XENIX* 286 INSTALLATION AND CONFIGURATION GUIDE

*XENIX is a trademark of Microsoft Corporation.

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Order Number: 174386-001

XENIX* 286
INSTALLATION AND CONFIGURATION GUIDE

Order Number: 174386-001

*XENIX is a trademark of Microsoft Corporation.

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

REV.	REVISION HISTORY	DATE
-001	Original issue	11/84

CONTENTS

CONTENTS **PAGE**

TABLES

FIGURES

intel®

CHAPTER 1
INTRODUCTION

Manual Overview

This manual is designed to guide you while installing and configuring the XENIX 286 Operating System on an Intel System 300 Series microcomputer. The manual provides necessary background information about the operating system, theoretical information about installation and configuration, step-by-step instructions for installing XENIX, and fundamentals of configuring XENIX. While using this manual, you will occasionally be referenced to another manual within the XENIX 286 operating system library set; Appendix B contains a complete list of the manuals in the library set along with ordering information.

After you have installed XENIX, you must configure it. Configuring XENIX means specifying the hardware configuration, the hardware parameters, and the software requirements for running the operating system. The system administrator (or whoever has **root** privileges) is responsible for configuring the system. Unlike the installation, ideally done only once, XENIX must be configured any time the hardware specifications of the computer change.

This manual contains the following chapters and appendixes:

Chapter 1--Introduction
: This chapter introduces the reader to the *XENIX 286 Installation and Configuration Guide*. Included is an overview of the manual, a description of who needs to use the manual, a list of notational conventions used throughout the manual, and a brief synopsis of the chapters and appendixes in the manual.

Chapter 2--Installing XENIX 286
: Chapter 2 provides the reader with instructions for installing XENIX 286. This chapter includes a description of equipment needed for installing XENIX 286, a brief description of the XENIX Basic and Extended System software packages and the materials provided with each, an overview of the installation process, and a step-by-step installation table.

Chapter 3--Configuring XENIX 286
: This chapter defines system configuration and describes the fundamentals of configuring XENIX. Included are a list of the files involved during configuration, a discussion of when configuration is necessary, and examples.

Appendix A--Release Notes
: This appendix explains the differences between XENIX 286 Release 1.0 and XENIX 286 Release 3.0.

Appendix B--Related Publications
: This appendix is a list of publications related to the XENIX 286 product. In the case of Intel publications, the Intel order number is included.

Intended Users

This installation and configuration guide is intended for use by anyone who installs the XENIX 286 operating system on a System 300 Series microcomputer. Frequently, the person installing XENIX is the system administrator who usually has some experience with operating systems, though not always. If you have little or no experience with operating systems, read this manual completely before starting the installation and configuration. Chapter 3, "Configuring XENIX 286," is intended for experienced XENIX users familiar with the C programming language and device drivers. All users should also read the *Overview of the XENIX 286 Operating System* before installing or configuring XENIX.

Notational Conventions

This manual uses the following notational conventions:

- Literal names are bolded where they occur in text, e.g., **/sys/include, printf, dev_tab, EOF.**
- Syntactic categories are italicized where they occur and indicate that you must substitute an instance of the category, e.g., *filename.*
- In examples of dialogue with the XENIX 286 system, characters entered by the user are bolded.
- In syntax descriptions, optional items are enclosed in brackets (**[ ]**).
- Items that can be repeated one or more times are followed by an ellipsis (**...**).
- Items that can be repeated zero or more times are enclosed in brackets and followed by an ellipsis (**[ ]...**).
- A choice between items is indicated by separating the items with vertical bars (**|**).

intel®

CHAPTER 2
INSTALLING XENIX 286

Introduction

This chapter gives the user an overview of the XENIX 286 installation process. Included are descriptions of the standard hardware configurations that can be used with XENIX 286, the XENIX 286 Basic and Extended Operating System packages, and the installation process.

Hardware Configurations

This section describes the Intel hardware systems (standard configurations) available for use with the XENIX 286 operating system. At present, two Intel systems can be used with XENIX 286: the System 286/310 and the System 286/380. Both systems use the iSBC® 286/10 Single Board Computer as their processor board. For use with XENIX 286, the System 286/310 is available in three standard configurations and the System 286/380 is available in one standard configuration. Table 2-1 lists the available standard configurations for each system and the equipment for each.

Table 2-1. System 286/300 Series Standard Configurations

System		Standard Hardware Configurations						
System	Model	Processor Board	Memory Boards	Communications Board	Hard Disk Controller Board	Flexible Disk Controller Board	Winchester Hard Disk Drive	Flexible Disk Drive
310	-17	iSBC 286/10	iSBC 012CX	———	iSBC 215G	iSBX 218A	5.25" 19 MB	5.25"
	-40	iSBC 286/10	iSBC 012CX	———	iSBC 215G	iSBX 218A	5.25" 42 MB	5.25"
	-41	iSBC 286/10	iSBC 012CX	iSBC 188/48	iSBC 215G	iSBX 218A	5.25" 42 MB	5.25"
380	—	iSBC 286/10	iSBC 012CX	———	iSBC 215G	iSBX 218A	8" 32 MB	8"

If you are using a System 286/300 Series microcomputer that is not one of the configurations described in Table 2-1, it must have the following equipment before you can run XENIX 286:

- iSBC 286/10 Single Board Computer
- 512KB of RAM
- Hard disk with minimum of 10MB capacity (formatted)
- 5¼-inch Flexible disk drive

If your system does not meet the above minimum equipment requirements, XENIX 286 will not run on it.

XENIX 286 System Packages

Intel has divided the XENIX operating system into two different products to satisfy different user requirements. These products are the Basic System and the Extended System.

The Basic System is intended for all users. It has all of the things needed to run applications software and to administer the system. It also has general-purpose tools like the **ed** and **vi** text editors, electronic communications, and many commands.

The Basic System comes with seven manuals and the following diskettes:

8-inch media (6 diskettes)	2 boot diskettes 4 Basic System diskettes
5¼-inch media (14 diskettes)	2 boot diskettes 12 Basic System diskettes

The Extended System is made up of software development tools and text formatting tools. The software development tools include utility programs, standard C libraries, system calls, a C compiler, an assembler, a linker, a loader, a debugger, a lexical analyzer, and a compiler-compiler (a program that generates a compiler). The text formatting tools include commands for improving writing, **mm** (memorandum) macros, and standard **nroff** and **troff** programs. The **mm** macros are codes that you use to prepare memos, letters, reports, and other documents. The **nroff** program formats documents for a printer, and the **troff** program formats documents and prints them on a phototypesetter.

The Extended System comes with four manuals and the following diskettes:

8-inch media	3 Extended System diskettes
5¼-inch media	10 Extended System diskettes

When you receive the XENIX distribution package, check it immediately to be sure that all the manuals and diskettes are there and the diskettes undamaged. If you find that the distribution package is incomplete or damaged, contact your Intel sales representative immediately.

Installation Process Overview

Installing the XENIX 286 operating system is a simple process and takes about one hour. If you are installing XENIX 286 on a microcomputer system that has an earlier release of XENIX 286, be sure to back up the system before starting the installation process. When you start the installation process and boot the system from a flexible diskette, XENIX first initializes the Winchester disk, deleting any data on it. By backing up the system before starting, you may be able to save hours of re-entering data since XENIX 286 Release 3.0 has provisions for converting from Release 1.0 to Release 3.0 (there is no Release 2.0). The information for converting data and programs is located in Appendix A, "Release Notes."

Begin by powering up the system and running the System Confidence Test (SCT). If the system is already running, reset it and run the SCT. After the SCT executes, you will boot the operating system from the boot diskettes and initialize the Winchester disk. The boot diskettes contain the base operating system and the programs needed to initialize the Winchester disk and prepare it to receive the files from the Basic System diskettes. When XENIX has loaded and transferred files from both boot diskettes, the system will shut down; you must reset it and load XENIX from the Winchester disk.

If the system has a hard disk with a corrupted track 0, XENIX displays this prompt asking you to identify the hard disk type:

```
The following configurations are available:
    a) 10Mb CMI disk. System 286/310-5 standard. [model 5612]
    b) 15Mb CMI disk. System 286/310-17 standard. [model 5619]
    c) 36Mb Quantum disk. System 286/310-40 standard. [model Q540]
    d) 32Mb Priam disk. System 286/380 standard. [model 3450]
    e) 62Mb Priam disk. System 286/380 with iSXM 70Mb. [model 7050]
```

Select the drive type by entering the letter that is listed next to the type drive your system has. XENIX then prompts you to insert each Basic System diskette and press RETURN. After the system has transferred all the files from the last Basic System diskette, XENIX prompts you to insert the next diskette and press RETURN. Enter "no" in response to this prompt. When the system has completed installing the Basic System files, it will ask you if you also wish to install the Extended System. If you answer "yes" to this question, the system prompts you to insert each Extended System diskette and press RETURN. After the Extended System is installed, XENIX tells you that the installation is complete; you may then configure XENIX for your hardware. If you answer "no" to the prompt for installing the Extended System, the installation process is complete and you may then configure the operating system.

Installation Instructions

Table 2-2 is a step-by-step procedure for the XENIX 286 installation process. For brevity, only the system prompts that require a user action are displayed in the "system prompt" column. The system displays other information during the installation; unless an error message appears, the information displayed requires no action. If an error message appears, refer to the *XENIX 286 User's Guide* or the *XENIX 286 System Administrator's Guide* for an explanation.

Table 2-2. XENIX 286 Installation Instructions

TASK	SYSTEM PROMPT	USER ACTION	COMMENT
Power up system		Turn system on	If system is on, press RESET switch on system front panel
Run SCT	*	Enter: U	Enter U within 12 seconds. Satisfies baud rate search and starts SCT
	8274 MPSC ENTER <cr> or (.)	Enter: .	Causes SCT to exit to iSDM monitor after execution
	Interrupt 3 at xxxx.yyyy	Insert diskette labeled "Boot Diskette, 1 of 2"	xxxx.yyyy is a memory address
Load XENIX from flexible diskette	.	Enter: b :wfo:xenix.f	System prompt is a period
Format hard disk and install basic files	bootsys>	Enter: /etc/mksys	If drive is non-Intel, bad track information must be entered by the user
	Please insert diskette labeled "Boot 2 of 2" into the drive hit <cr> when ready to proceed:	Remove diskette from drive and insert diskette labeled "Boot Diskette, 2 of 2" and press RETURN	Formatting the hard disk takes anywhere from 10 to 30 minutes, depending on the size of the device
Reset system and boot XENIX from hard disk	**NORMAL SYSTEM SHUTDOWN**	Remove the second boot diskette and press the RESET switch on the front panel	Removing the diskette before receiving the shutdown message could damage the operating system
Install XENIX Basic System	newsys>	Enter: /etc/instlsys	
	Insert first diskette of the Basic System and type a <cr>:	Insert diskette labeled "Basic System 1 of y" and press RETURN	"y" is the total number of Basic System diskettes in the package
	Insert next diskette (Basic System) and type a <cr> (type "no" if there are no more diskettes to be installed)	Remove diskette from drive, insert the next diskette in sequence, and press RETURN	Use **only** the Basic System diskettes for this portion of the installation; when the last diskette has been installed, answer "no" to the system prompt asking for the next diskette

Table 2-2. XENIX 286 Installation Instructions (Continued)

TASK	SYSTEM PROMPT	USER ACTION	COMMENT
	The Basic System is now installed. Do you also wish to install the Extended System [yes/no]?	Answer "yes" or "no" to the system prompt	If you are not installing the Extended System, installation is complete; disregard the following steps about installing the Extended System and go to the step with system prompt "Installation of Xenix is complete"
Install the Extended System	Installing Extended System... Insert first diskette of the Extended System and type a <cr>	Insert diskette labeled "Extended System 1 of y, Tar Format" and press RETURN	"y" is the total number of Extended System diskettes in the package
	Insert next diskette (Ext. System) and type a <cr> (type "no" if there are no more diskettes to be installed):	Remove diskette from drive and insert the next diskette in sequence (x of y); press RETURN	When the last Extended System diskette has been installed, answer "no" to the system prompt asking for the next diskette
	Installation of Xenix is complete	Remove last diskette from drive	
Access multiuser mode	newsys>	Enter: CONTROL-D	
Set current date and time	Enter new time ([yymmdd]hhmm):	Enter current date and time according to displayed format	You do not need to shut down the system to use it. To shut down, follow the next 3 steps
Login as super-user	login:	Enter: root	Permits system administrator privileges needed to shut down the system properly
Shut down system	#	Enter: /etc/shutdown	If system is not shut down properly, the file system could be destroyed
	NORMAL SYSTEM SHUTDOWN		XENIX installation is complete; you may now use the system after rebooting

intel®

CHAPTER 3
CONFIGURING XENIX 286

Introduction

This section is designed to provide fundamental information necessary to configure the XENIX system; the information contained in this section is intended for experienced XENIX users familiar with the C programming language and device drivers.

In simple terms, configuring XENIX means tailoring XENIX parameters to match the hardware in the computer system and ensuring that all the device drivers and configuration files are in place and functioning. These device drivers and configuration files enable the software, hardware, and user to communicate with each other. This chapter deals with the fundamentals of configuring XENIX and provides two simple examples to clarify the configuration process. Information about writing and configuring device drivers can be found in the *XENIX 286 Device Driver Guide.*

XENIX 286, as installed on the computer system, is configured for a console terminal, an iSBC 215G Winchester Disk Controller Board (with an iSBX™ 218A Flexible Diskette Controller Board, a single Winchester disk, and a single flexible disk drive), an iSBC 188/48 communications board, and a line printer. Adding any other devices or boards to the computer requires that the operating system be configured.

Files Involved

Configuring XENIX involves working with files contained in the **/sys** and **/dev** directories. One exception to this is if you are adding a communications board and additional terminals; in this case, you must also work with files in the **/etc** directory. Within **/sys** are the subdirectories **cfg, conf, h, io,** and **net.** To configure XENIX, you need only be concerned with the subdirectories **cfg** and **conf. cfg** contains the individual configuration files for the devices and boards in the system, and **conf** contains the **xenixconf** and **master** files, which tell the system, respectively, what devices are in the system and what configuration files need to go into the XENIX kernel. When you configure the operating system, you simply edit **cfg** and **conf** to include the necessary information and parameters for the boards and devices in the system. The **/dev** directory contains a file for each device in the system; the individual files are the system identifiers, called device nodes, for each device. During the configuration, you must create or modify the device nodes in **/dev.**

The Configuration Process

This section will guide you through the configuration process; examples will demonstrate how to add an iSBC 188/48 communications board and a CMI 5¼-inch, 19MB Winchester disk drive to a System 286/310-17. Figure 3-1 is a flow chart of the XENIX 286 configuration process and is useful as a quick reference.

Note that this section is not designed to be a step-by-step set of instructions, but rather, a set of guidelines. When adding boards and devices to the system, be sure to configure the operating system before actually installing the hardware.

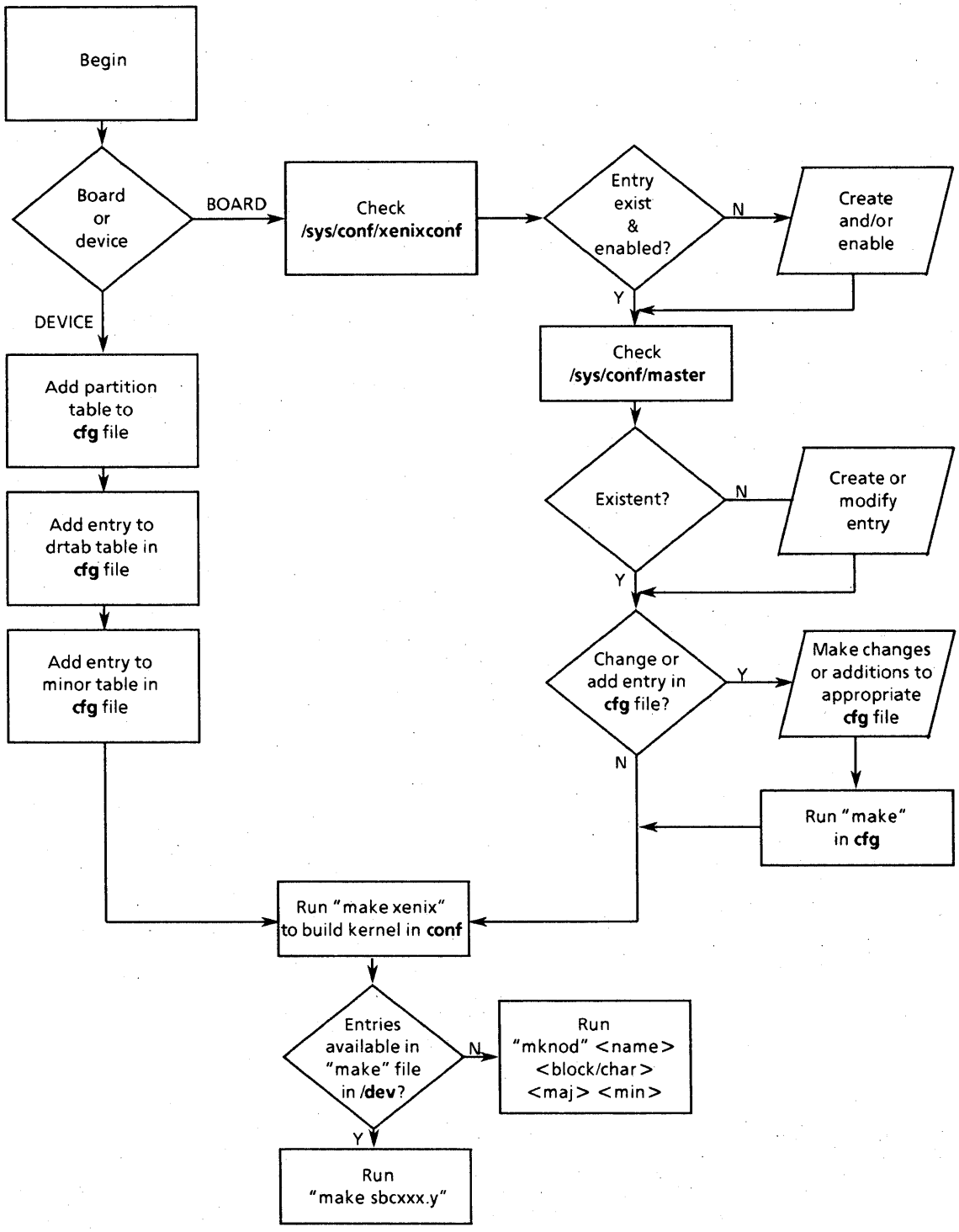

Figure 3-1. Flowchart of the Configuration Process

Adding Boards

You can add different boards to a system to expand or upgrade it. When a board is added, you must also configure XENIX to accept the new board. To illustrate the process, the following example adds a second iSBC 188/48 communications board to a system so more users can be added.

The /sys/conf/xenixconf File

Because you are adding a board to the system, the first step is to check the file **/sys/conf/xenixconf** to be sure that there is an entry for the iSBC 188/48 board and that the entry is enabled. Figure 3-2 illustrates a partial listing of the **/sys/conf/xenixconf** file; note that there is an entry for an iSBC 188/48 board. Each device entry (e.g., i188) is followed by a number, either 1 or 0. The number 1 indicates that the entry is enabled and that the device driver for that device will be included in the XENIX kernel. When the entry is followed by a 0, the entry is not enabled and the device driver is not part of the XENIX kernel.

In this instance, the entry for a 188/48 board is present and enabled. This indicates a 188/48 board already exists in the system; you do not need an entry for each board, only for each type of board. If an entry for the board does not exist, use one of the XENIX text editors and create the entry yourself; include the board designation (i.e., 188) preceded by an "i", and followed by a "1" or "0" to tell the system whether the board is enabled or not.

```
*
*
*           Devices
*
i188        1
i215        1
i534        0
i544        0
i74         1
lp          1
rand        1
debug       1
root        i215    1
pipe        i215    1
swap        i215    2   1   1188
*
*           local parameters
*
timezone    (8*60)
daylight    1
cmask       0
```

Figure 3-2. Listing of **/sys/conf/xenixconf**

The /sys/conf/master File

After you have made sure that an entry exists in **/sys/conf/xenixconf** and is enabled, change directories to the file **/sys/conf/master.** As was mentioned earlier in the "Files Involved" section, **master** contains the specifications for the boards in the system. Figure 3-3 shows a partial listing of the **master** file. There are 14 columns in the listing, but for this example you need only be concerned with two: column 2, labeled "vsiz", and column 12, labeled "vec2".

Column 2 specifies the number of each board type present in the system; the iSBC 544 serial I/O board is the only exception. Because all 544 boards share the same interrupt level, you need only specify "1" in column 2 even if there is more than one board in the system. For all other boards in the system, use one of the text editors to edit the file and specify the actual number of boards of each type present. For this example, find the i188 entry and change the "vsiz" entry to 2.

Column 12 of the listing, "vec2", is the interrupt level specification. If you have more than one board of a particular type, you must specify a separate interrupt level for each of the boards (with the exception of the iSBC 544 board, as noted previously). Use one of the XENIX text editors to enter an interrupt level in column 12 for the second 188/48 board. For this example, use interrupt level 0002.

The first twelve entries in both the "bdevsw" and the "cdevsw" are reserved for use as block devices. The last four of these entries are reserved for additional Intel devices and customer block devices. All block devices have the same "bdevsw" and "cdevsw" number. The "cmaj" number 1 is reserved for use by the memory driver.

name 1	vsiz 2	msk 3	typ 4	hndlr 5	na 6	bmaj 7	cmaj 8	# 9	na 10	vec1 11	vec2 12	vec3 13	vec4 14
i215	1	0137	014	i215	0	0	0	2	-1	0005	0	0	0a
i216	1	0137	014	i216	0	2	2	2	-1	0005	0	0	0a
i214	1	0137	014	i214	0	3	3	2	-1	0005	0	0	0a
i208	1	0137	014	i208	0	4	4	2	-1	0003	0	0	0a
ramd	1	0136	054	ramd	0	5	5	1	-1	0	0	0	0a
xlog	1	0137	014	xlog	0	6	6	2	-1	0005	0	0	0a

The next twelve entries in the "cdevsw" are reserved for character devices. The "cmaj" number 12 is reserved for use by the tty driver.

1p	1	0132	004	1p	0	0	13	1	-1	0107	0	0	0a
i74	1	0137	004	i74	0	0	14	1	-1	0006	0	0	0a
i188	2	0137	004	i188	0	0	15	2	-1	0003	0002	0	0a
i552	1	0137	004	i552	0	0	16	1	-1	0004	0	0	0a
i278	1	0137	004	i278	0	0	17	1	-1	0003	0	0	0a
i544	1	0137	004	i544	0	0	18	4	-1	0003	0	0	0a

Figure 3-3. Partial Listing of **/sys/conf/master**

The /sys/cfg File

The directory **/sys/cfg** contains the configuration files for boards listed in the **master** file. There are two configuration files for each board, one labeled with the extension **.c** and the other labeled with the extension **.o**. The configuration file specifies the number of boards present, among other parameters. You will be working with the file having the **.c** extension. Figure 3-4 illustrates a partial listing of the **c188.c** file.

Using one of the XENIX text editors, in the line "#define NUM188 1", change the 1 to a 2. As is noted in Figure 3-4, you may specify a maximum of four boards, but you must specify at least one. When you finish editing **c188.c**, exit from the editor and enter the **make** command to recompile the modified files and the library.

```
*To configure the maximum number of 188 boards in the system,
*  simply change NUM188.  There is a maximum of 4 and a minimum of 1
*/
#define NUM188      1
int     N188  =  NUM188;
```

Figure 3-4. Partial Listing of **/sys/cfg/c188.c**

Building and Implementing the Kernel

Now that the appropriate files have been edited and the parameters for the board properly set, you must build the kernel, incorporate it into the operating system, make the device nodes in the **/dev** directory, and boot the operating system. To build the kernel, you must be in the directory **/sys/conf**.

Change to directory **/sys/conf** and enter the following command:

```
make xenix
```

The **make** command calls the **config** utility, which constructs the C language file **c.c**. The **c.c** file is then compiled and the resulting output file is named **c.o**. Finally, **make** invokes the XENIX linker to resolve all unresolved references in **c.o** and stores the new kernel in the file **xenix** under the directory **/sys/conf**.

Before the system can use the new kernel, the kernel must be placed in the **root** directory. Change directories to the **root** directory and move the old kernel (also called **xenix**) into another file. One suggestion is to move the old kernel into a file with the name **xenix.old.** Do not destroy the old kernel; you may need it some day to boot the system. Now copy the new kernel (**xenix**) from **/sys/conf** to the **root** directory.

Next, you need to change to the **/dev** directory and make the device nodes there for the new board. Enter the command

```
make sbc188.1
```

This command specifies the board name (sbc188) and the board number (.1). When specifying the board number, remember that while you can have two boards in the system, they are numbered 0 and 1, not 1 and 2.

Finally, you may shut down the system and reset it; the system will boot the operating system with the new kernel. If you are installing extra terminals along with any of the communications boards, you must also modify the **/etc/ttys** and **/etc/ttytype** files to enable the new terminals. (The procedure for enabling terminals is in the *XENIX 286 System Administrator's Guide.*)

A final note about adding boards to the system: if you add memory boards to the system and total system memory exceeds 1MB, use one of the editors to change the amount of "swap space" specified in **/sys/conf/xenixconf.** The swap space must equal or exceed the amount of system RAM and cannot exceed the size of the "swap" partition on the disk.

Adding Devices

As is the case with adding boards to the system, there are a number of instances when you may want to add devices to the system. The most common instance of adding a device is adding a second hard disk drive to increase mass storage capacity. To demonstrate the configuration process when a device is added, the following example will add a second CMI 5¼-inch, 19MB Winchester hard disk drive to a System 286/310-17. The process for adding a device to a System 286/380 computer is the same, only the device names differ. The hardware reference manual for each system should contain a list of device names.

The /sys/cfg Configuration Files

The directory **/sys/cfg** contains the specific configuration file for each board in the system. Within each configuration file are the specifications for each device associated with that board. In this example, you are installing a Winchester disk drive controlled by an iSBC 215G controller board, so you need to modify the file **c215g.c.**

The Partition Table

A partition table defines the number, size, and names of divisions (partitions) on a Winchester disk. The first modification you must make to **c215g.c** is adding a partition table for the new Winchester disk. Each hard disk is divided into sections called partitions. When you create a partition table for each new disk, you specify the number, size, and name of partitions on the disk. Since there is already one hard disk in the system and that disk contains the operating system and file system, you need only specify three partitions for the new Winchester disk: one partition to specify the whole disk including alternate tracks and bad tracks, a second partition called "user" to hold user data, and a third partition called "error" to contain disk error information.

Look at **c215g.c** and you will find a table with the heading

```
struct      i215part  Piw0[ ] = {
```

This table is the partition table for the Winchester disk currently in the system (unit 0) and has a table designation of Piw0. The table designation for any other partition tables will be Piw*x*, where *x* is the next sequential number (e.g., *x* is 1 for the table being added). Use one of the text editors to create a similar partition table in **c215g.c** for the second Winchester disk (unit 1). Figure 3-5 illustrates what the partition table for unit 1 should look like.

The partition table contains three columns separated by commas: column 1 designates the beginning sector of the partition, column 2 specifies the length of the partition in sectors (not the ending sector number), and column 3 is a comment specifying the partition name and length. In the first entry of the table, labeled "0" in the comments column, the formula in column 2 is derived from the hardware specifications for the drive. The first number (305) is the number of cylinders minus one (for the diagnostic track), the second number (6) is the number of surfaces (drive heads), and the last number is the number of sectors per track. This formula will vary with the size and type of the disk being installed; these figures are valid only for the CMI 5¼-inch, 19MB drive.

```
struct      i215part    Piw1[ ] = {
                0,      305*6*9,        /* [0] whole disk (+alts & bad-track) */
                0,        16083,        /* [1] "user":  16084k */
            16083,            9,        /* [2] "error":  9k */
};
```

Figure 3-5. Partition Table Entry in **c215g.c**

The Device Table

The device table defines parameters for the devices attached to the controller board. Some information in the device table is the same as that used in the partition table. For the device table, you need to know the following information for each drive being installed (the numbers in parentheses are the parameters for a CMI 5¼-inch, 19MB Winchester disk drive):

- Number of cylinders (306)
- Number of fixed surfaces or heads (6)
- Number of removable surfaces or heads (0)
- Number of sectors per track (9)
- Sector size in bytes (1024)
- Number of alternative tracks (0)
- Partition table designation (Piw1)
- Unit number (1)
- Drive type (CMI 5.25")

Figure 3-6 shows the device table before editing. The entry in the table defines the parameters for the Winchester drive already in the system; you need to add an entry for the new Winchester drive. The device table you must edit has the heading "215 Board 0, Unit 0 (Wini) Device-Table Definitions (drtab's)". Using the parameters specified in the above list, edit the table and add the parameters under the proper columns. As you can see, the entry for the new drive is nearly identical to the original entry; the only difference is the the partition table designation and the unit number. The entry for the unit number and drive type is a comment and is not mandatory.

```
struct    i215cdrt    i215d00[ ]
/*Cyls,     Fixed,      Remov,      #Sec,      SecSiz,    Nalt,      Partitions*/
306,          6,          0,          9,        1024,      0,           Piw0
```

Figure 3-6. Device Table in **c215g.c**

The Board Configuration Table

The board configuration table in **c215g.c** contains a list of devices attached to the iSBC 215G controller and tells XENIX which device table is associated with each device. For each board in the system there is a corresponding configuration table in the board configuration file. The last column of the board configuration table is a comment and specifies the unit number of the device; you are installing unit 1. Figure 3-7 illustrates a partial listing of the **c215g.c** board configuration table.

```
struct  i215cfg  i215cfdg[ ] = {
/*  WUA          Dev-Code[0],          [1],          [2],      Int,   Device Table        [unit]
      */
     0x01000L,   DEVWINIG,     DEV5FLPY,      STREAMER,     5,     i215d00      /* Unit [0] */
                                                                   0            /* Unit [1] */
```

Figure 3-7. Partial Listing of Board Configuration Table in **c215g.c**

As you did in the device table, you must add an entry to the board configuration table for the new device. When you edit the table, put the new entry on the line with the comment "Unit [1]". Under the "Device Table" column, enter "i215d00" for unit 1.

The Minor Table

The minor table tells XENIX which tables to go to for information when it needs to access a device. Figure 3-8 shows a partial listing of the **c215g.c** minor table.

```
unsigned i215minor[ ] = {

/*      i215MINOR(board#,unit#,drtab#,partition#)*/
                                                    /* [minor] device */
        i215MINOR(0,0,0,0)                          /* [0] cmi track 0 - bad track */
        i215MINOR(0,0,0,1)                          /* [1] cmi root part */
        i215MINOR(0,0,0,2)                          /* [2] cmi swap part*/
        i215MINOR(4,0,0,0)                          /* [3] invalid */
        i215MINOR(4,0,0,0)                          /* [4] invalid */
        i215MINOR(4,0,0,0)                          /* [5] invalid */
        i215MINOR(4,0,0,0)                          /* [6] invalid */
        i215MINOR(4,0,0,0)                          /* [7] invalid */
```

Figure 3-8. Partial Listing of **c215g.c** Minor Table

When XENIX accesses one of the drives, it first looks at the minor table and determines, from the four-digit entry in the minor table, which board, unit number, device table entry, and partition number to work with. Figure 3-9 maps the minor table entries.

The first digit of the sequence directs XENIX to the appropriate board configuration table; a "4" in this position means that there is no device associated with that entry.

The second digit tells XENIX which entry in the board configuration table to read. The board configuration table supplies the name of the device table that XENIX must read.

The third digit tells XENIX which entry in the device table to read, which in turn tells XENIX the name of the partition table to use.

The last digit gives XENIX the number of the entry in the partition table to read. After XENIX gathers all this information, it can access the drive correctly.

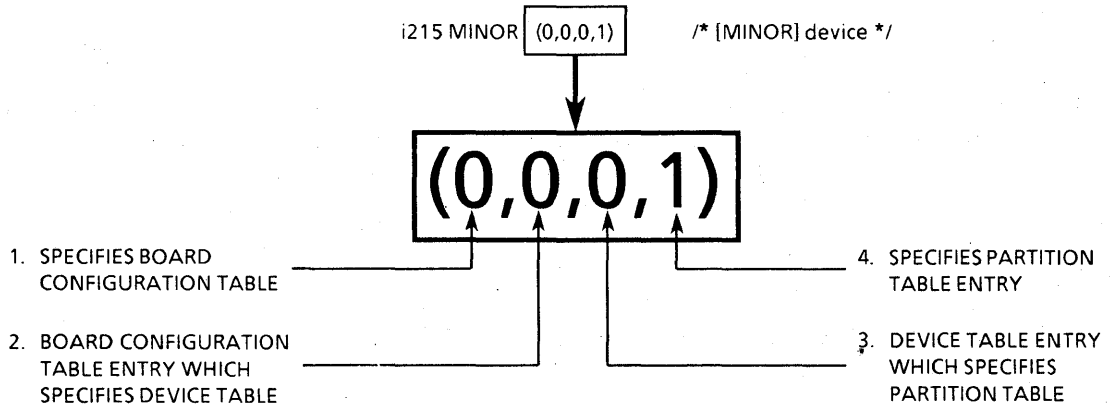

Figure 3-9. Minor Table Mapping Sequence

Since the minor table tells XENIX which partition on the disk to access, there must be an entry in the minor table for each partition on each disk (except the "error" partition). The disk you are installing has been configured for two partitions (plus the error partition); this means that you must make two entries in the minor table; you do not need to make an entry for the error partition. When determining the digits to be included in the minor table sequence, remember that the numbering begins with 0 and not with 1. That means that the first entry in a table is entry number 0, the second entry is number 1, and so on.

The changes you have made to the **c215g.c** file so far have been as follows:

- Partition table with the name Piw1 and three entries added
- Entry number 1 added to the device table
- Entry number 1 added to the board configuration table

To make the entries in the minor table, you will need to locate two lines that have the four-digit sequence (4,0,0,0) and are marked "invalid", as shown in Figure 3-8. Edit the "invalid" lines so that they look like the minor table entries shown in Figure 3-10.

```
i215MINOR(0,1,1,0)          /* [3] cmi track 0 - bad track */
i215MINOR(0,1,1,1)          /* [4] cmi user part */
```

Figure 3-10. Minor Table Entries for New Drive

After you have made the entries in the minor table, enter the **make** command to recompile the modified files and the library.

Building and Implementing the Kernel

Now that the appropriate files have been edited and the parameters for the board properly set, you must build the kernel, incorporate it into the operating system, make the device nodes in the **/dev** directory, and boot the operating system. To build the kernel, you must change to the directory **/sys/conf** and enter the following command:

```
make xenix
```

The **make** command calls the **config** utility, which constructs the C language file **c.c**. The **c.c** file is then compiled and the resulting output file is named **c.o**. Finally, **make** invokes the XENIX linker to resolve all unresolved references in **c.o** and stores the new kernel in the file **xenix** under the directory **/sys/conf**.

The major number for the **mknod** command can be found in the **/sys/conf/master** file. Column numbers seven and eight are labeled "bmaj" and "cmaj" respectively. Column seven contains the block device major number [b] and column eight contains the character device major number [c]. Make a note of the major numbers, as you will need them later when you make the device nodes in **/dev**.

Before the system can use the new kernel, it must be placed in the **root** directory. Change directories to the **root** directory and move the old kernel (also called **xenix**) into another file. One suggestion is to move the old kernel into a file with the name **xenix.old**. Do not destroy the old kernel, as you may need it some day to boot the system. Now copy the new kernel (**xenix**) from **/sys/conf** to the **root** directory.

You now need to change to the **/dev** directory and make the device nodes there for the new device. The command for making the device nodes for a device is different than the command you used for making device nodes after adding a board. This is because the **make** file in **/dev** does not contain the information for individual devices. Consequently, you must use the **mknod** command rather than the **make** command. The syntax of the **mknod**? command is as follows:

mknod *devicename* [c] [b] *major# minor#*

For each partition listed in the **c215g.c** file minor table, you must invoke **mknod** twice: once for the character device specifications ([c]) and once for the block device specifications ([b]). The *devicename* may be any string, however, when you give the raw device name for the character device specification, you must use the same device name that you use for the block device specification and precede it with an "r". For example, the Winchester drive currently in the system has the block device name of "w0"; therefore, the character device name for the raw device is "rw0". Since the system has already established a naming convention for Winchester drives, you should call the new drive "w1" and "rw1".

The minor number is obtained from the minor table in the board configuration file. It is simply the entry number for each partition listed in the minor table (remember, the first entry is entry number 0).

The actual commands to create the device nodes for the new Winchester drive are as follows:

For partition 0:

```
mknod rw1 c 0 3
mknod w1 b 0 3
```

For partition 1:

```
mknod rw1a c 0 4
mknod w1a b 0 4
```

Finally, you may shut down the system and reset it; the system will boot the operating system with the new kernel. After you have completed the operating system configuration, you may install the hardware in the system.

intel®

APPENDIX A
RELEASE NOTES

Introduction

The purpose of this appendix is to assist in porting device drivers, applications, and other programs from XENIX 286 Release 1.0 to XENIX 286 Release 3.0. This appendix is not intended to be a comprehensive document describing all the differences between the two releases; it is intended to serve as a guide to porting code from one release to the other.

XENIX 286 Release 3.0 features numerous enhancements that stem from two sources: the introduction of a new compiler technology and a system based on AT&T's System 3 UNIX. These enhancements, in turn, influence three fundamental areas of the XENIX product: device drivers, the C compiler, and the utilities and libraries.

The new C compiler from Microsoft is more powerful than its predecessor. Some points to consider in compiling applications to run under Release 3.0 are described later in this appendix.

Likewise, System 3 UNIX represents an improvement over UNIX Version 7 (also known as the Seventh Edition UNIX), the operating system from which Release 1.0 was derived. As a result, the functions of many commonly used utilities, including **mv**, **ps**, **uucp**, and **vi**, have changed. If a system call or utility works in an unfamiliar way, consult the manuals shipped with the Release 3.0 product.

Compiler Changes

The following sections discuss the changes made to the compiler from XENIX 286 Release 1.0 to XENIX 286 Release 3.0.

Module Size Limitations

Under certain conditions, the new C compiler issues the error message

```
OUT OF HEAP SPACE
```

This message indicates that the module being compiled has too many statements, too many symbols, or some combination of these and other factors. Reducing the size of the module by splitting it into smaller modules usually eliminates this problem.

Addressing and Stack Size Problems

Some programs will compile without fatal errors yet, when invoked, cause the system to issue the error message

```
Memory fault - core dump
```

The program can cause this message to be issued if it is using pointers and integers interchangeably or if it is using more stack space than the compiler allocates. In the first case, the compiler issues an error message of the form

```
warning: different levels of indirection
```

during compilation. This message indicates problems in the way the program is using pointer variables. In the second case, recompiling the program with a larger value for the stack size (the -F option in the compiler invocation) will usually solve the problem.

Compiler Error Messages

Because the new C compiler does much stronger type checking than the previous version, programs that would compile and run under Release 1.0 may not compile under Release 3.0. Most compiler error messages concern type conflicts in statements. Noting the location of the error in the source file and cleaning up the declarations of the variables involved is usually sufficient to solve these problems. Further, the Release 3.0 compiler itself has many extensions over the Release 1.0 compiler, and they often clear up type conflict errors such as the type "void."

Compiler Invocation

In addition, many options to the C compiler have changed. Refer to the *XENIX 286 C Library Guide* for full descriptions of invocation options for the Release 3.0 compiler.

System Changes

The following sections discuss the changes made to the operating system from XENIX 286 Release 1.0 to XENIX 286 Release 3.0.

Header Files

Many of the **.h** files have been changed to include improvements to library routines and system calls. Before recompiling a Release 1.0 program, check the **.h** files it uses for any changes that may need to be reflected in the program code itself.

Library Changes

In XENIX 286 Release 3.0, some libraries have revised functions and/or names. For example, the routines that were called **index** and **rindex** in Release 1.0 are now named **strchr** and **strrchr**, respectively.

The Microsoft extensions to UNIX, such as the semaphore calls, reside in a library called ***libx.a** (where * is **S**, **M**, or **L**, meaning Small, Middle, or Large model, respectively). This library is *not* linked in automatically at compile time. It must first be specified with the -l option of the compiler. See the *XENIX 286 C Library Guide* for details.

Double Word Ordering Changes

XENIX 286 Release 3.0 features a change in the representation of 32-bit values that makes Release 3.0 compatible with standard Intel byte ordering. However, this change affects the compatibility between XENIX 286 Release 1.0 and XENIX Release 3.0. This section focuses on the design issues that led to the change, its impact on XENIX system users, and a procedure for converting from XENIX 286 Release 1.0 to XENIX 286 Release 3.0.

Background

The original C compiler supported by XENIX 86 and XENIX 286 was developed by Dennis Ritchie at Bell Laboratories. It was originalaly designed in 1973 around the PDP 11/44 architecture. In the late 1970's and early 1980's, the compiler was modified to support Intel's iAPX 8086 microprocessor.

With the first release of this compiler, Microsoft Corporation chose to support the same 32-bit ordering supported under the Ritchie compiler. Meanwhile, with the design of the 8087 Fast Floating Point microprocessor, Intel also standardized a 32-bit ordering for the iAPX 8086 architecture. Intel, however, chose the same byte ordering that Digital Equipment Corporation selected for its VAX family of computers. Consequently, Microsoft 32-bit ordering is incompatible with the Intel 8087, 287, and 386 processors' 32-bit ordering.

In C on a XENIX system, 32-bit values are declared by using the keyword **long.** Figure A-1 shows the internal representation of **long** values in the Microsoft XENIX C compiler.

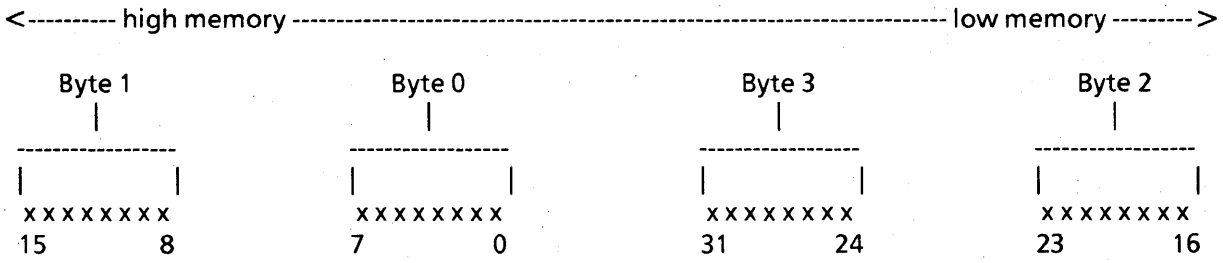

Figure A-1. Microsoft Representation of **long** Values

The Intel representation of 32-bit values appears in Figure A-2.

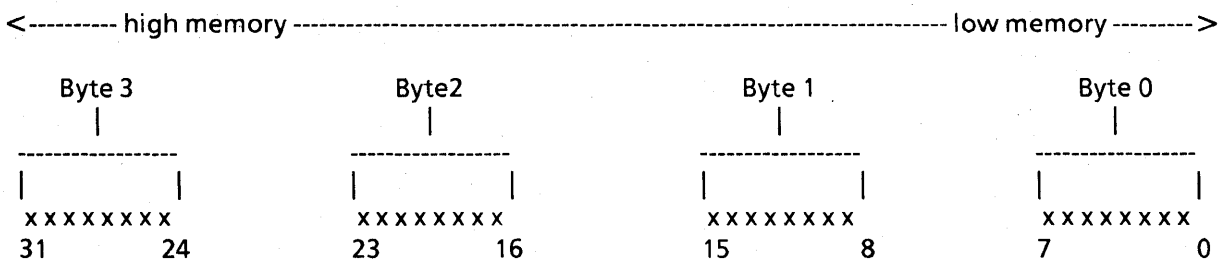

Figure A-2. Intel Representation of **long** Values

For the iAPX 8087 and 287 components to correctly interpret **long** values, the C compiler must perform a sequence of manipulations. For example, in order to provide support for the 8087 math component, the compiler generates instructions to swap the low word (bytes 0 and 1) with the high word (bytes 2 and 3). The 8087 then is able to operate on the representation it expects. Once the 8087 has completed an operation on a 32-bit value, the two words are swapped back.

In 1983, Microsoft and Intel were involved in the design of two strategically important projects. Microsoft was developing a new state-of-the-art compiler construction tool and a new C compiler for XENIX 286. At the same time, Intel began a project to support the Universal Development Interface (UDI) and translators for XENIX 286. During the development period, the two companies discovered the incompatibility. After careful analysis, they decided that Microsoft would make the data representation of 32-bit values compatible with the Intel architecture.

In the short term, this change allows programs compiled on Intel compliers to be linked to programs complied on the new Microsoft C compiler. In the long term, conversion allows both Microsoft and Intel to take full advantage of the 86 family architecture and thereby obtain the highest possible performance.

XENIX 286 Made Compatible with Components

XENIX 286 Release 3.0 from Intel (based on XENIX Version 3.0 from Microsoft) directly supports a representation compatible with that specified for the Intel 86 family architecture. The C compiler and XENIX system software has been modified to generate and interpret **long** values as represented in Figure A-2, shown earlier. However, the redesign of the C compiler produces an incompatibility between XENIX 286 Release 1.0 and XENIX 286 Release 3.0.

Implications of the Change

The change in **long** value representation will affect XENIX users who plan to migrate from XENIX 286 Release 1.0 to XENIX 286 Release 3.0 and beyond. In particular, it will affect the file system, binary programs, and data files on the user's system.

File System

The XENIX file system is defined as the collection of files stored on a Winchester disk or any other nonvolatile storage area. Building a file system requires formatting the disk; partitioning the disk into root area, swap space, user space, and inode locations; and distributing the files throughout the disk while maintaining a history of the files through a defined set of structures.

The XENIX file system contains 32-bit data structures. Therefore, XENIX 286 Release 3.0 cannot interpret an old XENIX file system. When converting to XENIX 286 Release 3.0, users must transport their old file systems.

Object Files

When a C program is compiled, an object file is created. Each object file contains a 32-bit value describing attributes of the executable file. If this object file contains a reference to a 32-bit value, then the object file will not be directly transportable from XENIX 286 Release 1.0 to XENIX 286 Release 3.0.

References may occur directly if the C program contains **long** value declarations, or indirectly if the C program invokes a program or system calls that contain a 32-bit parameter. Examples of system calls that require **long** values are **atol, chsize, lseek, locking, nap, times,** and **ulimit.** Any program that directly or indirectly invokes these system calls will not be binary-transportable. To ensure a reliable system environment, Microsoft and Intel are directing XENIX users to recompile all C programs.

Binary Data Files

Data files may be partitioned into two groups: ASCII data files composed of standard ASCII characters, and binary data files composed of some combination of ASCII characters and integer, real, logical, and/or binary values. ASCII files are completely transportable between XENIX 286 Release 1.0 and XENIX 286 Release 3.0. Likewise, data files that contain no 32-bit values are completely compatible between the two releases. However, any Release 1.0 binary data file that contains **long** values will *not* be compatible with Release 3.0 and must be recreated.

Binary data files are created in one of two ways. A C program may act as a filter, taking an ASCII file as input and producing a binary data file as output (for example, **nroff** or a compiler). A C program may act as an interpreter, taking some terminal interaction as input and producing a binary data file as output (for example, Multiplan and Mistress, two data base programs).

If the program acts as a filter, all inputs to that filter should be maintained and resubmitted. If the program acts as interpreter, a file conversion utility must be generated. This utility would read each 32-bit value, swap the words, then write the 32-bit value back into the same location in the file.

Migration to XENIX 286 Release 3.0

This procedure describes, step-by-step, the procedure for converting from a XENIX 286 Release 1.0 system to a XENIX 286 Release 3.0 system.

1. **Back up user and system files.**

   First, make sure there is *no* risk of losing files, and back up all user and system files with either the **tar** or **dump** utilities. Do not back up any system commands; they will be replaced. Use **tar** to verify that all backed-up files are readable by asking for a "table of contents" readout (**-t** option) for each backed-up diskette. Because the **tar** utility saves files in a format that does not use 32-bit values, it allows the files to be restored on the new system.

2. **Back up system-specific files.**

   To avoid recreating system-specific files, have the system administrator back up all system files, including

   password and group files
   mail files
   uucp files
   /etc/tty and **/etc/ttytype** files
   /usr/lib/crontab file
   /usr/lib/uucp/L* and **/usr/uucp/U*** files
   any configuration files unique to your installation

3. **Clean up user directories and back them up.**

   Delete all binary/object files from each user's directory, then save all remaining files and the file structure with **tar**. As an alternative, save each user's entire file structure and all files (including shell scripts, source files, data files, and data source files) with **tar**.

   Ensure that all storage areas have been backed up. If your system supports two Winchester disks, then make certain that both are mounted and saved.

4. **Install XENIX 286 Release 3.0.**

   Have the system administrator install Release 3.0 by using the software installation facility. During the installation procedure, the XENIX system will ask permission to reformat the Winchester disk. Give that permission: moving from Release 1.0 to Release 3.0 requires reformatting.

5. **Reformat secondary storage areas.**

   If your XENIX system contains a secondary storage medium, reformat it.

6. **Reinstall system-specific files.**

   Using **tar**, install the files backed up in step 2. Doing so will reinstate all user directory areas that resided on the Release 1.0 system.

7. **Reinstall user directories.**

   Have all users reinstall their own files/directories (those saved in step 3). Delete binary/object files as necessary.

8. **Recompile programs.**

   Have all users recompile all their own source files.

9. **Regenerate data files.**

   If the user system contains programs that produce data files with 32-bit values, all users should recreate their data files.

   If the program serves as a filter, then the users should resubmit source input files. If the program serves as an interpreter, then they should generate a set of conversion utilities. (Conversion utilities may be offered by the software vendor.)

   Figure A-3 offers a simple C program that converts the **file.data** file from the old 32-bit format to the new 32-bit format. Programs like the one shown must be written to convert data generated by filter-type creation programs.

```
#include <stdio.h>
#define FROM_CURRENT            1           /* lseek flag */
#define OFFSET      -41     /* offset from end of long word to beginning */

int one_zero;           /* high-order bytes in long file */
int three_two;          /* low-order bytes in old file */

main( )     /* bk January 12, 1985 */
{

     /* This program converts file.data from the XENIX 286 Release 1.0
        format to the XENIX 286 Release 3.0 format.  The long data word
        appears in the first 4 bytes of file.data. */


     int fd;

     /* open data file for reading and writing */

     if ((fd = open("file.data",2)) == -1 {
          printf("Cannot open file %s0,"file.data");
          exit(0);
     }

     /* first seek to long word value
     lseek(fd,offset,flag) flag=0, 1, or 2, see lseek */

     /* in this example, first word in file is a long value */

     lseek(fd,0,0);

     /* read in both words */

     read(fd,&one_zero,2);
     read(fd,&three_two,2);

     /* seek back over word */

     lseek(fd,OFFSET,FROM_CURRENT);

     /* write correct ordering */

     write(fd,&three_two,2);
     write(fd,&one_zero,2);
     close(fd);
}
```

Figure A-3. Sample Conversion Program

Summary

Intel suggests that all users prepare for conversion from XENIX 286 Release 1.0 to XENIX 286 Release 3.0 by maintaining all program sources and inputs to all programs that create binary data files. If your application contains programs that act as interpreters in creating binary data files, you may prefer to write a set of customized conversion utilities to perform the byte swapping.

intel®

APPENDIX B
master FILE TUNABLE PARAMETERS

Aliases

These entries form the alias table:

```
i215    disk
i188    serial
sm      sim
$$$
```

Tunable Parameters

These entries form the tunable parameter table:

PARAMETER	NAME	DEFAULT VALUE
buffers	NBUF	0
sabufs	NSABUF	20
hashbuf	NHBUF	128
inodes	NINODE	100
files	NFILE	100
mounts	NMOUNT	6
coremap	CMAPSIZ	(NPROC * 2)
swapmap	SMAPSIZ	(NPROC * 2)
calls	NCALL	25
procs	NPROC	50
texts	NTEXT	40
clist	NCLIST	120
locks	NFLOCKS	50
maxproc	MAXUPRC	15
timezone	TIMEZONE	(8 * 60)
pages	NCOREL	0
daylight	DSTFLAG	1
cmask	CMASK	0
maxprocmem	MAXMEM	0
shdata	NSDSEGS	25
maxbuf	MAXBUF	192

The kernel-tunable parameters, found at the end of the master file, enable you to "tune" your system to a particular environment. Because they affect data structures in the kernel data segment, you should take care in modifying any of these parameters.

Consider, for example, what might happen if a parameter were increased, causing the kernel data segment to exceed its limit of 64KB. While making XENIX, the **ld** utility will abort the link. In this case, the error message

```
Group "DGROUP" Larger than 64Kbytes
```

warns that you may need to decrease the size of another parameter to ensure that the kernel data stuctures reside within 64K bytes.

buffers NBUF 0

The **buffers** parameter denotes the number of exported buffers in the kernel's buffer pool.

The buffer pool contains two kinds of buffers: system-addressable buffers (inside the kernel data segment) and exported buffers (outside the kernel data segment). Since the kernel reserves three segments for the exported buffers, up to 192 1KB buffers can be configured into the system.

Since the buffers are outside the kernel data segment, the kernel must access the buffers by way of special assembly language procedures that use far pointers, which are a large model feature. For ease of implementation, the kernel accesses all buffers, both system-addressable and exported, in this manner. Thus, the kernel is selectively executing in large model when it manipulates data in the buffers.

If the default (buffers=0) is used, the kernel will help you determine the number of exported buffers required. It begins by computing the minimum number of system buffers required. When the amount of system memory is less than 512KB, no equation is needed: the kernel uses 40 buffers. When the amount of memory is larger than 512KB, the kernel uses this equation

```
buffers = 40+ ( ( amount of memory > 512K ) / 8 ) / 1024
```

to calculate the minimum number of system buffers. Because the number of exported buffers required is application-dependent, the best way to determine the optimal number is to run the application with a number of different configurations and then choose the one that yields the highest performance. Begin by trying 40 to 60 exported buffers for a 512KB system and 180 to 192 exported buffers for a 1MB system. (When the system boots, it displays the number of exported buffers being used; however, this number does not include the number of system-addressable buffers.)

sabufs NSABUF 20

System-addressable buffers (**sabufs**) are buffers directly addressable by the kernel. They reside in the kernel data segment.

All super-blocks and cylinder group structures are placed within the system-addressable buffers. Since the kernel accesses these data structures frequently, accessing data via far pointers would adversely affect system performance. For this reason, super-blocks and cylinder group structures are accessed differently in the kernel data segments.

System-addressable buffers are used for cylinder group structures, so they should be configured for the most memory available. To determine the maximum number of bytes, reconfigure the system until the system no longer announces an **ld** error (meaning that the kernel data segment is too large).

hashbuf NHBUF 128

The **hashbuf** parameter designates the size of the buffer hashing array.

The kernel uses a hashing algorithm to quickly locate a group of buffers, then it searches that group until it finds the required buffer. The algorithm works by determining the buffer header where the search will begin. (The system contains an array of buffer headers, each of which is linked to a group of buffers.)

The size of the array and the number of buffer headers are determined by **hashbuf.** The hashing algorithm, in turn, is determined by the macro **bhash** in the file **sys/h/buf.h.** If **hashbuf** is changed in any way, make sure that **bhash** will still map to all buffer headers in the array.

inodes NINODE 100

The **inodes** parameter indicates the total number of active inodes in the system.

The kernel keeps a cache of all active inodes in an internal inode table. If system usage exceeds the table, then the system displays this error message:

No file

A -1 is returned to the system call that caused the overflow, and the process that executed the system call will have its **errno** variable set to **ENFILE.**

files NFILE 100

The **files** parameter indicates the total number of active file descriptors in the system.

The kernel keeps a cache of all active file descriptions in a table. If system usage exceeds the table, then the system displays this error message:

No file

A -1 is returned to the system call that caused the overflow, and the process that executed the system call will have its **errno** variable set to **ENFILE.**

mounts NMOUNT 6

The **mounts** parameter indicates the size of the kernel's mount table, which is the maximum number of devices that can be mounted on the system at one time.

The kernel keeps a cache of mounted devices in an internal mount table. If system usage exceeds the size of the table, then a -1 is returned to the system call that caused the overflow, and the process that executed the system call will have its **errno** variable set to **EBUSY**.

coremap CMAPSIZ (NPROC * 2)

The **coremap** parameter indicates the size of the coremap array, which keeps track of all available memory in the system.

The array itself is an array of structures. Each structure describes a fragment of free memory. If system memory becomes fragmented enough to overflow the coremap array, then the system displays this error message:

```
coremap or swapmap overflow (xxxx), shutdown and reboot
```

The **(xxxx)** is a pointer to the coremap array in the kernel data segment.

swapmap SMAPSIZ (NPROC * 2)

The **swapmap** parameter signifies the size of the swapmap array, which keeps track of all available swap space.

The array itself is an array of structures. Each structure describes a fragment of free swap space. If the swap space becomes fragmented enough to overflow the swapmap array, then the system displays this error message:

```
coremap or swapmap overflow (xxxx), shutdown and reboot
```

The **(xxxx)** is a pointer to the swapmap array in the kernel data segment.

calls NCALL 25

The calls parameter indicates the size of the callout table, which the timeout function uses to keep track of procedures to be executed at a specific time. In general, the timeout function is used by serial device drivers.

If system usage overflows the callout table, then the system displays this error message:

```
panic: Timeout table overflow
```

The panic condition results because a timeout function is called only within the kernel. The kernel reacts by displaying the error message and suspending processing.

procs NPROC 50

The **procs** parameter defines the total number of active processes in the system.

If the number exceeds the **procs** parameter, then the system returns a -1 to a **fork** system call that caused the overflow. The process that executed the **fork** call has its **errno** variable set to **EAGAIN.**

An overflow usually occurs from the shell, which displays this error message:

```
Cannot fork: too many processes
```

texts NTEXT 40

The **text** parameter defines the size of the text table, which describes a unique process's text and data segments. The table notes whether the segments are located in memory or swap space.

The text table describes the maximum number of unique processes that can be active at any one time. If the table overflows, the system displays this error message:

```
Out of text
```

The **exec** system call is the only system call that detects this error condition. Because there is no return from an **exec** call, the process performing the **exec** is killed.

Since the processes listed in the table have separate text and data segments, a text segment can be shared among processes.

Note that all XENIX utilities have separate text and data segments, as do large- and middle-model processes. Small-model processes must be compiled with the -i option in order for them to have separate text and data segments.

clist NCLIST 120

The **clist** parameter defines the maximum number of **clist** structures, which constitute the buffer pool used for terminal I/O. Refer to the *XENIX 286 Device Driver Guide* for additional information.

locks NFLOCKS 50

The **locks** parameter defines the maximum number of file locks.

If system usage exceeds this number, then the system returns a -1 to the system call that caused the overflow. The process that executed the system call has its **errno** variable set to **EDEADLOCK.** Refer to the description of the **locking** system call in the *XENIX 286 Device Driver Guide.*

maxproc MAXUPRC 15

This is the maximum number of processes a user can create. (A user is defined by a user ID, or UID, contained in the **/etc/passwd** file.)

If a user exceeds **maxproc** number of processes, a -1 is returned to the **fork** system call that caused the overflow. The process that executed the **fork** call has its **errno** variable set to **EAGAIN.**

An overflow usually occurs from the shell, which displays this error message:

```
Cannot fork: too many processes
```

timezone TIMEZONE (8 * 60)

This is the number of minutes west of G.m.t.

pages NCOREL 0

Reserved for future use.

daylight DSTFLAG 1

This flag indicates the timekeeping method the system is currently using.

If the flag is set to 1, the system is operating on daylight savings time. If the flag is set to 0, it is operating on standard time.

The flag is used by the **ftime** system call.

cmask CMASK 0

This is the default **cmask** for all files. For more information, see the description of the system call **umask** in the *XENIX 286 Device Driver Guide* and the **umask** entry in the "Commands" section of the *XENIX 286 Reference Manual.*

maxprocmem MAXMEM 0

This parameter defines the size of memory in kilobytes. (If **maxprocmem** equals 512, for example, then the total system memory is 512KB.)

The kernel performs a memory scan to determine the size of contiguous system memory. The scan stops when it reaches **maxprocmem** or once it finishes scanning all contiguous memory, whichever comes first. If **maxprocmem** equals 0, then the kernel scans 16 Mbytes.

shdata NSDSEGS 25

The **shdata** parameter indicates the total number of shared data segments active in the system.

If system usage exceeds this parameter, then a -1 is returned to the system call that caused the overflow. The process that executed the system call has its **errno** variable set to **EMFILE.** For more information, see the descriptions of the system calls **sdenter, sdget, sdgetv,** and **sdwaitv** in the *XENIX 286 Device Driver Guide.*

maxbuf MAXBUF 192

This indicates the maximum number of exported buffers. (See the entry describing the **buffers** parameter, earlier in this appendix.)

The **maxbuf** parameter can be set to a number between 1 and 192 only; it cannot be set to a number greater than 192. If **maxbuf** is modified, then the **buffers** parameter must also be modified to a number within the **maxbuf** limits.

/sys/conf/xenixconf

The customary way to change tunable parameters is by editing the **xenixconf** file, not by modifying **master.** The **xenixconf** file contains parameters unique to your system, whereas **master** contains only the defaults. Further, any parameters in **xenixconf** override the defaults in **master.**

For example, in a 2MB, 12-user system, these tunable parameters set in **xenixconf** override the defaults set in **master:**

```
inodes    120
files     120
procs     70
sabufs    19
buffers   192
```

As the example indicates, **xenixconf** lists only the parameters that need to be changed.

intel®

APPENDIX C
RELATED PUBLICATIONS

Related Intel Publications

Copies of the following publications can be ordered from

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Overview of the XENIX 286 Operating System, Order Number 174385 -- XENIX history, XENIX uses, basic XENIX concepts, and an overview of other XENIX manuals.

XENIX 286 Installation and Configuration Guide, Order Number 174386 -- how to install XENIX on your hardware and tailor the XENIX configuration to your needs.

XENIX 286 User's Guide, Order Number 174387 -- a tutorial on the most-used parts of XENIX, including terminal conventions, the file system, the screen editor, and the shell.

XENIX 286 Visual Shell User's Guide, Order Number 174388 -- a XENIX command interface ("shell") that replaces the standard command syntax with a menu-driven command interpreter.

XENIX 286 System Administrator's Guide, Order Number 174389 -- how to perform system administrator chores such as adding and removing users, backing up file systems, and troubleshooting system problems.

XENIX 286 Communications Guide, Order Number 174461 -- installing, using, and administering XENIX networking software.

XENIX 286 Reference Manual, Order Number 174390 -- all commands in the XENIX 286 Basic System.

XENIX 286 Programmer's Guide, Order Number 174391 -- XENIX 286 Extended System commands used for developing and maintaining programs.

XENIX 286 C Library Guide, Order Number 174542 -- standard subroutines used in programming with XENIX 286, including all system calls.

XENIX 286 Device Driver Guide, Order Number 174393 -- how to write device drivers for XENIX 286 and add them to your system.

XENIX 286 Text Formatting Guide, Order Number 174541 -- XENIX 286 Extended System commands used for text formatting.

System 286/300 Series Diagnostic Software User's Guide, Order Number 173767 -- how to use the diagnostic software for 286/300 series systems.

Suggested Readings

The popularity of XENIX and other UNIX-like operating systems has caused many new books to appear in the bookstores. You may want to supplement the XENIX documentation with one or more of these books:

- Banahan, Mike, and Andy Rutter. *The UNIX Book*. New York: John Wiley & Sons, Inc., 1983.
- Bourne, S. R. *The UNIX System*. Reading, Mass.: Addison-Wesley Publishing Company, 1982.
- Christian, Kaare. *The UNIX Operating System*. New York: John Wiley & Sons, Inc. 1983.
- Groff, James R., and Paul N. Weinberg. *Understanding UNIX: A Conceptual Guide*. Indianapolis, Indiana: Que Corporation, 1983.
- Kernighan, Brian W., and Rob Pike. *The UNIX Programming Environment*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1984.
- Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1978.
- McGilton, Henry, and Rachel Morgan. *Introducing the UNIX System*. New York: McGraw-Hill Book Company, 1983.
- Sobell, Mark G. *A Practical Guide to the UNIX System*. Menlo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1984.
- Thomas, Rebecca, and Jean Yates. *A User Guide to the UNIX System*. Berkeley, Calif.: OSBORNE/McGraw-Hill, 1982.
- Yates, Jean, and Sandra L. Emerson. *The Business Guide to the UNIX System*. Reading, Mass.: Addison-Wesley Publishing Company, 1984.

intel®

INDEX

intel®

XENIX 286 Installation
and Configuration Guide
174386-001

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

_____

_____

_____

_____

2. Does this publication cover the information you expected or required? Please make suggestions for improvement.

_____

_____

_____

_____

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications are needed?

_____

_____

_____

_____

4. Did you have any difficulty understanding descriptions or wording? Where?

_____

_____

_____

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

(COUNTRY)

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
5200 N.E. Elam Young Parkway.
Hillsboro, Oregon 97123

ISO-N TECHNICAL PUBLICATIONS

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.

SOFTWARE

0043/5K/1184/OSPS