*NEXTSTEP In Focus*, Spring 1993 (Volume 3, Issue 2).


# The Big SetupÐ
# Behind the Scenes of Configure

Matt Watson

When you install NEXTSTEP on an Intel-based computer, you spend a few minutes (maybe more than a few) specifying which devices your system has and how they're set up. Then you click Save and continue installing.

But what happened to all those settings you so painstakingly selected? How did the installation program know which defaults to pick, and where did it store the list of devices you said you have? What if you change your system later and add more devicesÐhow will you update that list?

This article takes you behind the scenes of the Configure application, which handles these details for you, and shows you how and where the configuration information is stored and how to change it all later.

## DRIVER BUNDLES AND CONFIGURATION FILES

NEXTSTEP for Intel Processors stores configuration information for the system as a whole and for each deviceÐlike the mouse, video monitor, network card, external drives, and so onÐin a special directory called **/usr/Devices**. In this directory are driver bundles that contain the device configuration settings. Each system has a bundle called **System.config** that configures the whole system, plus a bundle called *Driver***.config** for each type of device, where *Driver* is a type of device or a device name.

For example, for an average system, **/usr/Devices** could contain the following

bundles:

```
ATIGXPro.config
ATIUltraPro.config
Adaptec1542B.config
Beep.config
BusMouse.config
CirrusLogicGD542X.config
DPT2012.config
EpsonWingine.config
EtherExpress16.config
Etherlink3.config
Floppy.config
IDE.config
JAWS.config
LGIWingine.config
PS2Keyboard.config
PS2Mouse.config
ParallelPort.config
ProAudioSpectrum.config
QVision.config
S3.config
SMC16.config
SerialMouse.config
SerialPorts.config
System.config
TokenExpress.config
TsenglabsET4000.config
VGA.config
```

The directory **/NextLibrary/Devices** is a link to **/usr/Devices** that you can see without being in UNIX Expert mode.

# WHAT'S IN THE DRIVER BUNDLES

Each driver bundle (including **System.config**) can contain the following files and directories:

```
Default.table
Driver.table
Instancen.table
DriverInspector
Language.lproj/
   Custom.nib
   Localizable.strings
```

```
   Info.rtf
Driver_reloc
```

You don't need to know what's in each file to configure the system, since Configure does the work for you. However, if you're curious about how these files are used, here's a summary. (For details, see the Driver Kit developer documentation.)

**Default.table** is a commented, read-only file that gives the default configuration settings for a generic device. *Driver***.table** is a commented, read-only file that gives the default settings for a particular manufacturer's device. Both of these files are optional, but at least one exists in every bundle. Configure uses them to build **Instance***n***.table** files, which contain specific configuration information for each instance of the device you have. For example, if you have two of the same device, Configure makes two files called **Instance0.table** and **Instance1.table**.

For each language you use, **Localizable.strings** contains the text strings that applications display about the device. For example, it includes the name of the device as it appears in the list of devices in Configure. **Info.rtf** (or **Info.rtfd**) contains the text and images displayed when you click the Info button in the Inspector panel.

The *Driver*_**reloc** file is the executable file of the device driver. The *DriverInspector* binary is the executable file for the Inspector panel; its name is the same as the *Driver* part of the bundle name. **Custom.nib** is the nib file for the Inspector panel.

### Indirect devices

Some device drivers don't control a card, but instead control something that's attached to a card. They're called *indirect device drivers*. For example, if you have a SCSI disk attached to a SCSI controller, the disk is controlled by an indirect device driver, and the controller by a direct device driver. An indirect device driver communicates with its hardware indirectly, by sending requests to its associated direct device driver.

Some indirect device drivers don't need any configuration information other than an **Instance***n***.table** file. The bundle for an indirect device driver can contain just **Instance***n***.table** files, *Language***.lproj** directories, and an executable file.

## The System configuration bundle

The **System.config** bundle is special in a few ways. Its **Instance0.table** has default configuration information for the system as a whole. For example, it specifies which device drivers to load at boot time and which to load later. It can also identify the bus architecture of the machine (such as ªISAº or ªEISAº) and the manufacturer's name and model number for the computer (like ªDell 450/DE2º, ªCompaq Q/Visionº, or ªISA Bus Systemº). The example below shows a sample **Default.table** from a **System.config** bundle.

*Default.table in System.config shows the default configuration for the system.*

```
"Version" = "1.0";                          /* Version of this file */
"Machine Name" = "Dell 450/DE2";            /* Manufacturer's name */
"Bus Type" = "EISA";                        /* Bus architecture */
"Boot Drivers" = "IDE Floppy AHA1542";      /* Drivers to load at boot time */
"Active Drivers" = "VGA SystemSerial";      /* Bundles to load after booting*/
"Kernel" = "sd(1)mach_kernel";              /* Pathname of the kernel to load */
"Kernel Flags" = "-s rootdev=sd1a nbuf=64"; /* Additional kernel settings */
```

## Configuration keywords

In the **Default.table** and **Instance*n*.table** files, these keywords indicate the devices' settings:

| Keyword | Description |
|---|---|
| Title | The name of the bundle |
| Family | The general category of the device: "Audio", "Network", "Parallel", "Pointing Device", "Printer", "Serial", "SCSI", "Video", or "Other" |
| Location | The location of the device, like "Slot1" or "System Baseboard" |
| Instance | The instance number of the file, like "0", "1", "2" |
| Version | The format version of the file, such as "3.1" or "2.245" |
| Driver Name | The name of the driver class in the Driver_reloc file, like "ATIDisplay" |
| Valid IRQ Levels | IRQ lines that the device could use (in base10 format, separated by single spaces), such as "7" or "614" |

| | |
|---|---|
| IRQ Levels | IRQ lines reserved for the device |
| Valid DMA Channels | DMA channels that the device can use (in base10 format, separated by single spaces) such as "3" or "27" |
| DMA Channels | DMA channels reserved for the device |
| Memory Maps | Memory ranges reserved for the device (in base16, separated by single spaces), such as "0x0D200-0x0D3FF 0x0E400-0x0E7FF" |
| I/O Ports | I/O port ranges reserved for the device (in base16, separated by single spaces), such as "0x280-0x28F" |

## WHAT GOES ON

The Configure application is actually not very complicated. Configuring is basically a matter of creating **Instance*n*. table** files with the right settings, and ensuring that no two devices conflict.

When you configure a device, Configure looks in the device's **Default.table** and *Driver***.table** for information such as the name and family of the device, default memory maps, and so on. It then creates an **Instance*n*.table** with this information and the device settings you selected for the device, such as IRQ lines and I/O ports.

If it discovers conflicts between your settings for devices, such as two devices with the same IRQ line, it doesn't create the **Instance*n*. table** files but instead alerts you of the problem and lets you fix it. Configure is also aware of subtle rules about the devices, such as the order in which they must be listed in the **Instance0.table** in **System.config**.

## RECONFIGURING IN A PINCH

Normally, you use the Configure application to add, remove, and reconfigure devices. However, there might be a time when you can't use Configure. For example, if something goes really wrong with your devices and you can't boot the system, you can boot in single-user mode and edit the configuration files by hand

using a UNIX editor.

Before you jump into manipulating the driver bundles, though, try booting your system with a default configuration. To do this, when the system starts, type the following at the boot prompt:

```
-s config=Default
```

This causes the boot program to use **Default.table** in **System.config** to try to boot, which usually works. Once you've booted, use Configure to fix the rest of the configuration.

If you think you know what's wrong in the bundles and don't want to go through the whole configuring process, then you can edit the bundles by hand. Check with NeXT Support first to make sure what you plan to do won't cause a problemÐconfiguring has lots of ªrules of thumbº and you might not know all the effects of a change. Then, type the following at the boot prompt:

```
mach_kernel -s
```

This boots the system in single-user mode. You can then use a single-user mode editor (such as **sed**) to edit the configuration bundles.

## THE SAFEST WAY TO CONFIGURE

Although you can edit the configuration bundles in **/usr/Devices** by hand, using the Configure application saves you the trouble, prevents easy mistakes like typos in an address range, and makes sure there are no device conflicts. So, it's the easiest and safest way to set up a new device. For more information on using Configure, see your installation guide.

*Matt Watson is a Software Engineer specializing in configuration and installation tools for NEXTSTEP for Intel Processors. You can reach him by e-mail at* ***Matt_Watson@next.com****.*