Applied
Microsystems
Corporation

XICE InstallationGuide for Motorola 68000, 68HC000, 68EC000 and 68302 Development Systems for DOS and UNIX Hosts

May 1993
P/N 922-17140-03

IBM XT and IBM AT are trademarks of IBM Corporation

Microsoft and MS-DOS™ are trademarks of Microsoft Corporation.

Microtec is a registered trademarks of Microtec Research, Inc.

SPARC, SPARCstation, Sun, Sun-3, Sun-4, NFS, and PC-NFS are trademarks o Microsystems, Inc.

UNIX is a registered trademark of AT&T.

VALIDATE is a registered tradmark of Applied Microsystems Corporation

Contents

Chapter 3
Installing on a PC

Appendix A
XICE.CFG for the 68000/HC000/EC000/302 Emulators

Appendix B
Troubleshooting

Chapter 1
Introduction

This installation guide explains how to set up the Applied Microsystems XICE debugger.

The XICE debugger software is shipped on its own medium. If you also purchased the Microtec Research language tools or XRAY simulator from Applied Microsystems, they are shipped on separate medium from the XICE debugger.

Note

Use this manual, the *XICE Installation Guide,* for XICE installations. Installation instructions for the language tools and the XRAY simulator are packaged with the compiler/assembler and the XRAY simulator. **Use the *Flexible License Manager Installation Guide* (Sun only) and the installation guides in Debugger Volume 1 (XRAY simulator), MCC (compiler/assembler), or CCC (C++ preprocessor) if you are installing those components.**

The installation processes for the XICE debugger and the language tools are independent and either may be performed first. XICE installation may be performed by the user. Some installation steps for the Sun language tools/simulator and Flexible License Manager are best performed by the system administrator.

Organization of the documentation

This manual covers XICE debugger installation for PCs and compatibles and Sun workstations. This manual is organized as follows:

Chapter 1: Introduction describes the organization of the documentation and provides an overview of the toolchain.

Chapter 2: Installing on a Sun Workstation gives the host requirements and covers the steps for installing the debugger software and for setting up the user environment on a Sun workstation.

Chapter 3: Installing on a PC gives the host requirements and covers the steps for installing the debugger software and for setting up the user environment on a PC.

Appendix A: XICE.CFG gives the information necessary for defining the emulator and debugger default operational states in the xice.cfg file.

Appendix B: Troubleshooting covers common problems that occur during installation and initial use of XICE.

The other toolchain manuals that you may have received are as follows:

- *XICE Supplement*
- *XRAY Documentation Set*
- *MCC Documentation Set*
- *ASM Documentation Set*
- *CCC Documentation Set*
- *Flexible License Manager Installation Guide* (for Sun installations of language tools and simulator)
- The *EL 1600 Hardware Setup and Reference Guide* specific to your emulator

Note

Applied Microsystems provides support for the 68000 microprocessor using two different hardware configurations. One configuration consists of an emulation board, a probe module and a probe tip; the second configuration consists of an emulation board and a probe tip. When there is a need to differentiate between the two versions, the first will be referred to as the probe module/probe tip configuration and the second will be referred to as the probe tip only configuration. See Chapter 1 of the *Hardware Setup and Reference Guide* to identify the configuration of your emulator.

Whether you install software first or configure the hardware is up to you. See Figure 1-1. Typically, you set up the hardware and configure communications, install the software, and configure the software, in that order. Be sure you have carefully followed the setup instructions in the hardware manual and the installation instructions for your software before attempting to use the emulator. When you have completed these tasks, see the *XICE Supplement* for an in-depth tutorial covering XICE.

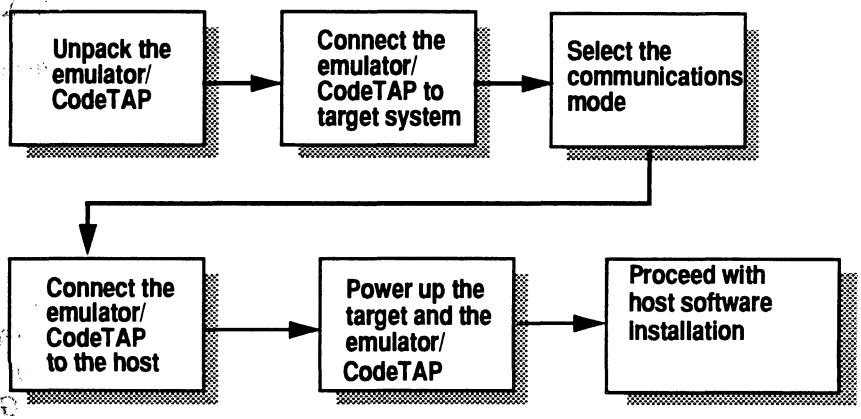

Figure 1-1 System installation steps

Overview of the toolchain

XICE

XICE is an interactive C-source and assembly-level software debugger. This installation manual applies only to the XICE debugger. The product is described in the *XICE Supplement* and the *XRAY Documentation Set.*

XICE is part of a complete embedded development toolchain. Other components of the toolchain that are available from Applied Microsystems are described below.

XRAY

XRAY is a software simulator for pre-integration testing. The product is described in the *XRAY Documentation Set.*

C Cross Compiler

The MCC optimizing C compiler package includes a C compiler, an assembler, linking-loader, and an object module librarian. These products are described in the *MCC Documentation Set* and the *ASM Documentation Set.*

C++ Compiler

The CCC C++ compiler includes C++ preprocessor, C++ translator, and data encapsulation class library capsule. The product is described in the *CCC User's Guide, CCC Reference Manual,* and *C++ Capsule Class Library.*

Chapter 2
Installing on a Sun Workstation

This chapter describes the six basic steps to installing the Applied Microsystems XICE debugger software:

- Verify system requirements
- Set up a parent directory
- Copy software from tape
- Define path and environment variables for each user
- Set up the xice.cfg configuration file
- Invoke XICE to test the installation

The XICE debugger software is shipped on a magnetic tape cartridge. If you also purchased the Microtec Research language tools or simulator from Applied Microsystems, this software is shipped on a separate magnetic tape cartridge, together with the Flexible License Manager software.

Note

Installation instructions for the language tools and the XRAY simulator are packaged with the compiler/assembler and the XRAY simulator. **Use the *Flexible License Manager Installation Guide* and the installation guides in Debugger Volume 1 (XRAY simulator), MCC (compiler/assembler), or CCC (C++ preprocessor) if you are installing those components.** Because root permissions and license installation are required, most installation steps for the language tools/simulator and Flexible License Manager are best performed by the system administrator.

Installation of the XICE debugger is independent of the language tools/simulator and Flexible License Manager, and either may be performed first. XICE installation may be performed by the user.

Step 1: Check minimum requirements

The following are the minimum hardware and software requirements. See the **.doc** file on your distribution media for requirements for the current release.

Requirements for Sun 4 (SPARC) workstations

- Sun 4 (SPARC) series workstation
- Sun OS 4.1.1 or Solaris 2.1
- Sun Openwin or X-Windows
- 13M disk space available
- Ethernet port and cable (for ethernet communications with the EL 1600 emulator)

Compiler/Assembler

This version of XICE is based on the latest version of Microtec Research XRAY68K. It fully supports the output of the 4.3 MCC68K compiler and 6.9 ASM68K assembler. Although it may function properly with output from earlier versions of MCC and ASM, Applied Microsystems cannot guarantee full backwards compatibility.

X-window operation

Like XRAY, Sun-hosted XICE requires an X-windows environment: X11 or Sun Openwindows. For detailed information about the X-windows interface, see Appendix A of the *XRAY User's Guide*.

/home/nps091/xice-el302/v6-40/bin

Step 2: Set up the directory structure

You may install the software on any drive with sufficient space available and use any name for the parent directory you choose. The recommended directory name is **amc_sun_tools**; this will be used as the default directory in this chapter. The sub-directory structure resulting from the installation process is **xice-*prod*/v*x-x***, where *prod* is an identifier associated with the emulation tool that XICE will drive and where *x-x* is the version number of XICE.

If it doesn't already exist, create the parent directory for the software; then change to the newly created directory. For example, put Sun-hosted software from Applied Microsystems in the parent directory **/amc_sun_tools**; this prevents possible future file overwriting if you have a combined Sun and PC network. The commands to do so would be:

mkdir amc_sun_tools
cd amc_sun_tools

When you **tar** the software from the tape in the next step, the appropriate version-controlled **xice** subdirectories will be created as part of the **tar** process. For example, a structure similar to the following is created under the **/amc_sun_tools** parent:

```
/amc_sun_tools
    |
    /xice-el.68k
        |
        /v6-40
            |
            /bin        /demo
```

The specific default directory name for the supported emulation devices are as follows:

Emulation Device	Default Directory
EL 1600 68000/68HC000/ 68EC000 (68000 - probe tip only configuration)	/amc_sun_tools/xice-el.68k/*vx-x*
EL 1600 68000/68302 (68000 - probe module/probe tip configuration)	/amc_sun_tools/xice-el.302/*vx-x*

Identification of the directory by version number prevents overwriting an existing version that you may still want to access after you install an update or upgrade. Applied Microsystems generally recommends that only one version of the software be present on the drive at any time. If you choose to keep multiple versions available, be sure to change the paths contained in any user files (includes, startup, etc.) and to modify your path statement and environment variables as explained later in this chapter. In addition, be sure to check your compiler and XICE release notes for any changes in compatibility.

Step 3: Copy the software from the tape

Copy the software into the directory you created by typing:

tar xvf /dev/*drive* (typically rst0 or rst8 for SPARCstations)

Next, type **ls** to display the information copied onto the host. The file **file.lis** provides a listing of the directories and files, and gives a one-line description of each file.

The **doc** directory contains a **.doc** file for XICE. Be certain to read this file prior to attempting to work with the debugger. It describes the enhancements, corrections, and restrictions associated with the current release of the software.

The **cdemon** demonstration program is included on the distribution cartridge. It can be used for the tutorial in the *XICE Supplement* if you have overlay RAM, an Applied Microsystems demonstrator module, or RAM available on your target.

Step 4: Define individual user's setup

In order to use the debugger from other directories, each user must edit his or her **.cshrc** or **.login** file to change the path information and to add the environment variables.

Setting the path

You may set the path at the beginning of each session or add the path to your **.login** or **.cshrc** file.

For example, to set the path for the current working session, you enter the following **csh** command if you placed the software in the parent directory **/amc_sun_tools/xice-el.68k/v6-40:**

set path = (/amc_sun_tools/xice-el.68k/v6-40/bin $path)

You can permanently append the new directories to your **.login** or **.cshrc** file. Use the full directory name.

Setting up environment variables

Each user must define the AMCEMUL environment variable and may also define some or all of the following environment variables:

- XRAY is used for locating user source and object files that do not reside in the current directory.

- XRAYLIB is used for locating the font files, the help files, and **xice.cfg**. It also specifies the location of the emulation configuration file, **emulcfg.dat.**
- XHS68KLIB replaces the default directory when locating simulator font files, help files, and start-up files. Use it if you run both XICE and XRAY, using XRAYLIB for XICE and XHS68KLIB for XRAY.
- XOS_OFF disables C++ features.
- AMCEMUL (required) points to the appropriate emulation device definition in the **xice.cfg** file.

The following examples show how you might set the environment variables. Note that the directory paths specified are only examples and may not be available on your system; you must select directories available on your system.

For example, if you selected the EL68302 emulator setup from the **xice.cfg** file and if you placed XICE in **/amc_sun_tools/xice-*prod*/v*x-x***, target code in **/project/new_code**, simulator start-up files in **/joe/work/xray/bin**, and XICE configuration and start-up files in **/joe/work/xice/bin**, the **csh** commands to set these variables might be as follows:

```
setenv AMCEMUL EL68302
setenv XRAY "@:/project/new_code"
setenv XRAYLIB /amc_sun_tools/xice-prod/vx-x/
bin:/joe/work/xice/bin
setenv XHS68KLIB /amc_sun_tools/xray68k/bin:
/joe/work/xray/bin
```

Note

- Appendix A describes how to set AMCEMUL.
- Environment variable names must be typed in all CAPITAL LETTERS.
- If you use XRAY to specify multiple paths, you must separate each path with a colon.
- The special pathname, @, used with XRAY allows searching from the path saved by the compiler in the absolute file.

When AMCEMUL is not set

If you do not define AMCEMUL, or if it does not match an emulator description in the **xice.cfg** file, then XICE reports an error and does not let you proceed. Appendix A in this manual explains the options you have for the value of AMCEMUL.

If you use one of the default emulator descriptions, you would use one the following:

For UNIX systems using C shell (**csh**):

setenv AMCEMUL EL68000 or
setenv AMCEMUL EL68HC000 or
setenv AMCEMUL EL68302

For UNIX systems using Bourne shell (**sh**):

set AMCEMUL=EL68000; export AMCEMUL or
set AMCEMUL=EL68HC000; export AMCEMUL or
set AMCEMUL=EL68302; export AMCEMUL

Locating user files (XRAY)

Use the **XRAY** environment variable to specify the paths (directories) that XICE and XRAY should use to search for absolute object files or high-level source files that are not found in the current directory.

When looking for high-level source files, place the @ character in your **XRAY** path to indicate that the XICE debugger and the XRAY simulator should use the path, if any, contained in the absolute file. The MCC compiler supplies this path to indicate where the source file was compiled if the **-Gf** compiler option is specified.

XICE searches for user files (object and source) to be debugged in the following order:

1. The directory containing the file loaded with the **load** command. For example, if a file were loaded with **load /SW/boot/src/binit.x**, the **/SW/boot/src** directory would be searched.

2. The current directory.
3. The directory specified by the XRAY environment variable.
4. The default installation directory.

Locating program files (XRAYLIB)

The debugger searches for all the files it needs for execution (including the help files, the XICE configuration file, the emulation configuration files, and font files) in the following order:

1. The current directory.
2. The directory specified by the environment variable XRAYLIB.
3. The path specified by USR_MRI (if defined)
4. The directories specified in the search path.

The following examples show how you can set the **XRAYLIB** environment variable. Use a colon as a a delimiter to separate multiple paths. Note that the directory paths specified are only examples and may not be available on your system; you must select directories available on your system.

For UNIX systems using C shell (**csh**):

setenv XRAYLIB `/amc_sun_tools/xice-el.68k/v6-40/bin:/joe/work/xice/bin`

For UNIX systems using Bourne shell (**sh**):

XRAYLIB `=/amc_sun_tools/xice-el.68k/v6-40/bin:/joe/work/xice/bin;` **export XRAYLIB**

When XHS68KLIB is used

If you plan to switch between XICE and XRAY frequently, use the XHS68KLIB environment variable to replace the default directory for XRAY font files, help files, and start-up files. XICE uses XRAYLIB only.

XRAY searches for font files, help files, and start-up files in the following order:

1. The current directory
2. The path specified by **XHS68KLIB** (if defined)
3. The path specified by **XRAYLIB** (if defined)
4. The path specified by **USR_MRI** (if defined)
5. The default installation directory

The following examples show how the **XHS68KLIB** environment variable can be set. Note that the directory paths specified are only examples and may not be available on your system; you must select directories available on your system.

For UNIX systems using C shell (**csh**):

setenv XHS68KLIB /amc_sun_tools/xray68k/bin

For UNIX systems using Bourne shell (**sh**):

XHS68KLIB=/amc_sun_tools/xray68k/bin; export XRAY68KLIB

When XOS_OFF is TRUE

When this variable is set, C++ features can be disabled:

Using C shell (csh):

setenv XOS_OFF TRUE

Using Bourne Shell (sh)

XOS_OFF=TRUE
export XOS_OFF

Appendix G of the *XRAY Reference Manual* covers C++ enhancements.

Note

Both XICE and XRAY can generate a start-up file with the same default name: **startup.xry.** If you plan to use start-up files with both XICE and XRAY, you should create files with distinct names so that the appropriate file is accessed at invocation.

Step 5: Modify the xice.cfg file

The information in the file **xice.cfg** configures the XICE debugger software for the emulator you are using, your target processor, and your method of communications. Appendix A in this manual describes **xice.cfg** and explains how to modify **xice.cfg** for your particular needs.

Note

Before modifying the default **xice.cfg**, save a backup copy in case you need to return to the default settings in the future.

If this is your first use of XICE, you may wish to use one of the default emulator settings until you have determined what specific settings to use with your target. In this case, you change only two items:

1. Edit the PORT definition to change the emulator name to the one you selected during hardware/communications setup.
2. Comment out the OVERLAY definition in the EMULATOR section if you have no overlay memory or do not wish to configure it at invocation (indent the comment symbols to at least column two of the text).

Please turn to Appendix A now and follow the instructions to make any necessary modifications to **xice.cfg**; then return to Step 6.

Step 6: Start and test the debugger

XICE allows you to debug software using either your target system or a null target, if one exists for your emulation tool. If you haven't already done so, connect the emulator to a target and configure Ethernet communications as described in the emulator installation manual.

XICE command

For each XICE, there is a unique executable appropriate to the processor you are emulating and the type of emulation system you have purchased. Each executable may be followed by a number of parameters.

For example, the XICE68K debugger for the 68HC000 processor can be started using the following syntax:

vxhc000 [-ni] [-b] [*abs_file*] **[-i** *include_file*][**-s** *setup_file*] **[-l** *log_file*] **[-j** *journal_file*] **[-e boot]**

The following table lists the executables by processor and emulation system. Parameters are explained in the "Debugger Invocation Parameters" later in this section.

Processor	Emulation Type	Executable
68000 (probe module/probe tip configuration)	EL 1600	vxel000
68000/68HC000/68EC000 (68000 - probe tip only configuration)	EL 1600	vxhc000
68302	EL 1600	vxel302

The absolute file, *abs_file*, is the name of the absolute object module that you wish to load into target memory. You must have a working or null target connected or be in null target mode before starting XICE.

Note

Only one absolute object module may be loaded into XICE from the UNIX command line. If you need to load multiple absolute object modules, use the LOAD command after starting the debugger. If the filename specified on the command line is a file written in assembly, XICE will enter assembly-level mode and will not be able to enter high-level mode during a debugging session.

Testing XICE installation

To test XICE, you need to have your emulator fully set up and connected to a working target system, a null target, or one of Applied's optional demonstrator modules (if available for your system). If you need to set up the hardware, see the hardware setup and reference guide appropriate to your emulator. You must also have the source files in either the local directory or the same directory as the absolute file.

The simplest way to test XICE is to invoke it using the `-e boot` parameters.

For example, to invoke and test XICE for 68302, enter:

vxel302 -e boot &

See the listing above for the executable name for your system.

The **-e boot** parameter forces download of the emulator operating system (the *xxx.shl* file). As the debugger comes up, it displays messages on the screen regarding initializing the system and downloading the operating system. The **&** puts the program in background, making it easier to "kill" should the need arise. Once the debugger prompt appears in the command viewport, you may assume that XICE is correctly loaded and available to users.

If XICE fails to load or operates erratically, see Appendix B for troubleshooting tips.

Note 

If you use other Applied Microsystems software or Applied-compatible debuggers with your emulator, you should use the **-e boot** option to ensure that XICE loads the appropriate shell code each time you use XICE after using another debugger. If XICE detects mismatched shell code, it warns you during boot up but does not abort.

To quit XICE, type:

QUIT Y (or **Q Y**)

Debugger invocation parameters

XICE may be invoked with any of the parameters described in the table below. These allow you to tailor each session as you want it.

From the UNIX command line, you specify these parameters using a hyphen (-). (INCLUDE, JOURNAL, and LOG can also be invoked from the XICE command line.)

For example, to invoke XICE for 68302 from the UNIX command line and load only the symbols from a named file, you would enter:

vxel302 -ni *file_name*

See the table above for the executable for your system.

Note 

Both XICE and XRAY can generate a start-up file with the same default name: **startup.xry.** If you plan to use start-up files with both XICE and XRAY, you should create files with distinct names so that the appropriate file is accessed at invocation.

The following table explains each of the available parameters.

Table 2-1 Invocation parameters

PARAMETER	DEFINITION
-ni *filename*	The "no image" option loads the symbols only.
-b	The "brief" option takes all command input from the include file specified by **-i** instead of from the keyboard. There is no screen output, and any error will cause the program to halt. (If the include file specifies "error=cont" then execution continues after an error.) If you wish to capture output, use **-j**.

Table 2-1 Invocation parameters

PARAMETER	DEFINITION
abs_file	The name of an absolute object module that is loaded into the debugger's simulated target memory. The default extension is **.x** (UNIX) or **.abs** (DOS). A working or null target must be connected before starting XICE. Only one absolute object module may be loaded into XICE on the command line. If you need to load multiple absolute object modules, use the LOAD command after starting the debugger. If the filename specified on the command line is a file written in assembly, XICE will enter assembly-level mode and will not be able to enter high-level mode during the debugging session.
-i *include_ filename*	The name of an include file that is read before any debugger commands are entered. The default extension is **.inc**.
-s *startup_ filename*	The name of a startup file containing startup parameter definitions. The default extension is **.xry**. If a startup file exists and you start XICE with -e boot OR with -s <filename> and -e boot, the -e boot is ignored and shell code is not reloaded; shell code would only be loaded if the startup file included the -e boot parameter.
-j *journal_ file*	Command output and viewport information is saved in the specified *journal_file*. The default extension is **.jou**.
-l log_ file	User commands and a record of any errors are placed in *log_file*. The default extension is **.log**.
-e boot	The boot option forces the emulator to reboot. This XICE emulator-only option downloads new emulator control code. Note that this option takes several seconds to complete and is not necessary each time you start the debugger. If a startup file exists and you start XICE with -e boot OR with -s <filename> and -e boot, the -e boot is ignored and shell code is not reloaded; shell code would only be loaded if the startup file included the -e boot parameter. The STARTUP command is used to save startup parameters to a file and will save the -e option if it was used to invoke XICE. This will result in loading emulator control code each time you start the debugger. If you wish to start XICE without reloading emulator control code each time, enter the STARTUP command only after starting XICE without the -e boot option.

Chapter 3
Installing on a PC

This chapter describes the five basic steps to installing the Applied Microsystems XICE debugger software:

- Verify that your system meets minimum requirements.
- Run the INSTALL program.
- Perform individual user's setup.
  - — Set up directory paths
  - — Set up environment variables
  - — Modify AUTOEXEC.BAT and CONFIG.SYS files
- Set up the XICE.CFG configuration file.
- Invoke XICE to test the installation.

If you also purchased the Microtec Research language tools or simulator from Applied Microsystems, the language tools and simulator software is shipped on separate media from the XICE debugger.

Note

Installation instructions for the language tools and the XRAY simulator are packaged with the compiler/assembler and the XRAY simulator. **Use the installation guides in Debugger Volume 1 (XRAY simulator), MCC (compiler/assembler), or CCC (C++ preprocessor) if you are installing those components.**

Step 1: Verify system requirements

The following are the minimum general requirements. See the **.DOC** file on your distribution media for the requirements for the current release.

Requirements for PC or compatible

- IBM PC or compatible (with 386 or better microprocessor)
- MS-DOS 5.0 or later operating system
- At least 4 MB RAM, with 550K free conventional memory and 2 MB free extended memory available
- Hard disk drive with 7M disk space available
- Monochrome or color monitor
- Serial port and cable set up as either COM1 or COM2 (for serial communications with the emulator)
- Optional Future Domain TMC-850M SCSI host adapter and cable (for HSP communications with the emulator)
- Optional Avalan REMOTELY POSSIBLE local area network remote control software

Compiler/Assembler

This version of XICE is based on the latest version of Microtec Research XRAY68K. It fully supports the output of the 4.3 MCC68K compiler and 6.9 ASM68K assembler. Although it may function properly with output from earlier versions of MCC and ASM, Applied Microsystems cannot guarantee full backwards compatibility.

DOS memory extender

XICE for the PC uses the Phar Lap DOS Extender. This software is loaded automatically as part of the install program and its operation is transparent with regard to XICE. The DOS extender software works with QEMM, HIMEM.SYS, or with no memory manager at all. It is also compatible with the PC-NFS drivers, whether those drivers are loaded low or high.

The Phar Lap DOS Extender has been tested with DOS 3.31 and QEMM 5.11, with DOS 5.0, and with DOS 5.0 and QEMM 6.0. The following two exceptions are the only compatibility problems that were found.

- QEMM VIDRAM PROGRAM. VIDRAM can be loaded in a standby (off) mode and then turned on and off from the command line. The DOS Extender works fine as long as VIDRAM is left off. When you turn on VIDRAM, however, it does not work, and when you turn it off again, it locks up the system.
- NOEMS PARAMETER. If you use either QEMM or EMM386 with the parameter NOEMS, the system will display a message that the CPU is already in Virtual 86 mode and that you must remove whatever program put it into that mode.

LAN remote control software

If you have purchased the optional Avalan remote control LAN software, you should install it first on your network. See the Avalan manual supplied with your REMOTELY POSSIBLE software. Using REMOTELY POSSIBLE, you should be able to run XICE from your PC and access any emulator on the network.

Note

Applied Microsystems has successfully used REMOTELY POSSIBLE with our emulators. However, Applied Microsystems neither warrants nor supports Avalan's product. If you encounter problems installing or using REMOTELY POSSIBLE, please contact Avalan, not Applied Microsystems.

Step 2: Run the install program

The software is shipped on floppy disks and includes an installation program disk.

You may install the software on any drive with sufficient space available and use any name for the main directory you choose. The defaults are drive C and \AMCTOOLS\XICE-*prod*\V*x-x*, where *prod* is an identifier associated with the emulation tool that XICE will drive and where *x-x* is the version number of XICE. The specific default directory names for the supported emulation devices are as follows:

Emulation Device	Default Directory
EL 1600 68000/68HC000/ 68EC000 (68000 -probe tip only configuration)	\AMCTOOLS\XICE-EL.68K\V*x-x*
EL 1600 68000/68302 (68000 - probe module/probe tip configuration)	\AMCTOOLS\XICE-EL.302\V*x-x*

Identification of the directory by version number prevents overwriting an existing version that you may still want to access after you install an update or upgrade. Applied Microsystems generally recommends that only one version of the software be present on the drive at any time. If you choose to keep multiple versions available, be sure to change the paths contained in any user files (includes, startup, etc.) and to modify your path statement and environment variables as explained later in this chapter. In addition, be sure to check your compiler and XICE release notes for any changes in compatibility.

Note

Before installation, make backup copies of the disks using the DOS DISKCOPY command. The program files are compressed and stored as a single file on the disks.

To install the XICE software, complete the following steps:

1. Insert the installation program disk into drive A. Make drive A the current drive by typing:

   ```
   A:
   ```

2. Start the installation program by typing:

   ```
   INSTALL
   ```

3. Follow the instructions that the INSTALL program presents on-screen. The INSTALL program prompts you to:
   - specify the name of the drive on which you want to install the files
   - specify the directory in which to install the files

Note

If the distribution directory already exists, the installation program displays a warning that the files in the distribution directory already exist. You will be prompted to enter one of four options (y, n, a, r):

y	the file is overwritten
n	the file is not overwritten
a	the current file and all other files will be overwritten with the versions being installed, without displaying a warning message
r	the file to be installed will be renamed and you will be prompted for the new name

We recommend that you choose "y."

4. The INSTALL program then reads the necessary files from the distribution disks and writes them to the directory you specified. You are prompted to change disks when necessary. You will be asked to insert the **last** disk first. If there is only one disk in the distribution, simply enter a carriage return; you do not need to re-insert the disk. You may enter Ctrl-C at any prompt to abort the installation.
5. When the copying process completes (i.e., when the PC finishes extracting the last file on the last disk), the DOS prompt will be displayed.

The CDEMON demonstration program is included on the distribution disks and is installed by the INSTALL program. It can be used for the tutorial in the *XICE Supplement* if you have sufficient RAM available on your target or in overlay, or if you are using the Applied Microsystems demonstrator module.

Your software includes a file called **FILE.LIS**, which describes briefly the function of each file. There is also a **.DOC** file for XICE, which you can find in the **DOC** directory. It is very important that you read the **.DOC** file, since it updates the information in this manual and provide other useful information regarding the software.

Step 3: Define individual user's setup

Each user needs to modify **AUTOEXEC.BAT** to incorporate the path and environment information for the debugger. Each user should also modify the **CONFIG.SYS** file to increase the files and buffers sizes.

Note

The PC must be rebooted for modifications to **CONFIG.SYS** and **AUTOEXEC.BAT** files to take effect.

Setting the directory PATH

Incorporating the software into your path is necessary in order for you to use the software from other directories. The software uses the **PATH** variable to find its own required files, such as the help file and configuration database.

To display the current path, type **PATH** at the DOS prompt. To set a new path, type in the full path at the DOS prompt adding to it the information for the debugger and language tools software. For example, if the **PATH** command reports your path as:

PATH = C:\;C:\DOS

you would type the following, if the software is in C:\AMCTOOLS\XICE-EL.68K\V6-40:

PATH = C:\;C:\DOS;C:\AMCTOOLS\XICE-EL.68K\V6-40\BIN

Note

Your path should be based on how the files are organized on *your* PC.

Once set, this path string becomes part of the DOS environment and instructs DOS to look first in the current directory (it does this automatically), then in the directories specified in the path.

To permanently set the path information, you must modify the path statement in the **AUTOEXEC.BAT** file. Otherwise, you must use the **PATH** command each time you restart DOS.

Setting up the environment variables

Each user must define the AMCEMUL environment variable and may also define some or all of the following environment variables.

Use the environment variables to specify the search paths for the debugger files as you installed them on your system. Put the appropriate variables in your **AUTOEXEC.BAT** file:

- AMCEMUL (required) points to the appropriate emulation device definition in the **xice.cfg** file.
- XRAY is used for locating user source and object files that do not reside in the current directory.
- XRAYLIB is used for locating the help files, and **XICE.CFG**. XRAYLIB also specifies the location of the emulation configuration file, **EMULCFG.DAT**.
- XHS68KLIB replaces the default directory when locating simulator font files, help files, and start-up files. Use it if you run both XICE and XRAY, using the XRAYLIB for XICE and XHS68KLIB for XRAY.
- XOS_OFF disables C++ features.
- XRAYTMP specifies where XICE should put its temporary files.

For example, if you selected the EL68302 emulator setup from the **xice.cfg** file, the debugger is installed in C:\AMCTOOLS\XICE-EL.68K\V6-40, the target code resides in C:\PROJECT\NEW_CODE, temporary files go to C:\TEMP, the XICE and XRAY configuration files reside in C:\JOE\WORK, the commands to set these variables would be as follows:

```
SET AMCEMUL=EL68302
SET XRAY=C:\PROJECT\NEW_CODE\V6-40;@
SET XRAYLIB=C:\AMCTOOLS\XICE-EL.68K\V6-40\
BIN;\JOE\WORK\XICE\BIN
SET XHS68KLIB=C:\AMCTOOLS\XRAY68K\V2-3\BIN;
\JOE\WORK\XRAY\BIN
SET XRAYTMP=C:\TEMP
```

Note

- Appendix A describes how to set AMCEMUL.
- You can set only one environment name each time you use the SET command.
- If you use XRAY to specify multiple paths, you must separate each path with a semicolon.
- The special pathname, @, used with XRAY allows searching from the path saved by the compiler in the absolute file.

When AMCEMUL is not set

This environment variable must be set to use XICE. If you do not define AMCEMUL, or if it does not match an emulator description in the **xice.cfg** file, XICE reports an error and does not allow you to proceed. Appendix A in this manual explains the options you have for the value of AMCEMUL.

If you use one of the default emulator descriptions, you would use one the following:

SET AMCEMUL=EL68000 or
SET AMCEMUL=EL68HC000 or
SET AMCEMUL=EL68302

Locating user files (XRAY)

Use the **XRAY** environment variable to specify the paths (directories) that XICE and XRAY should use to search for absolute object files or high-level source files that are not found in the current directory.

When looking for high-level source files, place the @ character in your **XRAY** path to indicate that the XICE debugger and the XRAY simulator should use the path, if any, contained in the absolute file. The MCC compiler supplies this path to indicate where the source file was compiled if the **-Gf** compiler option is specified.

XICE searches for user files (object and source) to be debugged in the following order:

1. The directory containing the file loaded with the **load** command. For example, if a file were loaded with:
   load \SW\boot\src\binit.abs
   the **\SW\boot\src** directory would be searched.
2. The current directory.
3. The directory specified by the XRAY environment variable.
4. The default installation directory.

Locating program files

The debugger searches for all the files it needs for execution (including the help files, the XICE configuration file, and the emulation configuration file) in the following order:

1. The current directory.
2. The directory specified by the environment variable XRAYLIB.
3. The directory specified in the search path:
   \AMCTOOLS\XICE-*prod*\V*x-x*\BIN

When XHS68KLIB is used

If you plan to switch between XICE and XRAY frequently, use the XHS68KLIB environment variable to replace the default directory for XRAY font files, help files, and start-up files. XICE uses XRAYLIB only.

XRAY searches for font files, help files, and start-up files in the following order:

1. The current directory
2. The path specified by **XHS68KLIB** (if defined)
3. The path specified by **XRAYLIB** (if defined)
4. The path specified by **USR_MRI** (if defined)
5. The default installation directory.

The following example shows how the **XHS68KLIB** environment variable can be set. Note that the directory paths specified are only examples and may not be available on your system; you must select directories available on your system.

SET XHS68KLIB=C:\AMC_SUN_TOOLS\XRAY68K\ V2-3\BIN;\JOE\WORK\XRAY\BIN

When XOS_OFF is TRUE

When XOS_OFF is set, C++ features are disabled.

Using DOS:

set XOS_OFF=TRUE

Appendix G of the *XRAY Reference Manual* covers C++ enhancements.

Note

Both XICE and XRAY can generate a start-up file with the same default name: **startup.xry.** If you plan to use start-up files with both XICE and XRAY, you should create files with distinct names so that the appropriate file is accessed at invocation.

Modifying CONFIG.SYS

Before using the debugger or language tools, each user must edit the system configuration file **CONFIG.SYS** to include an entry for FILES and for BUFFERS.

These variables should be set to 15 or higher. For example:

FILES=20
BUFFERS=15

In some cases they may have to be set higher than the values indicated above, especially if you use many include files. Keep in mind, however, that increasing the files and buffers values reduces available memory.

In some cases an "out of environment space" message may be displayed. The CONFIG.SYS file may need to be modified to provide more environment space. For example, the following line could be included in the file:

SHELL=C:\DOS\COMMAND.COM C:\DOS\ /E:1500 /P

Please refer to your DOS user's guide for more information on environment space.

Always reboot your system after modifying the **CONFIG.SYS** and **AUTOEXEC.BAT** files.

Step 4: Modify the xice.cfg file

The information in the file **xice.cfg** configures the XICE software for the emulator you are using, your target processor, and your method of communications. Appendix A in this manual describes **xice.cfg** and explains how to modify **xice.cfg** for your particular needs.

Note

Before modifying the default **xice.cfg**, save a backup copy in case you need to return to the default settings in the future.

If this is your first use of XICE, you may wish to use one of the default emulator settings until you have determined what specific settings to use with your target. In this case, you change only the following items:

1. Edit the PORT declaration of the EMULATOR section to the appropriate RS-232 or HSP selection, as described in Appendix A.
2. If you are using HSP communications, edit the DEVICE and PHYDEVID fields of the HSP PORTS section to match your host and emulator configurations.
3. If you are using RS-232 communications, edit the PORT and BAUD fields of the RS-232 PORTS section to match your host configuration.
4. Comment out the OVERLAY declaration of the EMULATOR section if you have no overlay memory or do not wish to configure it at invocation (indent the comment symbols to at least column two of the text.).

Please turn to Appendix A now and use this information to make any necessary modifications to **xice.cfg**, then return to the next section.

Step 5: Start and test the debugger

XICE allows you to debug software using either your target system or a null target, if one exists for your emulation tool. During the pre-integration phases of your development work, the XRAY simulator allows you to debug software without being plugged into a target or the emulator's null target.

XICE command

For each XICE, there is a unique executable appropriate to the processor you are emulating and the type of emulation system you have purchased. Each executable may be followed by a number of parameters.

For example, the XICE68K debugger for the 68HC000 processor and EL 1600 emulation system can be started using the following syntax:

vxhc000 [-ni] [-b] [*abs_file*] **[-i** *include_file*][**-s** *setup_file*] **[-l** *log_file*] **[-j** *journal_file*] **[-e boot]**

The following table lists the executables by processor, emulation system, and communications mode. Parameters are described in "Debugger Invocation Parameters" later in this section.

Processor	Emulation Type	Comm Mode	Executable
68000 (probe module/ probe tip configuration)	EL 1600	RS-232 and HSP	vxel000.exe
68000/68HC000/ 68EC000 (68000 - probe tip only configuration	EL 1600	RS-232 and HSP	vxhc000.exe
68302	EL1600	RS-232 and HSP	vxel302.exe

The absolute file, *abs_file*, is the name of the absolute object module that you wish to load into target memory. You must have a working or null target connected or be in null target mode before starting XICE.

Note

Only one absolute object module may be loaded into XICE from the DOS command line. If you need to load multiple absolute object modules, use the LOAD command after starting the debugger. If the filename specified on the command line is a file written in assembly, XICE will enter assembly-level mode and will not be able to enter high-level mode during a debugging session.

Testing XICE installation

To test XICE, you need to have your emulator fully set up and connected to a working target system, a null target, or one of Applied's optional demonstrator modules (if available for your system). If you need to set up the hardware, see the hardware setup and reference guide appropriate to your emulator. You must also have the source files in either the local directory or the same directory as the absolute file.

The simplest way to test XICE is to invoke it using the `-e boot` parameters.

For example, to invoke and test XICE for 68302 on an EL 1600 emulator, enter:

vxel302 -e boot

See the listing above for the executable name for your system.

The **-e boot** parameter forces download of the emulator operating system. As the debugger comes up, it displays messages on the screen regarding initializing the system and downloading the operating system. Once the debugger presents the prompt in the command viewport, you may assume that XICE is correctly loaded and available to users.

If XICE fails to load or reports an error, see Appendix B, "Troubleshooting."

Note

If you use other Applied Microsystems software or Applied-compatible debuggers with your emulator, each time you use XICE after using another debugger you should use the **-e boot** option to ensure that XICE loads the appropriate shell code. If XICE detects mismatched shell code, it warns you during boot up but does not abort.

To quit XICE, type:

QUIT Y (or **Q Y**)

Debugger invocation parameters

XICE may be invoked with any of the parameters described in the table below. These allow you to tailor each session as you want it.

From the DOS command line, you specify these parameters using a hyphen (-). (INCLUDE, JOURNAL, and LOG can also be invoked from the XICE command line.)

For example, to invoke XICE for 68302 from the DOS command line and load only the symbols from a named file, you would enter:

vxel302 -ni *file_name*

See the table above for the executable for your system.

The following table explains each of the available parameters.

Note

Both XICE and XRAY can generate a start-up file with the same default name: **startup.xry.** If you plan to use start-up files with both XICE and XRAY, you should create files with distinct names so that the appropriate file is accessed at invocation.

Table 3-2 Invocation Parameters

PARAMETER*	DEFINITION
-ni	The "no image" option loads the symbols only.
-b	The "brief" option takes all command input from the include file specified by **-i** instead of from the keyboard. There is no screen output, and any error will cause the program to halt. (If the include file specifies "error=cont" then execution continues after an error.) If you wish to capture output, use **-j**.
abs_file	The name of an absolute object module that is loaded into the debugger's simulated target memory. The default extension is **.x** (UNIX) or **.abs** (DOS). A working or null target must be connected before starting XICE. Only one absolute object module may be loaded into XICE on the command line. If you need to load multiple absolute object modules, use the LOAD command after starting the debugger. If the filename specified on the command line is a file written in assembly, XICE will enter assembly-level mode and will not be able to enter high-level mode during the debugging session.
-i *include_ filename*	The name of an include file that is read before any debugger commands are entered. The default extension is **.inc**.
-s *startup_ filename*	The name of a startup file containing startup parameter definitions. The default extension is **.xry**. If a startup file exists and you start XICE with -e boot OR with -s <filename> and -e boot, the -e boot is ignored and shell code is not reloaded; shell code would only be loaded if the startup file included the -e boot parameter.
-j *journal_ file*	Command output and viewport information is saved in the specified *journal_file*. The default extension is **.jou**.
-l log_ file	User commands and a record of any errors are placed in *log_file*. The default extension is **.log**.

Table 3-2 Invocation Parameters

PARAMETER*	DEFINITION
-e boot	The boot option forces the emulator to reboot. This emulator-only option downloads new emulator control code. Note that this option takes several seconds to complete and is not necessary each time you start the debugger. If a startup file exists and you start XICE with -e boot OR with -s <filename> and -e boot, the -e boot is ignored and shell code is not reloaded; shell code would only be loaded if the startup file included the -e boot parameter. The STARTUP command is used to save startup parameters to a file and will save the -e option if it was used to invoke XICE. This will result in loading emulator control code each time you start the debugger. If you wish to start XICE without reloading emulator control code each time, enter the STARTUP command only after starting XICE without the -e boot option.

Appendix A
XICE.CFG for the 68000/HC000/EC000/ 302 Emulators

Xice.cfg is a configuration file that lets you define one or more sets of default operational states for one or more emulators each time you invoke XICE. AMCEMUL is an environment variable that specifies which operational set to call when you invoke XICE. Both must be set up correctly for successful operation.

The information that follows describes setting up the file **xice.cfg**[1] for the 68000, 68HC000, 68EC000, and 68302 EL1600 emulators. The master media for the XICE debugger provides a default **xice.cfg** that you must adapt to match your particular setup needs.[2] The configuration files are slightly different depending on whether you use UNIX or DOS. You should print a copy of your specific **xice.cfg**, which is in the **bin** directory, to use as a reference as you read this section.

1. **xice.cfg** is used only with the XICE debugger. It is not used with the other components of the toolchain.

2. When you make changes to **xice.cfg**, be careful to observe the following rules of syntax:

   - A backslash (\) indicates that information in that section continues to the next line.There must be a space between the last character of the line and the backslash. There must be no characters between the backslash and the end of the line.
   - Don't start descriptions in the first column of a line.
   - Match the colon and parenthesis format of existing lines.

Note 

Before you begin modifying **xice.cfg**, make a copy of the original so you can return to the defaults if you need to. If you are a UNIX user, you will also need to have write permission so that you can edit the file.

Note 

Applied Microsystems provides support for the 68000 microprocessor using two different hardware configurations. One configuration consists of an emulation board, a probe module and a probe tip; the second configuration consists of an emulation board and a probe tip. When there is a need to differentiate between the two versions, the first will be referred to as the probe module/probe tip configuration and the second will be referred to as the probe tip only configuration. See Chapter 1 of the *Hardware Setup and Reference Guide* to identify the configuration of your emulator.

Using one or multiple configuration files

There are two basic ways to set up **xice.cfg**. The first is to set up a single **xice.cfg** file with multiple emulation descriptions that cover all the emulator setup needs for all the emulation projects being worked on. This allows the specification of one or more setups for one or more emulators from a single source. The other method is to place a copy of **xice.cfg** in each directory from which XICE will be run with a setup in that file to meet the needs of that particular project or user.

Setting AMCEMUL

Whether you use a single **xice.cfg** file and list multiple emulation descriptions in it, or multiple **xice.cfg** files and keep them in the project directories, you must set the environment variable AMCEMUL to specify one of the emulation descriptions defined in **xice.cfg**. For example, if you have two emulators, you can set **xice.cfg** to define the first emulator as **emul1** and the second emulator as **emul2**. In your **.login** (UNIX) or **AUTOEXEC.BAT** (DOS), you then set AMCEMUL to point to **emul1** as the default emulator.

Whenever necessary, from the DOS or UNIX command line you can dynamically redefine AMCEMUL to request **emul2** any time that **emul1** is not available, using the **SET** (DOS) or **setenv** (UNIX) commands.

The command to set AMCEMUL, if you use UNIX with a csh shell, is:

setenv AMCEMUL *emulator_type*

The command to set AMCEMUL, if you use DOS, is:

set AMCEMUL=*emulator_type*

Unless you plan to switch frequently from one emulator type to another, you should place the variable statement in your **AUTOEXEC.BAT** or **.login** file.

Configuration file structure

The file **xice.cfg** has two major parts. The first lists high-level descriptions of each emulation setup to be used. The second part of **xice.cfg** is subdivided into three or four subsections that list the low-level parameters for each of the lines in the emulation descriptions. You may edit this file with any text editor; however, be sure to save it as a non-formatted text file.

At the end of **xice.cfg**, a convenient set of brief comments also explains the switches and lists the registers supported by the debugger. This section is informational only and is provided as a quick reference to the registers and switch set specifications.

Emulation descriptions

Xice.cfg begins with one or more high-level descriptions of emulation setups that follow the same four or five-line format. In the **xice.cfg** file on the distribution media, there is usually only one emulator description for each emulator hardware configuration supported. You may copy the format and define as many of these as you want to specify different emulation devices or different default configurations of the same device.

A typical example of an entry for an emulator is as follows:

```
/*Emulator Description*/

EMULATOR

  EMUL1        EMUL_CONFIG_FILE:emulcfg.dat \
               PORT: eth1 \
               SWITCHES:sw1 \
               OVERLAY:ovl1\
               REGISTERS:reg1
```

The EL 1600 system provides three emulator descriptions. Use the one appropriate to your system to set AMCEMUL. An example of the three descriptions provided for Unix hosts follows:

```
EMULATOR

/* old 68000 */
EL68000  EMUL_CONFIG_FILE:emulcfg.dat \
         PORT:eth68000 \
         SWITCHES:SW68000 \
         OVERLAY:OVLCDEMON \
         REGISTERS:REG68000
```

```
/* 68hc/ec000, new 68000*/
EL68HC000    EMUL_CONFIG_FILE:emulcfg.dat \
             PORT:eth68000 \
             SWITCHES:SW68HC000 \
             OVERLAY:OVLCDEMON \
             REGISTERS:REG68000

/* 68302 */
EL68302  EMUL_CONFIG_FILE:emulcfg.dat \
         PORT:eth68302 \
         SWITCHES:SW68302 \
         OVERLAY:OVLCDEMON \
         REGISTERS:REG68302
```

Note

For 68000 emulators using the probe module/probe tip configuration, use the "old 68000" emulator description. For 68000 emulators using the probe tip only configuration, use the "68hc/ec000, new 68000" emulator description.

The EMUL_CONFIG_FILE descriptor specifies the name of the configuration file for the emulation device (**emulcfg.dat**). Unlike the other descriptors in the emulation description, this field is the only one that does not have a corresponding section to define it. If you change the name of the emulation configuration file or place it anywhere other than the installation **bin** directory, you must specify its new name and/or path in **xice.cfg**. **Emulcfg.dat** provides the emulator with hardware-specific code for downloads.

The PORT descriptor specifies the communications method and, in networked environments, the emulator name. Port settings must match the type of communication established during emulator setup and configuration. See your hardware setup and reference guide for procedures if you have not already configured host-emulator communications. The PORTS section of **xice.cfg** is described more fully below.

The SWITCHES descriptor specifies the set of default operational settings to apply to the emulation device when you invoke XICE. Most of these softswitch settings may be changed interactively during a debugging session. See SWITCHES below for more complete explanations of each switch setting.

For those emulation devices with optional overlay memory, the OVERLAY descriptor specifies any default overlay memory mappings. These let you automatically load a program into overlay memory at invocation. The OVERLAY descriptor is described more fully below.

The REGISTER descriptor specifies the values for any registers that need to be preset. The REGISTER descriptor and the registers supported by the emulation device are described below.

Defining the descriptor statements

The rest of this appendix describes the sections of the **xice.cfg** file that are used to generate the low-level definitions for each of the lines of an emulation description.

PORTS

The PORT section sets up the specifications for the communications method and the emulator name. Port settings must match the type of communication established during setup and configuration of your emulator. See your hardware setup and reference guide for procedures if you have not already configured host-emulator communications.

If the factory defaults do not match your host configuration, simply modify the defaults, or create additional statements with a uniquely numbered label. The following example of an additional statement shows the unique values in bold. To use it, you would change the EMUL1 PORT descriptor to ETH2.

```
ETH1  TYPE:ethernet  HOST:testemul
ETH2  TYPE:ethernet  HOST:alternative
```

PC hosts

If you are a PC user, the PORTS definition in the **xice.cfg** file on the distribution is set up as follows:

For an EL 1600 emulator for 68000, 68HC000, 68EC000, or 68302 processors:

```
HSP1    TYPE:hsp    DEVICE:ca000  PHYDEVID:1
SER1    TYPE:rs232  PORT:com1     BAUD:115200
```

Specify HSP1 in the EMULATOR PORT declaration to use high-speed parallel. Be sure the DEVICE field matches the address jumpered on the Future Domain adapter (Default:CA000). Likewise, check that the PHYDEVID field matches the setting of SW-2 on the back of the emulator.

Sun SPARC hosts

If you are a UNIX user, the default PORTS definition in **xice.cfg** is as follows:

For an EL 1600 emulator for 68000, 68HC000, 68EC000 processors:

```
ETH68000 TYPE:ethernet HOST:el68000
```

For an EL 1600 emulator for 68302 processors:

```
ETH68302 TYPE:ethernet HOST:el68302
```

Make sure that the HOST field specifies *your* emulator name, the one you chose when you modified /etc/ethers and /etc/hosts during hardware setup. (This is explained in "modifying /etc/ethers and /etc/hosts" in the Ethernet communications chapter of your *Hardware Setup Guide*.) For example, if you are a UNIX user and you have two emulators,cpu_1 and cpu_2, the PORTS section would read:

```
ETH1  TYPE:ethernet  HOST:cpu_1
ETH2  TYPE:ethernet  HOST:cpu_2
```

and there could be at least two emulator descriptions listed in **xice.cfg**, one with the PORTS descriptor `PORT:eth1` and one with the PORTS descriptor `PORT:eth2`.

SWITCHES

The XICE softswitches let you configure defaults for many of the features of the EL 1600 emulator. Table A-1 is organized alphabetically by the switch name and provides an overview of all the switches making up the default set. The default for each softswitch is shown in boldface.

The **xice.cfg** file, as shipped on the distribution media, provides only one set of switch settings, called SW1. This is the full set of all XICE switches, set to their default values. You may add as many additional switch *sets* as you wish and have different switch sets for different emulation devices. All switches should be included in any additional switch sets so that the features are in a known state for your debugging session.

To change the default settings you may use a text editor to edit **xice.cfg**, or you may choose to change individual settings once you have invoked XICE. With the exception of EVTMODE, the settings for the softswitches can also be changed at the command line. The commands for changing the softswitches and a detailed explanation of each switch can be found in Chapter 2 of the *XICE Supplement*. Command names are the same as the softswitch name itself, minus the prefix SW_. Interactive modifications of settings made within XICE are not saved to **xice.cfg**.

Table A-1: XICE Emulator Switches

Name	Settings	Description
SW_BPSPACE	**any**	Specifies breakpoint space as any (user and supervisor).
	user	Specifies breakpoint space as user.
	supervisor	Specifies breakpoint space as supervisor.
SW_BTE	**off**	Bus timeout disabled.
	on	Bus timeout enabled.
SW_BUSTIME	**off**	Records the current interrupt level in trace.
	on	Records bus timing information in trace (under the heading IPL).
SW_CAS	**off**	No address strobe to target during pause.
	on	Continuous address strobe to target during pause.
SW_DBP	**off**	Reports bus error signals on peeks/pokes.
	on	Ignores bus error signals on peeks/pokes.
SW_DNL_GAP	*number*	Specifies maximum number of bytes between download blocks to consider them contiguous. The default is **1**.
SW_DRTMR (68302 only)	**none**	Does not allow a DRAM refresh.
	tmr1	Use TMR1 to trigger a DRAM refresh.
	tmr2	Use TRM2 to trigger a DRAM refresh.
	scc1	Use SCC1 to trigger a DRAM refresh.
	scc2	Use SCC2 to trigger a DRAM refresh.
	scc3	Use SCC3 to trigger a DRAM refresh.
SW_EVTARM	**on**	Enables automatic trigger arming.
	off	Requires that triggers must be armed explicitly.

Table A-1: XICE Emulator Switches

Name	Settings	Description
SW_EVTGRP	**1**	Specifies that events will default to group 1 when armed.
	2	Specifies that events will default to group 2 when armed.
	3	Specifies that events will default to group 3 when armed.
	4	Specifies that events will default to group 4 when armed.
SW_EVTMODE	**off**	Disables event mode. Within the advanced event system, allows use of only simple breakpoints (without macros) for event system breaks.
	on	Enables event mode. Only one event system breakpoint may be armed at a time. It allows assignment of a macro to an advanced event using a complex breakpoint (BC) to tie a macro to a trigger. Note: simple breakpoints not associated with events and triggers (BA,BI,BR,BW) are not affected by SW_EVTMODE. This switch may be set only in the xice.cfg file.
SW_EXVEC	*number*	Specifies the software execution breakpoint trap number, where *number* may be from 1 to 15. The default is **15**.
SW_FAST	off	Does not allow interrupts.
	on	Enable interrupts immediately upon entering RUN mode.
SW_FRZ (68302 only)	**off**	Allows peripheral activity during PAUSE mode.
	on	Disables peripheral activity during PAUSE mode.
SW_FTO	**off**	Bus time-outs require 28,672 clock cycles.
	on	Bus time-outs occur in 112 clock cycles.
SW_MWARN	**on**	Displays a warning if a requested memory access is outside the memory endpoints.
	off	Does not display a warning if requested memory access is outside the memory endpoints.

Table A-1: XICE Emulator Switches

Name	Settings	Description
NULL_TGT (68000/HC000/EC000 configuration only)	on	Enables null target mode, which allows operation without a target.
	off	Disables null target mode.
	auto	Enables automatic sensing of target power; when target power is absent, null target mode is enabled. (68000/HC000/EC000 probe tip only configuration; does not apply to 68000 hardware configurations with emulator board, probe module and probe tip)
SW_OVE	*value*	Specifies which memory spaces overlay responds to. The valid values are as follows: 0x01 SC0 Reserved memory space 0x02 UD or SC1 User data space 0x04 UP or SC2 User program space 0x08 SC3 Reserved memory space 0x10 SC4 Reserved memory space 0x20 SD or SC5 Supervisor data space 0x40 SP or SC6 Supervisor program space 0x80 CPU or SC7 CPU space The default for OVE is **0x66**, which represents 0x40, 0x20, 0x04 and 0x02 ORed together.
SW_OVS	*number*	Inserts *number* wait states, where *number* is a value between 0 and 7. The default is **0** (68302) or **1** (68000, 68000/HC000/EC000).
SW_PERFFMT_STAT		Specifies the display format of the statistical performance data. The PERFORMAT command description in the XICE supplement explains each of the settings.
	standard	Displays percent, histogram.
	percent	Displays percent.
	hits	Displays hits.
	bar	Displays histogram.
	ph	Displays percent, hits.
	pb	Displays percent, histogram
	hb	Displays hits, histogram.
	phb	Displays percent, hits, histogram.
	all	Displays percent, hits, histogram.

Table A-1: XICE Emulator Switches

Name	Settings	Description
SW_PERFINT	1-120 (**3**)	Specifies performance analysis data collection time interval in seconds.
SW_PERFMODE	**demand** always	Performance analysis data stored for later display. Performance analysis data displayed whenever received.
SW_PERFTOL	1... (**2000**)	Specifies symbol search distance.
SW_PPT	**off** on	Disables tracing of peek/poke cycles during PAUSE. Enables tracing of peek/poke cycles during PAUSE.
SW_RFS	**off** on	Does not perform a memory refresh during PAUSE mode. Performs a memory refresh during PAUSE mode.
SW_RFSADR	*address*	Specifies the address for memory refreshing during PAUSE mode. The default is **0**.
SW_RFSASP	*space*	Specifies the address space for memory refreshing. The valid values are as follows: 0 SC0 Reserved memory space 1 UD or SC1 User data space 2 UP or SC2 User program space 3 SC3 Reserved memory space 4 SC4 Reserved memory space 5 SD or SC5 Supervisor data space 6 SP or SC6 Supervisor program space 7 CPU or SC7 CPU space The default is **5**.
SW_RFSMSK	*address_mask*	Specifies the address mask for memory refreshing during PAUSE mode. The default is **0**.
SW_RIRR (68302 only)	**on** off	Restores original registers on reset. Does not restore original registers on reset.
SW_RUN_POLL	*number*	Controls the number of times per second the emulator is polled while in run mode. Valid values are 1-20. The default is **5**.

Table A-1: XICE Emulator Switches

Name	Settings	Description
SW_RUN_TIME	**0** ...	Specifies the amount of time in seconds the emulator will be allowed to run before emulation is broken. Zero is indefinitely.
SW_SCRATCH	*address*	Specifies the starting address in RAM for the 8 bytes of scratch memory in the supervisor program space needed for execution breakpoints. The default is **0xfff0**.
SW_SIA	*address*	Specifies the address of the special interrupt vector. The default address for SIA is **0**.
SW_SIZE *memory_access_type*	**1** 2 4	Sets memory access size to 1 byte. Sets memory access size to 2 bytes. Sets memory access size to 4 bytes. The valid memory access types are the same as for SW_OVERLAY.
SW_SLO	**off** on	Uses the setting for FAST for interrupts. Inserts a 160 clock cycle delay before allowing an interrupt. (If both FAST and SLO are set to on, FAST takes precedence.)
SW_SPACE *memory_access_type* *address_space*	**sp** for code **sd** for others	Specifies the memory space used for various memory accesses. The valid memory access types are the same as for SW_OVERLAY. The valid address spaces are as follows: SC0 Reserved memory space UD or SC1 User data space UP or SC2 User program space SC3 Reserved memory space SC4 Reserved memory space SD or SC5 Supervisor data space SP or SC6 Supervisor program space CPU or SC7 CPU space
SW_TAD	**off** on	Addresses generated during PAUSE mode are output to the target system. Address bus is tri-stated while the emulator is PAUSED and during peeks and pokes.

Table A-1: XICE Emulator Switches

Name	Settings	Description
SW_TED (302 only)	**off**	Disables tracing of external DMA. External DMA cannot access overlay or internal registers and dual-port locations.
	on	Enables tracing of external DMA. External DMA can access overlay, internal registers, and dual-port locations.
SW_TID (302 only)	**off**	Disables tracing of internal DMA. Internal DMA cannot access overlay.
	on	Enables tracing of internal DMA. Internal DMA can access overlay.
SW_TRCFRAME	**0**...	Specifies the cycle number for time 0 timestamp alignment.
SW_TRCINT	**offset**	Timestamps displayed as time relative to bus cycle number specified by TRCFRAME.
	interval	Timestamps displayed as time interval between successive bus cycles.
SW_TRCMODE	**both**	Displays source and assembly instructions interleaved together.
	asm	Displays assembly instructions only.
	src	Displays source level instructions only.
SW_TSTAMP	**off**	Trace LSA data.
	on	Trace timestamp data.
SW_TUNITS	*units*	Timestamp units to be used for trace information. The default for TUNITS is **0**.
SW_UIR (68302 only)	**on**	Automatically updates the emulators internal chip control registers whenever an emulation break occur or whenever the registers are set by the user.
	off	The emulator's internal copy of the chip select registers are not automatically updated.
SW_VERIFY	**on**	Enables memory read-after-write verification.
	off	Disables memory read-after-write verification.

OVERLAY

The OVERLAY section specifies how the emulator's overlay memory should be mapped so that a program can be automatically loaded into overlay memory. If you have no overlay or do not want to map overlay at invocation, simply comment out the OVERLAY lines. Use C language comment structure and be careful *not* to place the comment marks in the first column of the text.

The default overlay mapping in **xice.cfg** file is as follows:

For the EL 1600 for 68000, 68HC000, 68EC000 or 68302:

```
OVERLAY
        OVLCDEMON MAP:(ADDRESS:0x0 LENGTH:0x10000 TYPE:RW)
```

The OVERLAY specification may have multiple lines. You may also have multiple OVERLAY specifications set up and use different OVERLAY statements with different emulation descriptions; simply create OVL2, OVL3, etc. Then change the OVERLAY descriptor in EMUL1.

For each map statement, specify the start address and length of the map (in hex) and the map type:

```
TGT: target memory
RW: overlay read/write
RO: overlay read only

ILLEGAL: illegal access
```

Each map statement should be a multiple of 2K boundaries. Once XICE68K is running, the overlay setup can be changed with the RAM, ROM, and NOMEM commands.

REGISTERS

The registers section allows you to preset the values for any of the valid CPU registers. As with PORTS, SWITCHES, and OVERLAY sections, you may define as many different register setups as you need for register start-up values, then use different register statements with different emulation descriptions.

The default register settings in **xice.cfg** are as follows:

```
REGISTERS
      REG68000 /* No 68000 registers need to be set. */
      REG68302    BAR:0x0e00 OR0:0xdfe0 BR0:0xc001
       /* Set BAR to 0xe00000 */
       /* Set chip select 1 to cover OVLCDEMON map */
```

You can enter registers in either upper case or lower case, but they are displayed in the case shown below.

The 68000, 68HC000, 68EC000, and 68302 registers supported in XICE are as follows:

Register	Name
A0-A7	Address Registers
D0-D7	Data Registers
PC	Program Counter
SSP	Supervisor Stack Pointer
SR	Status Register
USP	User Stack Pointer

The following additional registers, which are related to just the 68302, are also supported by XICE:

Register	Name
BAR	Base Address Register
BCR	Byte Count Register
BR0	Base Register 0
BR1	Base Register 1
BR2	Base Register 2
BR3	Base Register 3
CMR	Channel Mode Register

CR	Command Register
CSR	Channel Status Register
DAPR	Destination Address Pointer Register
DSR1	SCC1 Data Sync Register
DSR2	SCC2 Data Sync Register
DSR3	SCC3 Data Sync Register
FCR	Function Code Register
GIMR	Global Interrupt Mode Register
IMR	Interrupt Mask Register
IPR	Interrupt Pending Register
ISR	Interrupt In-Service Register
OR0	Option Register 0
OR1	Option Register 1
OR2	Option Register 2
OR3	Option Register 3
PACNT	Port A Control Register
PADAT	Port A Data Register
PADDR	Port A Data Direction Register
PBCNT	Port B Control Register
PBDAT	Port B Data Register
PBDDR	Port B Data Direction Register
SAPR	Source Address Pointer Register
SCCE1	SCC1 Event Register
SCCE2	SCC2 Event Register
SCCE3	SCC3 Event Register
SCCM1	SCC1 Mask Register
SCCM2	SCC2 Mask Register
SCCM3	SCC3 Mask Register
SCCS1	SCC1 Status Register
SCCS2	SCC2 Status Register
SCCS3	SCC3 Status Register
SCM1	SCC1 Mode Register
SCM2	SCC2 Mode Register
SCM3	SCC3 Mode Register
SCON1	SCC1 Configuration Register
SCON2	SCC2 Configuration Register
SCON3	SCC3 Configuration Register
SCR	System Control Register
SIMASK	Serial Interface Mask Register
SIMODE	Serial Interface Mode Register

SPMD	SCP, SMC Mode and Clock Control Register
TCN1	Timer Counter Register
TCN2	Timer Counter Register
TCR1	Timer Capture Register
TCR2	Timer Capture Register
TER1	Timer Event Register
TER2	Timer Event Register
TMR1	Timer Mode Register
TMR2	Timer Mode Register
TRR1	Timer Reference Register
TRR2	Timer Reference Register
WCN	Watchdog Counter
WRR	Watchdog Reference Register

Chip Select Registers

A special case that you need to consider in setting up the registers is the set of chip select registers for the 68302. You must configure the 68302 chip select registers to match the target system before starting up XICE. The EL 1600 requires that the chip select registers be programmed to respond in all function code spaces to permit target memory operations to work correctly in some targets. Memory operations (also referred to as peeks and pokes) include actions such as displaying memory, downloading code and data, loading the reset vectors, and fill and block moves.

The registers that need to be set up are BAR and the chip select registers BR0, BR1, BR2, BR3, OR0, OR1, OR2, OR3.

Chapter 1 of the *XICE Supplement* provides an example for setting up the chip selects registers for the Applied Microsystems sales demonstration program **cdemon**.

Appendix B
Troubleshooting

The following list covers the most common problems that occur during installation and initial use of XICE. They are grouped according to when the failure typically occurs:

- during invocation at the command line
- at the sign-on screen
- during normal operation

Each group consists of one or more symptoms. Locate the symptom that suits your situation, and check the causes and solutions listed for each. Where problems are specific to a particular host or communications type, it is noted.

Common Problems

XICE invocation fails at the command line	
Symptoms	MS-DOS: When attempting to run the executable, you get a message 'Bad command or file name'
	UNIX: When attempting to run the executable, you get a message 'xxx: Command not found.'
Cause	The executable is not in your search path.
Solution	Verify the directory containing the executable is listed in the PATH environment variable.
Cause	The new path is not loaded into memory
Solution	MS-DOS: Use the PATH command to load the new path. If you placed the PATH statement in your AUTOEXEC.BAT, you must re-boot the PC to load the new AUTOEXEC.BAT into memory.
	UNIX: Use the setenv command to load the new path into memory. If you placed the new path in your .login or .cshrc file, you must source this file to load it into memory.
Cause	Permissions incorrect (UNIX only).
Solution	Perform an 'ls -l' command on the executable to verify that you have execute permission for the executable. Attempt to run the executable by specifying the full path (this will probably have other problems, but you will at least know the executable can be run).

XICE invocation fails at the command line (continued)	
Symptom	The operating system reports insufficient memory or disk space.
Cause	PC: Not enough memory
Solution	XICE must be run in a 386-class or better machine that has at least 4 MB of memory. There must be at least 540K of DOS memory, and 2 MB of extended memory available to XICE. You may need to reduce the number/size of any TSRs running. Disk caches (such as the DOS SMARTDRV) are notorious for consuming memory, but usually have command-line options for making them smaller.
Cause	PC or Sun: Not enough disk space
Solution	Some versions of XICE require disk space to swap the symbol table information. You need to have about 500KB free, just to bring up XICE. If you do not have this, some things will need to be removed.

XICE fails at sign-on screen	
Symptom	XICE reports that it cannot find xice.cfg
Cause	XICE.CFG missing or not in search path
Solution	The xice.cfg file is required to establish communications and configure the emulator. XICE searches in the following order: 1. Current directory 2. Path specified in XRAYLIB 3. Search path specified in path statement Check that there is a copy of xice.cfg in your current directory OR in a directory in your path OR in the directory pointed to by the XRAYLIB environment variable (preferred). As a temporary test, provide a copy of the xice.cfg file in the local directory, and try again. (UNIX): If using the XRAYLIB environment variable, enter the command 'ls $XRAYLIB' and see if the xice.cfg file is listed.
Cause	XICE.CFG renamed
Solution	You cannot rename the XICE configuration file. Change the name to xice.cfg.
Symptom	XICE reports that the AMCEMUL environment variable is not set
Cause	The variable is not set, or was misspelled when entered
Solutions	(UNIX): Enter the command 'setenv' to show your current environment, and determine that AMCEMUL is set. Enter the command 'echo $AMCEMUL' to make sure that it is found. If necessary, reset it as explained in Appendix A.

Symptom (continued)	XICE reports that the AMCEMUL environment variable is not set
	(DOS): Enter the command 'set' to show your current environment, and determine that AMCEMUL is set. Make sure the only space in the line is between 'set' and 'AMCEMUL' as in 'set AMCEMUL=emul1'. If necessary, reset it as explained in Appendix A.
Symptom	XICE reports it can't find the emulator specified by the AMCEMUL environment variable.
Cause	Variable set incorrectly
Solution	See above to determine the value of the variable. Make sure that this emulator entry exists in the xice.cfg file. If necessary, reset it as explained in Appendix A.
Cause	Wrong xice.cfg being used
Solution	Copy the xice.cfg file to the local directory, and try again. If this works, then XICE is locating a different copy of the xice.cfg file when it is being loaded.
	XICE searches for xice.cfg in the following order:
	1. Current directory
	2. Path specified in XRAYLIB
	3. Search path specified in path statement
Symptom	XICE reports it cannot find the necessary shell files.
Cause	XRAYLIB environment variable may not be set correctly.
Solution	The XRAYLIB variable is used to inform XICE where to find all necessary support files, including the emulator's operating system. The file emulcfg.dat specifies which files are to be downloaded to the emulator, and those files should be in the XICE installation directory pointed to by the PATH variable or in the directory pointed to by the XRAYLIB environment variable.

Symptom	PC: XICE reports a virtual memory failure
Cause	Not enough memory
Solution	XICE must be run in a 386-class machine, that has at least 4MB of memory. There must be at least 540K of DOS memory, and 2 MB of extended memory available to XICE. You may need to reduce the number/size of any TSRs running. Disk caches (such as the DOS SMARTDRV) are notorious for consuming memory, but usually have command-line options for making them smaller.
Symptom	XICE cannot connect to the emulator.
Cause	Emulator is turned off
Solution	Make sure power is connected and emulator is turned on.
Cause	Emulator is not connected properly to the host or network
Solution	(Ethernet) Tighten the ethernet cable connection between the emulator and network or host. (HSP) Tighten the HSP/SCSI cable connection between the emulator and the PC. Be sure the cable on the PC is plugged into the Future Domain SCSI controller and not a serial port (the connectors are the same gender). If an extension cable is used, be sure the Pin 1 keys line up at the cable connection. (RS-232) Tighten the cable connection between the emulator and the host. If you substituted your own cable, check the pinout with that shown in the hardware manual.
Cause	Emulator is connected, but switches are incorrect.
Solution	Verify the switch settings on the emulator device are correct for the type of communications desired. See the appropriate communications chapter in the hardware manuals. Pay careful attention to baud rate (RS-232), or HSP ID (HSP). Check these against those selected in xice.cfg. if needed.

Symptom (continued)	XICE cannot connect to the emulator.
Cause	Switches are correct, but xice.cfg is incorrect
Solution	Check that the PORT line of the EMULATOR type and the PORTS definition match, and that the PORTS definition is appropriate to the switch settings on the emulator or CodeTAP. See Appendix A for explanations of each section of xice.cfg.
Cause	(Ethernet) Incorrect host name.
Solution	Using the command /usr/etc/ping, ping the emulator using the hostname you believe to be the emulator. For example, if the emulator's name should be 'emul1' (verify with your system administrator), enter the command */usr/etc/ping emul1* (Syntax for ping may vary by site; see your system administrator.) This should report the emulator to be alive. If not, the emulator has been installed incorrectly, and you may need sysad assistance to correct it. Verify each step of the Ethernet installation given in the hardware manual. If the ping works, shut the emulator off, and try again, to verify that you are talking to the emulator. The ping should fail. If this happens, you have the correct address of the emulator, so turn it back on. Now check the PORTS section of xice.cfg to verify that you have entered the emulator name correctly and that the PORT line EMULATOR section points to the appropriate line in the PORTS section. See Appendix A if you need further explanation of each section of xice.cfg.

Symptom (continued)	XICE cannot connect to the emulator.
Cause	(HSP) Incorrect Future Domain adapter address.
Solution	The adapter's address must be entered in the DEVICE line of xice.cfg. Notice that a truncated version is used; e.g., CA000 instead of CA00:0000. Refer to the Future Domain manual for the correct jumper settings for your card. Verify that this is specified in the DEVICE field of the PORTS section of xice.cfg. Procedures are covered in the HSP communications chapter of the hardware manual. This information is also reported when the XICE starts running, so you may verify it there, as well.
Cause	(HSP) Incorrect HSP physical ID specified.
Solution	Check SW-2 setting of the emulator. The Physical ID specified in the DEVICE line of xice.cfg must be the same as that specified by the SW-2. See Appendix A for an explanation of the PORTS section of xice.cfg and the DEVICE field.
Cause	(HSP) Adapter address conflict.
Solution	In PC's with many cards, especially video cards, the Future Domain card can sometimes conflict with some of the other cards, making it unable to communicate. Check for conflicts between the Future Domain card and other cards in the PC. You will need to change one of the card's configurations to remedy the conflict. The HSP communications section of the hardware manual explains how to resolve address conflicts.

Symptom (continued)	XICE cannot connect to the emulator.
Cause	(RS-232) Incorrect serial port
Solution	Verify the physical RS-232 port (COM1~4) is correctly specified in the xice.cfg file. Check that the EMULATOR PORT definition points to the appropriate line in the PORTS section of xice.cfg. The port XICE is attempting to connect to is reported when the program starts running; so you may verify it there as well.
Cause	(RS-232) Incorrect baud rate
Solution	Verify the emulator switch setting is correct for the baud rate selected, and make sure the switch specifies the same speed as that specified in the xice.cfg file. Setting the baud rate switch is covered in the RS-232 communications section of the hardware manual. XICE requires use of settings A-E. The EMULATOR PORT field in xice.cfg must point to the appropriate field in the PORTS section of xice.cfg.The selected baud rate is also reported by XICE when the program starts running, so verify it there as well.

Symptom (continued)	XICE cannot connect to the emulator.
Cause	(RS-232) Conflicts or memory allocation errors
Solution	Sometimes TSRs and other devices loaded by the CONFIG.SYS cause conflicts with XICE. Likewise, variables set in AUTOEXEC.BAT may cause problems. Try creating a bare-bones systems disk with a CONFIG.SYS and AUTOEXEC.BAT set up solely for XICE. If booting your system with this disk allows you to invoke XICE with no problems, experiment with the settings in your original CONFIG.SYS and AUTOEXEC.BAT to find a setup that works with XICE and your other configuration elements. XICE must be run in a 386-class or better machine that has at least 4 MB of memory. There must be at least 540K of DOS memory, and 2 MB of extended memory available to XICE. You may need to reduce the number/size of any TSRs running. Disk caches (such as the DOS SMARTDRV) are notorious for consuming memory, but usually have command-line options for making them smaller.

Emulator works, but after some time XICE reports a timeout error

Symptom	XICE reports a communication timeout.
Cause	Loose connectors
Solution	All connectors at the rear of the emulator can be fastened securely. As soon as initial configuration is completed, it is wise to fasten all cables.
Cause	(RS-232) Incorrect baud rate
Solution	Make sure the baud rate switch on the emulator is set to a number between A and F. The RS-232 communications section of the hardware manual covers baud rate switch settings. In a few cases the baud rate may need to be reduced to 57600 or lower if you PC will not reliably support higher speeds.

Symptom	Red light on the front of the emulator remains on or is blinking steadily.
Cause	Internal failure
Solution	During normal operation, the red light lights only during host-emulator communication. Regular blinking denotes some internal emulator failure. Call Customer Support at 1-800-ASK-4-AMC.

Download errors

Symptom	(PC) Downloads fail with a message about virtual memory simulator failure.
Cause	Not enough disk space.
Solution	This version of XICE is swapping symbols to disk and needs more disk space to continue. Clear more disk space.
Symptom	(PC) Downloads fail with a message about lack of symbol space in debugger.
Cause	Not enough extended memory
Solution	This version of XICE is storing symbols in extended memory, and has run out of space. You need at least as much memory available as the size of the object file being loaded, and usually you need about 25% more. So, if you are loading a 4MB object file, you will need at least 8MB of memory in your PC (4 for XICE, and 4 for the symbolic information).

Miscellaneous errors

Symptom	XICE reports the shell is newer or older than expected.
Cause	Emulator has not been re-booted.
Solution	When you first run a new version of XICE, it is a good idea to invoke it with the **-e boot** switch, so the operating system in the emulator can be updated with the latest version provided on the XICE distribution.

Symptom (continued)	XICE reports the shell is newer or older than expected.
Cause	Path includes earlier versions of the shell file.
Solution	You are encouraged to keep only the most current version of XICE on you system. If you choose to keep earlier versions available, be sure to modify all path statements, environment variables, and user files containing file pointers (includes, start-up file etc.) to search the appropriate new paths. XICE searches for its support files in the following order: 1. current directory 2. path specified in XRAYLIB 3. paths specified in PATH variable
Cause	Corrupted distribution
	To test the integrity of the shell file shipped with this version of XICE, copy the file xxx.shl from the distribution media into your current directory, and invoke XICE with the -e boot option. If XICE still reports a version error, please call Customer Support at 1-800-ASK-4-AMC.

Calling Customer Support

If none of the suggested solutions works, please call the Applied Microsystems' Customer Support department at 1-800-ASK-4-AMC. To make this process easier for you, please have the following information available:

- Your Support Agreement number (if applicable).
- The host you are running on (Sun 4 or PC). If PC, be prepared to give its available conventional and extended RAM.
- The emulation device you are connected to (68030 emulator, 80960 CodeTap, etc).
- The version of XICE you are using, as well as the information reported by the HWCONFIG command in XICE (assuming you can get this).
- The exact sequence of operations/commands used to duplicate the problem.
- The serial numbers of your hardware and software (the serial number of XICE is displayed on the start-up screen, and the hardware is on a small sticker located somewhere (usually back or bottom) on the emulator or CodeTAP.

Index

Symbols

A

B

C

D

E

F

H

T

U

X–Y–Z

Applied Microsystems Corporation

Applied Microsystems Corporation maintains a worldwide network of direct sales offices committed to quality service and support. For information on products, pricing, or delivery, please call the nearest office listed below. If you are unsure which office to contact, call 1-800-426-3925 for assistance.

CORPORATE OFFICE
Applied Microsystems Corporation
5020 148th Avenue Northeast
P.O. Box 97002
Redmond, WA 98073-9702
(206) 882-2000
1-800-426-3925
Customer Support
1-800-ASK-4AMC
TRT TELEX 185196
Fax (206) 883-3049

EUROPE
Applied Microsystems Corporation Ltd
AMC House
South Street
Wendover
Aylesbury, Bucks
HP22 6EF England
44 (0) 296-625462
Telex 265871 REF WOT 004
Fax 44 (0) 296-623460

GERMANY
Applied Microsystems GmbH
Dammstrasse 6
W-6453 Seligenstadt
Germany
06182/9203-0
Fax 06182/9203-15

JAPAN
Applied Microsystems Japan, Ltd.
Nihon Seimei
Nishi-Gotanda Building
7-24-5 Nishi-Gotanda
Shinagawa-Ku
Tokyo T141, Japan
3-3493-0770
Fax 3-3493-7270

U.S. SALES OFFICES
Applied Microsystems
Corporation of Washington
3333 Bowers Avenue
Suite #220
Santa Clara, CA 95054
(408) 727-5433
Fax (408) 727-9011

Applied Microsystems
Corporation of Washington
25909 Pala Place
Suite #280
Mission Viejo, CA 92691
(714) 588-0585
Fax (714) 588-1476

Applied Microsystems Corporation
14643 Dallas Parkway
Suite 230, LB-76
Dallas, Texas 75240
(214) 991-6344
Fax (214) 991-4581

Applied Microsystems
919E North Plum Grove Road
Schaumburg, IL 60173
(708) 240-2000
Fax (708) 240-1309

Applied Microsystems
Corporation of Washington
6 Cabot Place
Stoughton, MA 02072
(617) 341-3121
Fax (617) 341-0245

Part No.	Revision History	Minimum SW Ver.	Date
922-17140-00	First 68000/302 release on the EL1600.	XEL 2.0	4/92
922-17140-01	4.2/6.8 toolcahin, Flexlm, XICE 6.0.		10/92
922-17140-02	For DEC installations, use MRI Flexible License Manager Toolchain.		12/92
922-17140-03	Add support for PC HSP communications and remove support for PC SCSI communications. Include installation procedures for XICE only; move Toolchain installation to Toolchain manuals.	XICE 6.40	5/93

P/N 922-17140-03
May 1993