controller (containing VMEbus arbiter, clock drivers, and so on), which may be disabled by a jumper.

The interface hardware does not support the following features:

- The master does not implement the VMEbus block transfer protocol, but uses address pipelining to achieve equivalent performance.

- The master cannot issue ADDRESS-ONLY data transfer requests.

- The master cannot issue READ-MODIFY-WRITE data transfer requests, but atomic operations are supported by inhibiting bus release.

Software provided with Genera supports efficient access to all interface features, while shielding the client software from irrelevant hardware details. Data transfers may be performed in isolation via function calls that read or write specified bus locations, or by obtaining a physical address that the interface will map to a range of VMEbus addresses, or by using a Lisp indirect array which may be manipulated by normal array operations and facilities such as BITBLT. Full support for delivering and handling interrupts is provided.

For more information about the VMEbus hardware specification, see "The VMEbus Specification", Revision C.1, published by Printex. For more information about the electrical and mechanical characteristics of the XL VMEbus card cage, contact your Symbolics sales representative.

VMEbus Data Transfers

There are four basic techniques for performing VMEbus data transfers:

- Isolated transfers may be performed by calling the functions **sys:bus-read** and **sys:bus-write**, specifying the desired VMEbus address (and perhaps data) and any optional parameters. This technique is the most flexible, since it uses the polled access hardware in the interface which supports non-32-bit transfers.

However, it incurs some overhead programming the hardware and is therefore not very efficient.

- The function **sys:make-bus-address** may be called to map a portion of the VMEbus into the Ivory address space, and return a physical address pointing to it. That address may be manipulated using subprimitives such as **sys:%pointer-plus**, **sys:%p-ldb**, **sys:%p-dpb**, **sys:%block-read**, and **sys:%block-write**. This technique is very efficient, but is rather cumbersome. It also works for 32-bit slaves only.

- The function **sys:make-bus-array** may be called to map a portion of the VMEbus into the Ivory address space, and return an indirect array pointing to it. This allows high-level, bounds-checked access to array elements of any type, and the array may also be passed to Lisp facilities such as **bitblt**. This technique also works for 32-bit slaves only.

- Atomic operations may be performed by calling **sys:bus-store-conditional**, which works for VMEbus locations the same way **store-conditional** works for virtual memory locations.

All these techniques provide some way to configure the interface hardware to enable options such as bit shuffling, nonstandard address modifiers, and arbitration parameters. For polled transfers (via **sys:bus-read** and **sys:bus-write**), the options are specified as simple keyword arguments. For direct transfers (via **sys:make-bus-address** and **sys:make-bus-array**), most of the options are specified by the **sys:with-bus-mode** macro, which must surround any use of VMEbus addresses. See the section "Summary of VMEbus Transfer Options" for a description of the available options.

In general, clients should use polled transfers to refer to isolated registers on the VMEbus, and direct transfers to map memories, frame buffers, large register banks, etc., into the Ivory physical address space. See the section "VMEbus Direct Data Transfers".

VMEbus Direct Data Transfers

The VMEbus master can perform direct data transfers, in which a portion (called a *window*) of the VMEbus address space is mapped into the Ivory physical address space, and accessed as though it were (32-bit wide) Ivory memory. For direct transfers, some of the data transfer options, such as the arbitration parameters, are controlled by hardware registers that must be set up prior to the data transfer. Others, such as data shuffling, are controlled by fields within the Ivory physical address decoded by the VMEbus interface. **sys:with-bus-mode** and the address-generating functions **sys:make-bus-address** and **sys:make-bus-array** conspire to keep the hardware parameters consistent with the client's intent.

sys:with-bus-mode establishes a context within which VMEbus addresses may be generated and used; it is illegal to use a VMEbus address returned by **sys:make-bus-address** or **sys:make-bus-array** outside the dynamic scope of the **sys:with-**

bus-mode in which it was created. **sys:with-bus-mode** programs the VMEbus interface according to the specified options, and guarantees that those parameters will be maintained throughout the dynamic extent of the macro, even if some other process is trying to use the VMEbus simultaneously in a completely different manner.

The first time an address is generated (that is, **sys:make-bus-address** or **sys:make-bus-array** is called) within a given **sys:with-bus-mode**, the direct access window in the VMEbus interface is programmed to encompass the specified addresses. A subsequent attempt to generate an address that doesn't lie within the same 267-megaword window will signal an error. If this restriction causes problems, they can often be resolved by using polled transfers to refer to some of the disparate locations.

Note that **sys:bus-read**, **sys:bus-write**, and **sys:bus-store-conditional** are polled transfers and are therefore not affected by **sys:with-bus-mode**; they may be used at any time.

Summary of VMEbus Transfer Options

The following options may be specified to **sys:bus-read**, **sys:bus-write**, **sys:make-bus-address**, **sys:make-bus-array**, and **sys:with-bus-mode**:

:shuffle
: One of **:none**, **:byte**, **:nibble**, or **:bit**, this specifies the permutation to be applied to the data words received or transmitted by Ivory. **:bit** shuffling reverses the order of all 32 bits. **:byte** shuffling reverses the order of the four 8-bit bytes in a word, but preserves the order within each byte. **:nibble** does the same for 4-bit groups. The default is **:none**.

:data-type
: One of **:fixnum** or **:single-float**, this specifies the tag to be appended to data received by Ivory. **:fixnum** is the default, **:single-float** might be useful when communicating with an array processor or similar device. This is meaningful only for the XL400.

The following options may be specified to **sys:bus-read**, **sys:bus-write**, and **sys:with-bus-mode**:

:address-modifier
: The 6-bit numeric VMEbus address modifier code to be driven onto the bus during a data transfer cycle. The default is #x09, indicating that the address is 32 bits wide, for a data cycle.

:ownership
: One of **:release-when-done**, **:release-on-request**, or **:bus-hog**, this specifies the condition under which the VMEbus interface will relinquish ownership of the bus once it has control. The default is **:release-on-request**.

:arbitration-priority
An integer from 0 to 3, indicating the priority the VMEbus interface will assert when requesting access to the bus. The default is 3.

The following options may be specified to **sys:bus-read** and **sys:bus-write**:

:byte-size
One of 1, 2, 3, or 4, this specifies the number of bytes of VMEbus data. The VMEbus interface will issue an 8, 16, 24, or 32 bit operation as necessary to perform the transfer.

:byte-offset
One of 0, 1, 2, or 3, this specifies the first significant byte of the VMEbus data.

When using the **:byte-size** and **:byte-offset** options, note that all the specified bytes must be contained within an aligned 32-bit word. That is, the size plus the offset must be greater than zero and less than five.

VMEbus Interrupts

Interrupts may be posted on the VMEbus using the function **sys:post-bus-interrupt**, which issues an interrupt at a specified level, waits for the receiver to acknowledge, then delivers the specified status byte to the interrupt handler. Note that the VMEbus interface cannot deliver an interrupt to itself.

The VMEbus interface will interrupt the Ivory processor upon receipt of any VMEbus interrupt for an enabled level. Which levels are enabled is controlled by a mask in the interface hardware, which may be examined and altered using **sys:logior-bus-interrupt-mask** and **sys:logand-bus-interrupt-mask**. The mask contains a 1 in each bit for which an interrupt is enabled; for example, if the mask were #b00001010, interrupts at levels 1 and 3 would be received, and all others would be ignored. Upon receipt of an interrupt request, the VME software issues an interrupt acknowledge cycle to retrieve the status byte, and calls the appropriate client interrupt handler in a Genera simple process.

You can also use **sys:enable-bus-interrupt** and **sys:disable-bus-interrupt** to enable interrupts. Both functions take a level as an argument.

Client software may supply a handler function for a specific status/ID using **sys:install-bus-interrupt-handler**. An interrupt handler function is a normal Lisp function that takes one argument: the status/id byte received during the interrupt acknowledge cycle. The interrupt level is not disabled when the interrupt is received; the programmer must manage this.

VMEbus Slave Interface

The VMEbus interface for the XL400 and Symbolics UX-family contains a 32K by 32 bit memory that appears on the VMEbus as a slave device. The slave supports A24 and A32 address modes, and D08(EO), D16, and D32 data transfers.

The VMEbus interface for the XL1200 has a 256K by 32 bit buffer that supports the same modes.

The XL slave buffer responds to the following VMEbus address modifiers:

Modifier	*Description*
#x39	Standard normal data access
#x3A	Standard normal program access
#x3B	Standard normal block transfer
#x3D	Standard supervisor data access
#x3E	Standard supervisor program access
#x3F	Standard supervisor block transfer
#x09	Extended normal data access
#x0A	Extended normal program access
#x0B	Extended normal block transfer
#x0D	Extended supervisor data access
#x0E	Extended supervisor program access
#x0F	Extended supervisor block transfer

The XL400 slave buffer responds to VMEbus address #xFADC0000 (extended) and #xDC0000 (standard). The XL1200 slave buffer responds to VMEbus address #xFAC00000 (extended) and #xC00000 (standard).

The UX-family machine slave buffer responds to the following VMEbus addresses:

UX400S Board	*Extended Address*
0	#xFADC0000
1	#xFAEC0000
2	#xFAF40000
3	#xFAF80000
4	#xFABC0000
5	#xFA9C0000
6	#xFAAC0000
7	#xFAB40000
8	#xFAB80000

UX1200S Board	*Extended Address*
1	#xFD000000
2	#xFD200000
3	#xFD400000
4	#xFD600000
5	#xFD800000
6	#xFDA00000

7	#xFDC00000
8	#xFDE00000

The slave buffer may be accessed from Ivory using **sys:make-bus-address** or **sys:make-bus-array**, simply by specifying a VMEbus address that falls within the range of the slave buffer. Note that data transfers to such an address don't actually incur any VMEbus traffic; internal data paths are used. The **:shuffle** and **:data-type** options are supported for slave buffer transfers, and work just as they do for normal VMEbus transfers. See the section "Summary of VMEbus Transfer Options".

- The slave buffer address on XL1200 boards can be set via jumpers on the processor board. They are set at the factory to #xFAC00000.

- The mailbox address for each board is at #x100000 beyond the slave-buffer. For example, for UX1200S #1, (+ #xFD000000 #x100000) → #xFD100000

- Lisp keeps track of the location of the slave buffer in the variables **sys:*vme-slave-buffer-base*** and **sys:*vme-slave-buffer-end***. These addresses are in words, so use **(lsh sys:*vme-slave-buffer-base* 2)** to get the VME address of the slave buffer.

If a different VME address is used for the slave buffer, you can inform Lisp of the change by using the keyword :Slave Buffer Address to the Set Boot Options FEP command.

Note: Sections of slave buffer memory are reserved for use by Symbolics for certain hardware configurations. For more information, see the function **sys:allocate-slave-buffer-memory**.

Resetting the XL-Family and Symbolics UX400S VMEbus

The VMEbus SYSreset signal is asserted on the XL backplane shortly after initial powerup, and whenever the RESET button on the front panel is pressed. The XL processor board responds to SYSreset by initializing the Ivory processor and the I/O board, and cold-booting the FEP. The contents of the XL main memory are preserved, and the FEP software should be able to warm boot Genera if it was running prior to the SYSreset.

The XL400 does not generate or respond to the VMEbus SYSfail signal; the XL1200 does generate SYSfail.

Sun systems also assert SYSreset on powerup. The UX-family machine's processor board responds to SYSreset by initializing the Ivory processor and sending a signal to the machine's life support. When the UX-family machine's life support becomes available, it will cooperate with the UX-family machine's processor board in cold-booting the FEP. The contents of UX-family machine's main memory are preserved, and the FEP software should be able to warm boot Genera if it was running prior to SYSreset.

You can initiate SYSreset on an XL1200 by using the function **cli::merlin-ii-sysreset**. This is equivalent to pressing the RESET button on the front panel.

Examples of Using the VMEbus Interface

This section shows several different ways to perform a simple VMEbus data transfer operation in which the goal is to copy a contiguous block of 32-bit words from one VMEbus address to another, reversing the 4 8-bit bytes with each word.

```
;;; Given an A32 D32 slave, use polled transfers to copy each
;;; word.  The bytes are shuffled by the interface hardware as
;;; each word is read from the source.  Simple but slow.
(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  (loop repeat words
        for s from source-bus-address
        for d from destination-bus-address
        do
    (sys:bus-write d (sys:bus-read s :shuffle :byte))))

;;; Given an A32 D16 slave, use polled transfers to copy each
;;; 32-bit word in two halves.  The bytes within each 16-bit word
;;; are shuffled by the interface hardware as each word is read
;;; from the source, but we have to manually interchange the two
;;; halves of each 32-bit word.
(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  (loop repeat words
        for s from source-bus-address
        for d from destination-bus-address
        do
    (let ((v (sys:bus-read s :shuffle :byte :byte-size 2
                           :byte-offset 0)))
      (sys:bus-write d v :byte-size 2 :byte-offset 2))
    (let ((v (sys:bus-read s :shuffle :byte :byte-size 2
                           :byte-offset 2)))
      (sys:bus-write d v :byte-size 2 :byte-offset 0))))
```

```
;;; Given an A16 D8 slave, use polled transfers to copy each
;;; 32-bit word in four separate bytes.  We have to do the byte
;;; swapping manually.  We have to use the :address-modifier
;;; option to specify short (A16) addresses.
(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  ;; This with-bus-mode isn't actually required, we could instead
  ;; specify an :address-modifier option to every bus-read and
  ;; bus-write.  But the options for those operations take their
  ;; defaults from the ambient with-bus-mode, so this is
  ;; syntactically cleaner.
  (sys:with-bus-mode (:address-modifier #x29)
    (loop repeat words
          for s from source-bus-address
          for d from destination-bus-address
          do
      (let ((v (sys:bus-read s :byte-size 1 :byte-offset 0)))
        (sys:bus-write d v :byte-size 1 :byte-offset 3))
      (let ((v (sys:bus-read s :byte-size 1 :byte-offset 1)))
        (sys:bus-write d v :byte-size 1 :byte-offset 2))
      (let ((v (sys:bus-read s :byte-size 1 :byte-offset 2)))
        (sys:bus-write d v :byte-size 1 :byte-offset 1))
      (let ((v (sys:bus-read s :byte-size 1 :byte-offset 3)))
        (sys:bus-write d v :byte-size 1 :byte-offset 0)))))
```

The remaining examples use direct transfers to perform this same operation, and therefore work for D32 slaves only.

```
;;; Map the VMEbus addresses into Lisp arrays, then use a Common
;;; Lisp sequence operator to do the copying. The bytes are
;;; shuffled by the interface hardware as each word is read from
;;; the source. Simple and reasonably efficient for large
;;; transfers, although the setup overhead is fairly high.

(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  ;; with-bus-mode must be wrapped around all uses of
  ;; direct-transfer addresses.
  (sys:with-bus-mode ()
    (stack-let ((s (sys:make-bus-array source-bus-address words
                                       :shuffle :byte))
                (d (sys:make-bus-array destination-bus-address words)))
      (replace d s))))
```

```
;;; Map the VMEbus addresses into physical addresses and use
;;; simple memory subprimitives to do the copying.  Efficient for
;;; short transfers because of the low setup overhead, but low
;;; level and error prone.
(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  ;; with-bus-mode must be wrapped around all uses of
  ;; direct-transfer addresses.
  (sys:with-bus-mode ()
    (loop repeat words
          for s first (sys:make-bus-address source-bus-address words
                                            :shuffle :byte)
                then (sys:%pointer-plus s 1)
          for d first (sys:make-bus-address destination-bus-address words)
                then (sys:%pointer-plus d 1)
          do
      (sys:%memory-write d (sys:%memory-read s)))))

;;; Map the VMEbus addresses into physical addresses and use
;;; block memory operations to do the copying.  This is the most
;;; efficient way to do bulk transfers.
(defun copy-VME-memory-shuffling (source-bus-address
                                  destination-bus-address words)
  ;; with-bus-mode must be wrapped around all uses of
  ;; direct-transfer addresses.  Direct transfers will work for
  ;; A24 and A16 slaves, using the :address-modifier option to
  ;; with-bus-modes as follows.  If we're really trying to be fast
  ;; and don't mind being nasty, we can do the entire transfer
  ;; without ever relinquishing the bus to another master, using
  ;; the :ownership option.
  (sys:with-bus-mode (:address-modifier #x39 :ownership :bus-hog)
    ;; with-block-registers must be wrapped around all uses of
    ;; block registers.
    (sys:with-block-registers (1 2)
      ;; Use block register 1 to address the source
      (setf (sys:%block-register 1)
            (sys:make-bus-address source-bus-address words
                                  :shuffle :byte))
      ;; Use block register 2 to address the destination
      (setf (sys:%block-register 2)
            (sys:make-bus-address destination-bus-address words))
      ;; Use an unrolled loop to copy the words, which makes the
      ;; memory pipeline operate most efficiently.
      (sys:unroll-block-forms (words 4)
        (sys:%block-write 2 (sys:%block-read 1))))))
```

Dictionary of VMEbus Functions

sys:allocate-slave-buffer-memory *name words* &key *:from-end* *Function*

Returns a starting and ending address in words. There is no enforcement, but this a simple check-out scheme for slave buffer memory so you do not accidentally use memory allocated by the system (as on the UX-family machine or by another application (like FrameThrower). Specific allocations can be added to an initialization list. System allocations take place on the system initialization list.

sys:bus-error *Flavor*

This condition is signalled if there is a VMEbus error such as a request timeout. Errors are signalled only on read operations; the XL400 processor stores errors that occur on write operations to be signalled by a future read operation.

sys:bus-read *bus-address* &rest *options* *Function*

Reads the location specified by *bus-address* using a polled transfer. All options default to those specified by the ambient bus mode.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.

sys:bus-store-conditional *bus-address old new* &rest *options* *Function*

Checks to see whether the specified bus location contains *old*, and, if so, stores *new* in that location. The test and set are done as a single atomic operation; no other bus operations are allowed between the two. Both the read and the write are performed using the specified bus options, if any, which default to those specified by the ambient bus mode, if any. **sys:bus-store-conditional** returns **t** if the test succeeded and **nil** if the test failed. See the function **store-conditional**.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.

sys:bus-write *bus-address value* &rest *options* *Function*

Stores the specified *value* into the location specified by *bus-address* using a polled transfer. All options default to those specified by the ambient bus mode.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.

sys:deallocate-slave-buffer-memory *name* *Function*

name represents the portion of the slave buffer that was allocated by **sys:allocate-slave-buffer-memory**.

sys:disable-bus-interrupt *level* *Function*

Disables VMEbus interrupts at *level*. *level* can be between 1 and 7.

sys:enable-bus-interrupt *level* *Function*

Enables VMEbus interrupts at *level*. *level* can be between 1 and 7.

sys:install-bus-interrupt-handler *function status-id* *Function*

Installs *function* as the interrupt handler for interrupts within the specified *status-id*. When an interrupt is detected at that interrupt level, and that interrupt level is enabled in the interrupt mask, *function* will be called with one argument, the status/id byte supplied by the interrupter. The handler will be called in a simple process, and therefore must not depend on the dynamic environment (special variable bindings, catch tags, and so on.

Note that if *function* is redefined, the handler must be installed again for the new definition to take effect.

sys:logand-bus-interrupt-mask *mask* *Function*

Atomically reads the VMEbus interrupt enable mask register, logands it with the *mask* argument, and stores the result back in the register. This function is useful for disabling particular interrupts. It returns the new value, so the current state of the interrupt mask can be read as follows:

```
(sys:logand-bus-interrupt-mask -1)
```

sys:logior-bus-interrupt-mask *mask* *Function*

Atomically reads the VMEbus interrupt enable mask register, logiors it with the *mask* argument, and stores the result back in the register. This function is useful for enabling particular interrupts. It returns the new value, so the current state of the interrupt mask can be read as follows:

```
(sys:logior-bus-interrupt-mask 0)
```

sys:make-bus-address *bus-address size* &rest *options* *Function*

Returns an Ivory physical address usable to access the specified location on the VMEbus. This address is usable within only the ambient **sys:with-bus-mode**. All options default to those specified by the ambient bus mode. An error is signalled if there are any conflicts between the specified options and the hardware configuration specified by the ambient bus mode, or if the desired address range is not supported by the hardware. The first call to **sys:make-bus-address** or **sys:make-bus-array** within a **sys:with-bus-mode** will set up any necessary address window; if a subsequent call specifies an address range outside that window an error will be signalled.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.

sys:make-bus-array *bus-address dimensions* &rest *options* *Function*

Returns an indirect array pointing to the specified VMEbus address, using the direct transfer facility. This array is usable only within the ambient **sys:with-bus-mode**. The options include all the **make-array** options, but note that the array cannot contain arbitrary Lisp objects, only integers and single-precision floating point numbers; see the section "Keyword Options for **make-array**". The options may also include any applicable bus options, which default to those specified by the ambient bus mode. An error is signalled if there are any conflicts between the specified options and the hardware configuration specified by the ambient bus mode, or if the desired address range is not supported by the hardware. The first call to **sys:make-bus-address** or **sys:make-bus-array** within a **sys:with-bus-mode** will set up any necessary address window; if a subsequent call specifies an address range outside that window an error will be signalled.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.

sys:post-bus-interrupt &optional *(level 0) (status 0)* *Function*

Issues an interrupt request for the specified level on the bus, waits for the interrupt acknowledge cycle, then transmits the specified status/id byte to the interrupt handler.

sys:*vme-slave-buffer-base* *Variable*

The starting address for the slave buffer, in words.

sys:*vme-slave-buffer-end* *Variable*

The ending address for the slave buffer, in words.

sys:with-bus-mode *(*&rest *options)* &body *body* *Macro*

Establishes a context within which VMEbus addresses may be generated and used; it is illegal to use a VMEbus address returned by **sys:make-bus-address** or **sys:make-bus-array** outside the dynamic scope of the **sys:with-bus-mode** in which it was created. **sys:with-bus-mode** programs the VMEbus interface according to the specified options, and guarantees that those parameters will be maintained throughout the dynamic extent of the macro, even if some other process is trying to use the VMEbus simultaneously in a completely different manner.

See the section "Summary of VMEbus Transfer Options" for a description of the applicable bus options.