Addendum

XENIX Configuration Kit Version 03.01.01 (Cat. No. 700-3033)

The kernel built by this package is version 03.01.01 and is identical to the kernel for version 03.01.02 except for the version number displayed on startup. If you have upgraded your system to version 03.01.02 and then install this package, XENIX displays the following message:

```
Warning:  This configuration kit is for version 03.01.01,
and you are not running Xenix core system version 03.01.01.
Proceed with the installation?  (yes/no):
```

You can ignore this message and answer "yes" to the prompt.

Please note that it is also acceptable to use this configuration kit with XENIX kernel version 03.01.00.

Tandy Corporation

XENIX 6ØØØ Configuration Kit

Catalog Number 7ØØ-3Ø33

The XENIX 6ØØØ Configuration Kit lets you customize the kernel for your system by modifying the files in the usr/sys/conf directory. To modify the kernel, you must first install the XENIX Software Development Tools of the XENIX Development System.

There are 3 steps to building a new kernel. They are:

- Installing the Configuration Kit
- Building the new XENIX Kernel
- Installing the new XENIX Kernel.

Information on contents of /usr/sys/conf is given at the end of this document. It is **strongly** suggested that you thoroughly read this document before attempting to customize your kernel.

Installing the Configuration Kit

Before installing the Configuration Kit, be sure that the version number of the kit matches the version number of your XENIX system.

1. Log in as root from the console.

2. Be sure that you do not already have a file or directory named /usr/sys. If you do, rename it using the mv command or remove it if you do not want to save it.

3. At the root prompt (#), type:

```
install <ENTER>
```

4. The screen shows the Installation Menu:

```
Installation Menu
-----------------
  1. to install
  q. to quit
```

5. Type 1 and press <ENTER> to install the Configuration Kit.

6. The screen shows:

   Insert diskette in Drive Ø and press <ENTER>

   Insert the Configuration Diskette into Drive Ø, and press <ENTER>.

7. XENIX reads the files off the diskette. When all files are written to the hard disk, the screen shows:

   Installation complete - Remove the diskette, and press <ENTER>

8. Remove the Configuration Diskette from Drive Ø, and press <ENTER>.

9. The screen shows the Installation Menu again. Type q and press <ENTER> to quit.

Your Configuration Kit is now installed. You can proceed to build a new kernel by first modifying the /usr/sys/conf/conf file.

Building the New XENIX Kernel

1. Again, log in as root. At the root prompt, type:

   cd /usr/sys/conf <ENTER>

2. Using any XENIX editor, modify the conf file and any other files you wish to change. Detailed information on the conf file is given later in this document.

3. When you finish editing the configuration files, type:

   make <ENTER>

4. XENIX creates a new file called "xenix" in the /usr/sys/conf directory. XENIX also displays the size of the new "xenix".

5. If XENIX displays an error message, see the CONFIG(CP) and MASTER(F) manual pages in your Development System manuals for more information. Also see the intermediate source file c.c that is produced.

6. Correct any problems and start again from Step 3.

Note: The new kernel that you build will not exactly match the distributed kernel, even if you did not change the configuration. The output of the "size" command will indicate that they are the same size. The file size, as shown by the "ls -l" command, will show 22 bytes difference in size. The code and data loaded are identical.

The kernel that you are running on your system probably has some patches made to it by the /etc/drive and /etc/tz commands which are run during the system initialization procedure. This causes a few bytes of data to be different between the kernel you build and the kernel you are running. You will make these patches to your new kernel during the installation procedure.

Installing the New XENIX Kernel

Before beginning, be sure that all users are logged off the system.

1. Log in as root. At the root prompt, type:

   cd /usr/sys/conf <ENTER>

   Note: After the next step, the ps command will not function properly. The ps and certain other commands use the /xenix file to obtain the addresses of variables in the kernel, and will not work properly if /xenix is missing or does not contain the currently running kernel. They will work again after the installation is complete.

2. The first thing you want to do is save the old kernel, especially if it is one distributed by Tandy or one that you were using successfully. This ensures that you have a good kernel to go back to if there is a problem with the new one. Save the kernel (/xenix) by renaming it. For example:

   mv /xenix /xenix.31 <ENTER>

   If for some reason you do not want to save the old kernel, you can remove it using the rm command.

3. Move the new kernel, created in the previous section, to the root directory. Type:

   mv xenix /xenix.31.new <ENTER>

4. Link the new kernel to /xenix by typing:

   ln /xenix.31.new /xenix <ENTER>

5. Shut down the system using /etc/haltsys:

   sync; sync; sync; /etc/haltsys <ENTER>

   Do not use the shutdown command.

6. Reboot the system. Xenix now uses the new kernel. The ps command should function properly at this point.

7. Use the /etc/drive command to set the appropriate speed for the floppy disk drives. See the "System Defaults" chapter of the System Administrator's Guide to XENIX for information on using drives.

8. Use the /etc/tz command to set the proper time zone. Type:

   /etc/tz <ENTER>

   You are prompted for information on daylight savings time, location (United States or non-United States), and the appropriate time zone. If you are in the United States, you are prompted to enter a selection from a menu for the proper time zone. If you are outside the United States, you are prompted to enter the time as minutes from Greenwich Mean Time.

If you cannot boot with the new kernel, or if it does not perform properly, you can switch back to the old kernel:

1. If you can shut down the system normally, do so. Press the Reset Button.

2. At the boot prompt (Xenix Boot>), type the name of the old kernel previously saved. For example, type:

   xenix.31 <ENTER>

3. When prompted, enter system maintenance mode by entering the root password.

4. At the root prompt, type:

   rm /xenix <ENTER>

5. Link the old kernel to /xenix:

   ln /xenix.31 /xenix <ENTER>

6. Type <CTRL><D> to start the system normally.

You can now fix the configuration and rebuild a new kernel.

More About Configuration

The conf file in /usr/sys/conf specifies the configuration. You should edit this file to reconfigure your kernel. Also included is conf.std. This is an extra copy of the standard distribution configuration. Do not use this file. Use it only as an example of a valid configuration.

For more information on configuration, refer to CONFIG(CP) and MASTER(F) in your XENIX Development System Manuals. However, any information in this document supersedes that in the Development System Manuals. Also, the examples given in CONFIG(CP) and MASTER(F) do not apply directly to your computer. There is an example given later in this document that does apply to your computer.

MASTER(F) refers to a file "/etc/master." This file is /usr/sys/conf/master in your Configuration Kit.

CONFIG(CP) refers to a file "dfile." This file is /usr/sys/conf/conf in your Configuration Kit.

The Configuration Kit is not intended to provide you with the capability to add your own device drivers. However, it is possible to add a line to the /usr/sys/conf/master file for a new device, include it in the /usr/sys/conf/conf file, and add the appropriate object files to the ld command line in the Makefile. Information on writing device drivers is not given with the Configuration Kit and is not available.

More On the conf File

You can look at the /usr/sys/conf/conf file by using the more command:

```
more /usr/sys/conf/conf  <ENTER>
```

The file should look like this:

```
**
*** Standard distribution large memory XENIX configuration file
**
disk    1
console 1
printer 1
screen  1
acu     1
*** note that the minor number Ø377 causes the kernel to autoconfigure
root    disk Ø377
pipe    disk Ø377
swap    disk Ø377 Ø 8ØØØ
*** tunables
timezone (6*6Ø)
maxprocmem 256
```

Part 1:

This part specifies the devices that are present. Disk and console are required. You can remove the others if you do not have the corresponding peripheral. Removing devices saves memory. Remember that the device does not function if it is removed from the configuration.

The following describes the devices that you can remove and how much memory they occupy:

acu — Refers to the dialer devices (/dev/cul* and /dev/cua*). This allows automatic dialing of Radio Shack modems. The driver occupies about 1 3/4 Kbytes.

printer — Refers to the printer devices for the parallel port (/dev/lp, dev/rlp, and dev/clp). The driver occupies about 1 Kbytes.

screen — Refers to the screen-image memory for the console screen (/dev/screen). Some application programs use this device. The driver occupies about 1 Kbytes.

Part 2:

This part specifies the devices to be used for root, pipe, and swap. Normally, you do not want to change root or pipe. However, you might want to change swap. For example, by moving the swap device to a low-activity hard disk, you might increase performance.

We recommend that you do not set the swap device as a floppy disk as this causes very slow performance.

Also, if you wish to change only the size of the swap area on Drive Ø, you do not need to reconfigure the kernel. This is because the standard kernel uses an autoconfiguration option. Autoconfiguration tells XENIX to use the space after the file system on the device from which you booted (Hard Drive Ø or Floppy Drive Ø) as the swap area. To change the swap area, save all your application and user files, reinstall XENIX on your hard disk, and restore the files. During this procedure, XENIX asks if you want to use a non-standard swap area and builds the root file system of the correct size.

To create the swap area on any other drive, specify swap in the form:

swap disk device# offset swapsize

The device# specifies the disk to be used for the swap area. You can use the entire drive or the area left after a file system. The device numbers are:

Hard Disk:

Swap Area	Drive Ø	Drive 1	Drive 2	Drive 3
Entire Drive	Ø4Ø	Ø5Ø	Ø6Ø	Ø7Ø
Space after File System	Ø42	Ø52	Ø62	Ø72

Floppy Disk:

Swap Area	Drive 1	Drive 2	Drive 3
Entire Drive	Ø1Ø	Ø2Ø	Ø3Ø
Space after File System	Ø12	Ø22	Ø32

Disk Cartridge: Cannot be used as a swap device.

Note: Leading Zeroes are significant. You must include them to signify octal numbers.

You can specify autoconfiguration for Drive Ø by using the device number Ø377. If you specify autoconfiguration, the specified swap size is not used. The standard kernel is set for autoconfiguration.

Offset specifies the number of blocks to reserve at the beginning of the disk partition before the swap area. Usually, you specify Ø so disk space is not wasted. Offset is primarily useful for systems that do not have the "space after file system" option available.

If you elect to use the "Space after File System" option, you must first make a file system. You cannot make a file system on drive that you are currently using. That is, the drive cannot be mounted. Also, if it already contains a file system, it will be overwritten with the new (empty) file system.

To make the file system, first use the diskstat command to determine the amount of available space on the disk. Type:

diskstat x <ENTER>

Replace x with the disk drive number. Note the number of blocks diskstat returns.

You must now decide how many of those blocks to use for swapsize and how many to use for the file system. The number of blocks you select must be a multiple of 17 (1 track). You should use at least 2ØØ6 blocks for swapsize. If you have a large system, you should use at least 4Ø97 blocks as swapsize. You must create the file system prior to specifying the swap area. Use the /etc/mkfs command:

/etc/mkfs /dev/hdx nnnnn 9 17 <ENTER>

Replace x with the drive number and nnnnn with the number of blocks for the file system. The file system size is the number of available blocks (returned by diskstat) minus swapsize.

Note: The maximum swapsize that can be used is 32759 blocks (16379K).

If you elect to use the "Entire Drive" option, use the diskstat command to determine the amount of available space on the disk. Type:

diskstat x <ENTER>

Replace x with the drive number. The number returned is the amount of available space you can use for swapsize. You do not have to make a file system on the drive with this option.

After changing the swap device, you must remake /dev/swap so the ps command and other programs will get the correct information when looking at the swap device. You should first save the old /dev/swap:

```
mv /dev/swap /dev/swap.old  <ENTER>
```

Then, use the mknod command to create the new /dev/swap:

```
mknod /dev/swap b Ø1 device#
```

Be sure to use the same device number that you used earlier. Now, type these commands:

```
chown sys /dev/swap  <ENTER>
chgrp sys /dev/swap  <ENTER>
chmod 64Ø /dev/swap  <ENTER>
```

Examples:

```
swap  disk  Ø377  Ø  8ØØØ
```

This is the standard configuration for the swap space. It uses the autoconfiguration mode to use the available space on the boot drive.

Suppose you have a 15 meg hard disk as Drive 2 and wish to use it for the swap space. The available space is 3Ø77Ø blocks. You decide on a swap space of 34ØØ blocks. You then create a file system with the following command:

```
/etc/mkfs /dev/hd2 2737Ø 9 17
```

You can now use the swap specification:

```
swap  disk  Ø62   Ø  34ØØ
```

This example tells XENIX to use 34ØØ blocks after the file system on Drive 2 as the swap space. The file system must already be created. You would now use the commands described earlier to remake the /dev/swap.
Part 3:

This part specifies a number of parameters called "tunables." These parameters let you set defaults for your XENIX system. The following describes each tunable and the default used if you don't specify it:

Name	Description
daylight	Sets daylight savings time to on or off. Specify 1 if daylight savings applies or Ø if it does not. It is best to use /etc/tz to set "daylight" and "timezone" because tz sets the environment variables that XENIX uses when displaying the time and date. This parameter does not affect the size of the kernel. Default = 1.
timezone	Specifies the number of minutes west of Greenwich Mean Time. Specify: 8*6Ø for Pacific Standard Time 7*6Ø for Mountain Standard Time 6*6Ø for Central Standard Time 5*6Ø for Eastern Standard Time It is best to use /etc/tz to set "daylight" and "timezone" because tz sets the environment variables that XENIX uses when displaying the time and date. This parameter does not affect the size of the kernel. Default = 8*6Ø.

maxprocmem Sets the maximum amount (Kbytes) of memory that 1 process can use. This prevents a process from using the entire system memory. We recommend that you not change this parameter. However, if you do, be sure that your swap space is at least 4 times the amount you set for "maxprocmem."

The maximum amount of memory that any process can use is the total user memory (given at boot time) If you set "maxprocmem" too high, XENIX uses only available memory. If you set "maxprocmem" smaller than 256, you might not be able to run some programs or even boot the system.

This parameter does not affect the size of the kernel. Default = 256.

maxproc Sets the maximum number of processes any 1 user-id can run at 1 time. This number includes all processes, regardless of the number of terminals the user might be using. This parameter does not affect the size of the kernel. Default =15.

procs Sets the maximum number of processes that can run in the system at 1 time. This number includes all system processes: swapper, /etc/init, /etc/cron, /etc/update, getty's (enabled lines), cron processes, etc. If a user program or the shell returns a "No more processes" message, you need to increase either "procs" or "maxproc." Each process structure occupies approximately 46 bytes. Default = 6Ø.

inodes Sets the maximum number of different files that can be open at 1 time. This number includes mounted file systems, current working directories, current root directories, special device files, semaphores, pipes, shared memory segments, tty devices, directories, and ordinary files. Several processes opening the same file count as only 1 inode structure. If you receive a console message, "Inode table overflow", increase the "inode" value. If you receive a console message, "Out of inodes on dev xx/yy", do not increase the value for "inode." This means that the disk ran out of space to store inodes on the disk. Increasing the value of "inode" does not solve this problem. Each inode structure occupies approximately 78 bytes. Default = 1ØØ.

files — Sets the maximum number of opens that can run at 1 time. Each open to a file is counted as a different file structure. However, open files passed from a parent process to a child process use a single file structure. Current working directories, current root directories, and mounted file systems do not require a file structure. However, if a program opens a directory, it is counted as a file structure. Pipes require 2 file structures: 1 for each end of the pipe. If you receive the console message, "no file" or the error message, "File table overflow", increase the value for "files." Each file structure occupies approximately 1Ø bytes. The limit for "files" should be equal to or slightly greater than the value set for "inodes." Default = 1ØØ.

mounts — Sets the number of file systems that can be mounted at any 1 time. This number includes root, but does not include devices that are not currently mounted (for example, swap). Different partitions of the same disk that contain different mounted file systems count as separate mount structures. Each mount structure occupies approximately 14 bytes. Default = 8.

texts — Sets the maximum number of **different** shared-text programs that can run at 1 time. Programs with the "sticky bit" (chmod u+t) set count even if it is not currently executing. Almost every distributed XENIX program is shared-text, as well as various shells (sh, csh, vsh and tsh). Shell scripts are not shared-text. Several users running the same program simultaneously use 1 text structure for that program. The console message, "out of text", indicates that the value for "texts" needs to be increased. Each text structure occupies approximately 14 bytes. Default = 4Ø.

locks — Sets the maximum number of file locks that can be in effect at 1 time. See LOCKING(S) for more information on file locks. Very few programs supplied with the XENIX Runtime (core) or Development system use file locking. Some applications might extensively use file locking. Each lock structure occupies approximately 2Ø bytes. Default = 2ØØ.

calls — Sets the maximum number of time delays that can be pending in the kernel at any 1 time. Normally, you do not need to increase this value unless you significantly increase the value for "procs." The console message, "Timeout table overflow", indicates that you need to increase the value of "calls." Each callout structure occupies approximately 12 bytes. Default = 25.

coremap
swapmap — Contain information about free memory and free swap space, respectively. Do not change these values. Default = NPROC/2.

sabufs — Sets the number of disk buffers available in the disk cache. Increasing the value of "sabufs" might increase the performance of your system. Also see hashbuf. Each disk buffer occupies approximately 554 bytes. Default = 62.

hashbuf — Sets the number of hash buffers available. If you change "sabufs", you should also change the value of "hashbuf" to the next power of 2 above the value of "sabufs". For example, if you set "sabufs" to a value between 64 and 127 (inclusive), set "hashbuf" to 128. Each hashbuf occupies approximately 12 bytes. Default = 64.

clists — Sets the maximum number of clists for the system. Clists store characters that are waiting for input to or output from a terminal. Increasing the number of clists allowed increases the number of character that the kernel can hold before the system start to lose characters. Remember that transferring a continuous high-speed stream of data with no character loss is only possible with flow control (^S / ^Q) or a positive acknowledgement protocol arranged between the other system and the program running on your XENIX system. Each clists occupies approximately 3Ø bytes and holds from 1 to approximately 24 characters. Default = 149.

hz — Sets the clock rate of the system. The clock rate is specified in "ticks" per second. Because the kernel determines the clock rate (either 3Ø for 6ØHz power or 25 for 5ØHz power) during initialization, there is no need to set this variable. Anything you set this to will be ignored. Default = 3Ø.

Notes:

- The variables daylight, timezone, maxprocmem, and maxproc do not affect the size of the kernel.

- The variables procs, inodes, files, mounts, texts, locks, and calls specify the size of various system tables. Making them larger (if you are not having any problems) does not increase system performance. In fact, the larger kernel can decrease system performance. If you are having problems with table overflows, check what was happening at the time the problem occurred. It is often better to avoid running tasks that use too many processes or resources than to attempt to reconfigure the kernel.

- The variables sabufs, hashbufs, and clists specify the size of various system tables. Increasing their value might increase system performance. However, making the kernel larger reduces the amount of user memory available.

- The kernel's size is rounded up to the next multiple of 4Ø96 bytes. If the new kernel is just over a 4Ø96 byte boundary, you might consider reducing the value of some variables to make the kernel smaller. If you cannot reduce the kernel, then you can take advantage of the wasted space and increase the value of some variables.

- Do not reduce any of the table size variables to zero. Some of the tables require more than one entry for the system to boot, and most must be at least 2Ø% of their default values to run anything practical.

Example

Suppose that your system has 1 hard drive, 512K of memory, and no terminals. The applications you run include large MBASIC programs that need as much memory as possible. You do not have a modem or need the screen device. Speed is a consideration, but the system must be able to handle large processes. Therefore, you want to raise the number for "maxprocmem" to 42Ø.

Your system's users also run complicated shell scripts that sometimes run over the process-per-user limit of 15. Therefore, you raise the limit of "maxproc" to 25.

After raising those variables, you decide to cut the size of most system tables in half. However, because file locking was apparently never used, and there will not be another user to lock out, you want to cut the number of locks available to 1Ø.

The resulting configuration is:

```
**
*** Custom Xenix configuration file for single-user system
**
disk    1
console 1
printer 1
*** note that the minor number Ø377 causes the kernel to autoconfigure
root    disk  Ø377
pipe    disk  Ø377
swap    disk  Ø377  Ø  8ØØØ
*** tunables
timezone (6*6Ø)
maxprocmem 42Ø
sabufs 31
hashbuf 32
inodes 5Ø
files 5Ø
mounts 4
procs 3Ø
texts 2Ø
clists 75
locks 1Ø
maxproc 25
```

After you build this kernel, you notice that the kernel's size is only 196 bytes over a 4Ø96 byte boundary. By reducing "sabufs" by 1, you can gain 4K of user memory. Reducing the number of buffers costs some speed, but if large processes cannot run at all, printing error messages more quickly does not help. Your system, in this case, would benefit from more memory and less buffers. So change "sabufs" to:

```
sabufs 3Ø
```

After building this kernel, you notice that the kernel is approximately 28K smaller than the distribution kernel. However, this did not quite reach the goal of running 42ØK processes. The actual user memory available, is:

```
512K - 1Ø4K = 4Ø8K
```

This is the largest size user process that can be run. If it turns out that 4Ø8K is not large enough, you will have to reduce the size of the kernel further.