KCMS Calibrator Tool Loadable Interface Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806-2919–10
February 2000

Contents

Preface

Introduction

The *KCMS Calibrator Tool Loadable Interface Guide* describes how to create a dynamically loadable device handler to be used in calibrating devices. Your colorimeter software interacts with the Kodak Color Management System (KCMS™) Calibrator Tool to gather the color measurements needed to correct ICC format profiles. The measurement data is then used by the KCMS framework to achieve the device independent display of color images. Currently calibration of color monitor devices only is supported.

This guide is part of the software development kit (SDK) portion of the KCMS software product.

Who Should Use This Book

This guide is intended to be used by programmers who are writing dynamically installable modules (for example, colorimeter device handlers) that provide KCMS Calibrator Tool with color measurement data. This guide assumes its readers are familiar with the KCMS color management software and the colorimeter hardware.

Before You Read This Book

Before reading this guide, you should

- Read the *KCMS Application Developer's Guide*. The *KCMS Calibrator Tool Loadable Interface Guide* assumes you understand the color concepts (such as characterization, profiles, and calibration) described in the Developer's Guide. The Developer's Guide also describes the KCMS API library and provides background information on the interface between Calibrator Tool, the KCMS library, and color profiles.
- See the on-line `SUNWrdm` packages for information on bugs and issues, engineering news, and patches. For Solaris™ installation bugs and for late-breaking bugs, news, and patch information, see the *Solaris 7 (Intel Platform Edition) Installation Library* and the *Solaris 7 (SPARC Platform Edition) Installation Library*.
- (SPARC™ systems) Consult the updates your hardware manufacturer may have provided.

How This Book Is Organized

This guide is organized into the following chapters:

Chapter 1 is an overview of color calibration. Read this chapter to understand the main software components involved in calibration. The chapter summarizes how Calibrator Tool (which is executed with the `kcms_calibrate(1)` command) interacts with the dynamically loadable module to obtain calibration data. (For details on `kcms_calibrate(1)`, see the man page description.)

Chapter 2 describes calibration from the end-user perspective and the action of the underlying sample software.

Chapter 3 details how you insert and delete `OWconfig` entries for dynamically loadable calibration modules.

Chapter 4 touches on the rationale Calibrator Tool uses to gather measurement data.

Chapter 5 describes the structures Calibrator Tool uses for measurement data.

Chapter 6 alphabetically presents and describes the interfaces a loadable module uses to interact with Calibrator Tool.

Glossary defines words and phrases used in this guide.

Related Books

Sun Publications

For information using KCMS Calibrator Tool see the *OpenWindows Advanced User's Guide*. The section entitled "Calibrating Your Monitor," in Chapter 10, "Customizing Your Environment," contains information on how to calibrate your monitor with Calibrator Tool.

For basic information on color concepts and KCMS, see the first two white papers listed in Table P–1. The other two white papers provide background information on your viewing environment. These files are located online in the `/usr/openwin/demo/kcms/docs/` directory.

TABLE P–1 KCMS White Papers

File Name	Title
`kcms-wp.ps`	*Introduction to the Kodak Color Management System*
`kcms-wp-solaris.ps`	*Kodak Color Management System*

X Window System Information

The following X Window System manuals are available through SunExpress or your local bookstore.

- *Xlib Programming Manual*, O'Reilly & Associates, Inc.
- *Xlib Reference Manual*, O'Reilly & Associates, Inc.
- *X Window System, Third Edition*, Digital Press

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com[SM] Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

What Typographic Conventions Mean

The following table describes the typographic changes used in this book.

TABLE P–2 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename.*
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide.* These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	#

Naming Conventions

In this guide, *KCMS* refers to the Kodak Color Management System. The names of calibration functions and data structures consist of the prefix string ‘KCMS’ plus other significant words that suggest what the function or data structure does. To illustrate, `KCMSCMonInit()` is a function name consisting of four significant words:

- KCMS
- C
- Mon
- Init

Table P–4 describes each significant word.

TABLE P–4 Naming Conventions

Significant Word	Meaning
KCMS	Kodak Color Management System
C	Calibrator

TABLE P–4 Naming Conventions *(continued)*

Significant Word	Meaning
Mon	Monitor device
Init	Initialization function

Note - The prefix strings '*KCS*' and '*kcs*' also are used to refer to the Kodak Color Management System product in related documentation.

New Features

This guide introduces the following new KCMS features.

Features Added With KCMS 1.0.1

Support for a New Loadable Driver

KCMS includes a loadable driver for the X-Rite DTP92 colorimeter. The driver is very similar to the `colorsense.c` module described in this guide. It supports the X-Rite colorimeter, which connects to the serial port to obtain color measurement data.

`OWconfig` File Modification

The procedure for updating the `OWconfig` file has changed. Instead of editing the file, you now use an interactive program called `OWconfig_calibrate` to insert and delete entries.

Features Added With KCMS 1.1

KCMS 1.1 supports multithreaded programs.

CHAPTER 1

Overview

In This Chapter

This guide describes how to create a dynamically loadable module that interfaces with Kodak Color Management System (KCMS) Calibrator Tool to calibrate color devices.

This chapter provides an overview of calibration. The chapter

- Introduces color calibration and KCMS Calibrator Tool
- Identifies the platforms that support calibration
- Summarizes the interaction between your dynamically loadable module and Calibrator Tool

Color Calibration

KCMS color calibration uses software to adjust the output of a color device for accurate color reproduction. Currently KCMS Calibrator Tool supports calibrating color monitor devices only. Monitor calibration is accomplished by displaying a programmed sequence of test colors and measuring the output of the display. The KCMS library then computes color correction factors necessary to compensate for discrepancies between the programmed colors and those actually displayed, and it updates profiles with the resulting data.

About KCMS Calibrator Tool

Purpose of Calibrator Tool

KCMS Calibrator Tool is an interactive program designed to allow users to calibrate their own monitors to later display color-corrected data. The Calibrator Tool program is executed by the `kcms_calibrate(1)` command. (For details on `kcms_calibrate(1)`, see the man page description.)

KCMS Calibrator Tool obtains color measurements from the measurement gathering device handler. It uses the data it collects to update the monitor's profile. A *profile* describes a device, including its type, chromaticities, and generic color response values.

KCMS software uses profiles to achieve the device-independent display of color data. The values KCMS Calibrator Tool writes to profiles represent the color response of the specific monitor from which the measurements were taken. The result is data whose color values are adjusted for that monitor to provide accurate color that is more consistent across different device types and under different viewing conditions.

(For details on the C-function interface between Calibrator Tool, the KCMS color management library, and the profiles, see the *KCMS Application Developer's Guide*.)

When one of the updated profiles is later used to display an image on the monitor, updated color data are used in place of the monitor profile generic data. The new profile more accurately represents the color image viewing conditions, monitor settings, and the state of the monitor (such as brightness and contrast). As such, the image rendered is a significant improvement over the one created by the generic profile.

This color correction process is similar to gamma correction. It can be more accurate, however, because it represents color correction at a finer level of detail than a single gamma value.

Colorimeter Software

You provide the device handling software for your colorimeter to measure the color monitor response. Using software functions, Calibrator Tool loads your module dynamically as a shared library. This guide refers to the user-written colorimeter device handler sample software as the *loadable module* to distinguish it from the KCMS Calibrator Tool (*main program*) with which it interacts.

Platforms and Support

KCMS Calibrator Tool currently calibrates color monitors only and is supported on workstations running the Solaris environment. The tool requires a color frame buffer and a color monitor. The code for one of the two sample loadable modules provided describes an interface to a colorimeter.

The ability to add loadable modules to Calibrator Tool is part of the KCMS portion of the Solaris SDK.

Features

Calibrator Tool provides the following features:

- It supports different measurement methods (with and without a colorimeter)
- It allows the end user to select from among several specific monitor devices to calibrate
- It automatically configures a system to use X Window System Version 11 visuals
- It provides error and status handling information
- It is internationalized using standard SunSoft utilities

Note that, although Calibrator Tool and the sample loadable modules provided with the KCMS portion of the SDK implement their GUI using Motif library routines, this is not a requirement for your loadable module. It can use alternate GUIs.

Installation

The `kcms_calibrate(1)` command is installed in the `/usr/openwin/bin` directory and the manual page, in the `/usr/openwin/man/man1` directory.

How Calibrator Tool Works With A Loadable Module

Software Components

There are two main software components to calibration: Calibrator Tool (main program) and one or more OEM-written loadable modules. Calibrator Tool is started with the `kcms_calibrate(1)` command. Then it works together with loadable modules to obtain color data and to update color profiles.

Calibrator Tool Responsibilities

Calibrator Tool obtains measurements to update a monitor's color profile. It obtains the data it needs from the loadable software module.

Calibrator Tool obtains the measurement data through two data structures that it passes to the loadable module. It allocates and deallocates the memory for the structures and sets up their formats based on the number and types of X11 visuals that the X Window System supports. Using standard X11 routines, it determines which X visuals are supported on the system. Then it creates a profile copy for each visual. Finally it hands off this information formally packaged in the data structures to the loadable module.

Once it retrieves the measurement data from the loadable module, Calibrator Tool updates the X visual profile copies on the local machine.

Loadable Module Responsibilities

The loadable module doesn't need to know anything about the color profiles or where they are stored. It simply performs the required measurements and fills out the data structures passed in by Calibrator Tool. Then it returns the information to the main program.

This is a simplification, of course. The module uses some mechanism for obtaining the calibration data from the monitor device. The KCMS portion of the Solaris SDK provides two sample dynamically loadable modules. One module (`colorsense.so.1`) requires the use of a hardware colorimeter connected to a serial port on the local system to obtain color data measurements. The other module obtains measurement data without a colorimeter. In either case, the module provides a GUI for interaction with an end user.

Note - KCMS includes a loadable driver for the X-Rite DTP92 colorimeter. The driver is very similar to the `colorsense()` module described in this guide. It supports the X-Rite colorimeter, which connects to the serial port to obtain color measurement data.

Calibrator Tool and Loadable Module Interaction

Calibrator Tool and the loadable module interact with each other using three application programming interface (API) functions. These are

- `KCMSCMonInit()`
- `KCMSCMonMeasure()`
- `KCMSCMonClose()`

Each programming interface identifies a phase (initialization, data collection, or closure) in the interaction between Calibrator Tool and the loadable module. To indicate the success or failure of a phase, each module returns status information. A zero status value indicates success; any nonzero value indicates that an error has occurred. If a nonzero status value is returned at any point in the calibration process, the loadable module application is terminated and Calibrator Tool displays a status message. (For details on Calibrator Tool status messages, see the section entitled "Customizing Your Monitor" in Chapter 10, "Customizing Your Environment," in the *OpenWindows Advanced User's Guide*.) The loadable module may have its own custom error and status messages.

The following summarizes the three phases in the interaction between Calibrator Tool and the module.

Initialization

During initialization, Calibrator Tool takes a handle to the dynamically opened module and accesses the KCMSCMonInit symbolic name using `dlsym`(3X). If it finds the name, it initializes and allocates space for the visual data.

The tool calls `KCMSCMonInit()` to allow the loadable module to initialize the serial port for the hardware colorimeter (if it uses one to take measurements), to start its own GUI-based application, and to make any additional preparations necessary to measure monitor data.

Note - The sample modules described in this guide set up the application user interface during initialization using included source code from TeleUSE (Motif GUI builder) and the X Toolkit library (available with Solaris 2.5 and later). (Refer to the `colorsense()` and `XSolarisVisualGamma()` source files.) Be aware that you are not required to use this code in the design of your module. The examples are an implementation only. All or parts of them can be used as a template, depending on your needs.

Data Collection

Calibrator Tool initializes and sets up the data structure to gather data. It makes standard X11 Window System calls to establish the number and types of X visuals for which it needs measurement data. The function's interface structure includes the array of visuals supported by the system, which is provided by the main program, and the matching array of measurements, which is supplied by the loadable module.

Calibrator Tool calls `KCMSCMonMeasure()` to "fill in" the measurement data.

Closure

If the loadable module successfully supplies the data measurements, it returns status value 0 (status OK) to Calibrator Tool. Calibrator Tool then calls `KCMSCMonClose()` to allow the loadable module to free any resources it is using, for example, to unlock the serial port and to close the associated file descriptor.

If, however, the status value returned is any other nonzero value, Calibrator Tool disregards all previously obtained data and informs the end user that calibration has not properly completed. In this case, the calibration process must be restarted from the very beginning with another call to the loadable module.

CHAPTER 2

Calibrating A Monitor

In This Chapter

Chapter 1 introduced the major software components involved in calibration. This chapter describes calibration from the end-user perspective. The end user starts calibration and provides the necessary parameters (such as the device type, what module to load, and various other options for performing the calibration).

Note - This chapter describes two “loadable” source modules provided with the SDK that work with Calibrator Tool to gather measurement data: `colorsense( )` and `XSolarisVisualGamma( )`. To use the `colorsense( )` module, the end user must purchase the `colorsense( )` colorimeter device. `XSolarisVisualGamma( )` presents the same GUI but does not require a colorimeter device.

The calibration process this chapter describes is how the sample loadable module works with Calibrator Tool. This method of obtaining measurement data may or may not apply to the design of your loadable module.

The chapter is meant to provide you with a better understanding of the sample loadable module. To accomplish this, it refers back and forth between the end user actions and those performed by the underlying software.

Before You Read Further

Before you read this chapter, you should review the section entitled “Calibrating Your Monitor” in Chapter 10, “Customizing Your Work Environment,” in the *OpenWindows Advanced User’s Guide*. That section provides additional information on calibration not repeated in this chapter, including

- General calibration concepts
- How to adjust the viewing environment before calibrating a monitor
- How to run Calibrator Tool

Revisiting the Calibrator Tool User Interface

Recall that when the end user starts Calibrator Tool, a setup window is displayed. This setup window is shown in Figure 2–1.

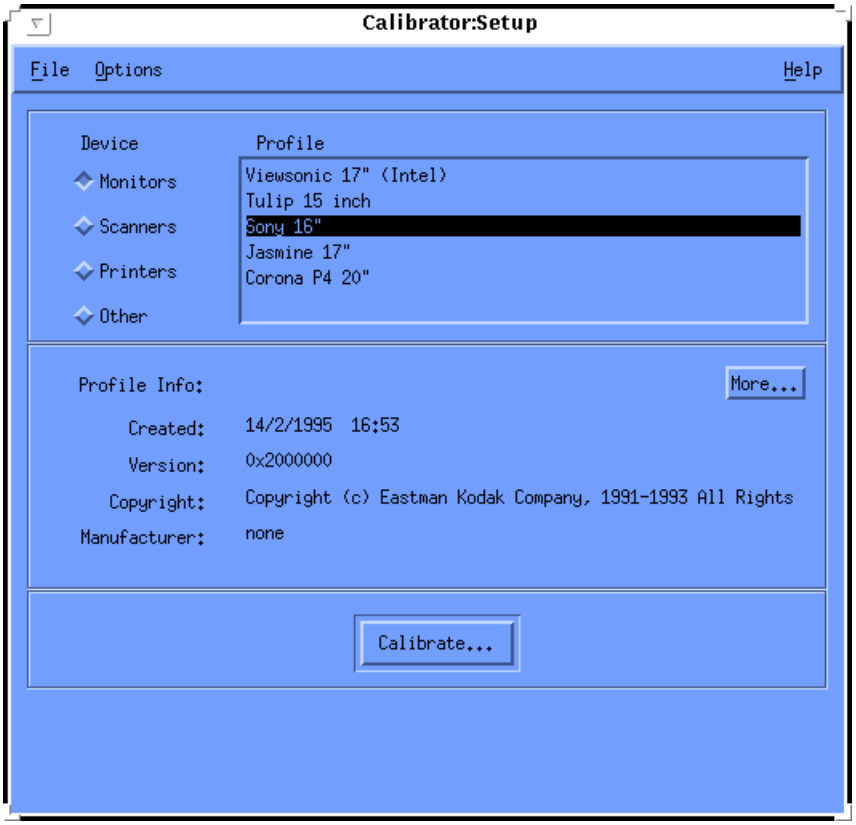

Figure 2–1 Calibrator Tool Setup Window

Table 2–1 summarizes the end-user actions and the actions of the underlying software that lead up to loading the `XSolarisVisualGamma()` or `colorsense()` module into memory.

TABLE 2–1 Events Leading Up To Loading Module

End-User Actions	Underlying Software
Starts Calibrator Tool by typing **`kcms_calibrate`** at the command-line prompt and pressing Return.	Calibrator Tool displays the window in `/usr/openwin/bin`. (See Figure 2–1.)

Selects Monitors as the Device type. (See Figure 2–1.)	By default, Calibrator Tool searches the following predefined libraries for monitor profiles:
	`/usr/openwin/etc/devdata/profiles`
	`/etc/openwin/devdata/profiles`

TABLE 2–1 Events Leading Up To Loading Module *(continued)*

End-User Actions	Underlying Software

Selects the monitor profile (for example Sony 16). (See Figure 2–1.)	KCMS library functions open and obtain the generic device description of the user-selected monitor. For each X11 Window System visual supported by the frame buffer, KCMS library functions make a copy of the generic profile from either of the following:
	`/usr/openwin/etc/devdata/profiles`
	`/etc/openwin/devdata/profiles`

Clicks on Calibrate. (See Figure 2–1.)	Calibrator Tool displays the Calibrator:Devices popup window. (See Figure 2–2.)
Selects the `XSolarisVisualGamma()` device, for example, and clicks on Load.	The server dynamically loads the `XSolarisVisualGamma()` module into memory shared with the Calibrator Tool main module. (The module is expected to be installed in the directory `/usr/openwin/etc/devhandlers`.) Then the loadable module displays its own GUI. (See Figure 2–3.)

The `/usr/openwin/etc/devdata/profiles` directory contains the names of files describing the color characteristics of the generic devices. Currently only monitors are supported by Calibrator Tool. Monitor profile names typically have the extension `.mon`. Profiles exist for other device types as well, such as scanners (profile names with the extension `.inp`) and printers (profile names with the extension `.out`). The Calibrator Tool window in Figure 2–1 also lists another category called Other for additional device types.

Figure 2–2 shows the Calibrator Tool popup window that lists the names of the loadable modules. The SDK includes the source code for sample modules shown in the figure. As previously mentioned, the `colorsense()` module listed in the figure uses a colorimeter (hardware puck connected to an RS-232C serial port on the monitor) that the end user would need to purchase to obtain measurement values. `XSolarisVisualGamma()` doesn't require this hardware. In either case, the measurement values obtained are used to build a table of color correction data for the particular visual.

The source code for the sample modules is provided in the `/opt/SUNWsdk/kcms` directory. You can use either of these modules as a template to create your own customized version of a loadable module.

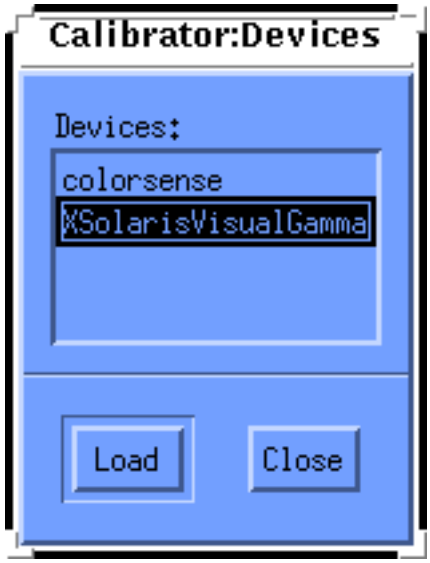

Figure 2–2 Selecting a Loadable Module

When the end user clicks on Load, Calibrator Tool uses the `OWconfig` database file to extract the names of the available loadable modules. You must edit `OWconfig` to add an entry for each loadable module you want to use with Calibrator Tool. `OWconfig` is the database from which a module is located and later dynamically loaded. For details on `OWconfig` and the format of calibration module entries, see Chapter 3.

Figure 2–3 shows the GUI that `XSolarisVisualGamma( )` (or `colorsense( )`) displays when the end user selects the module from Calibrator Tool's Calibrator:Devices window.

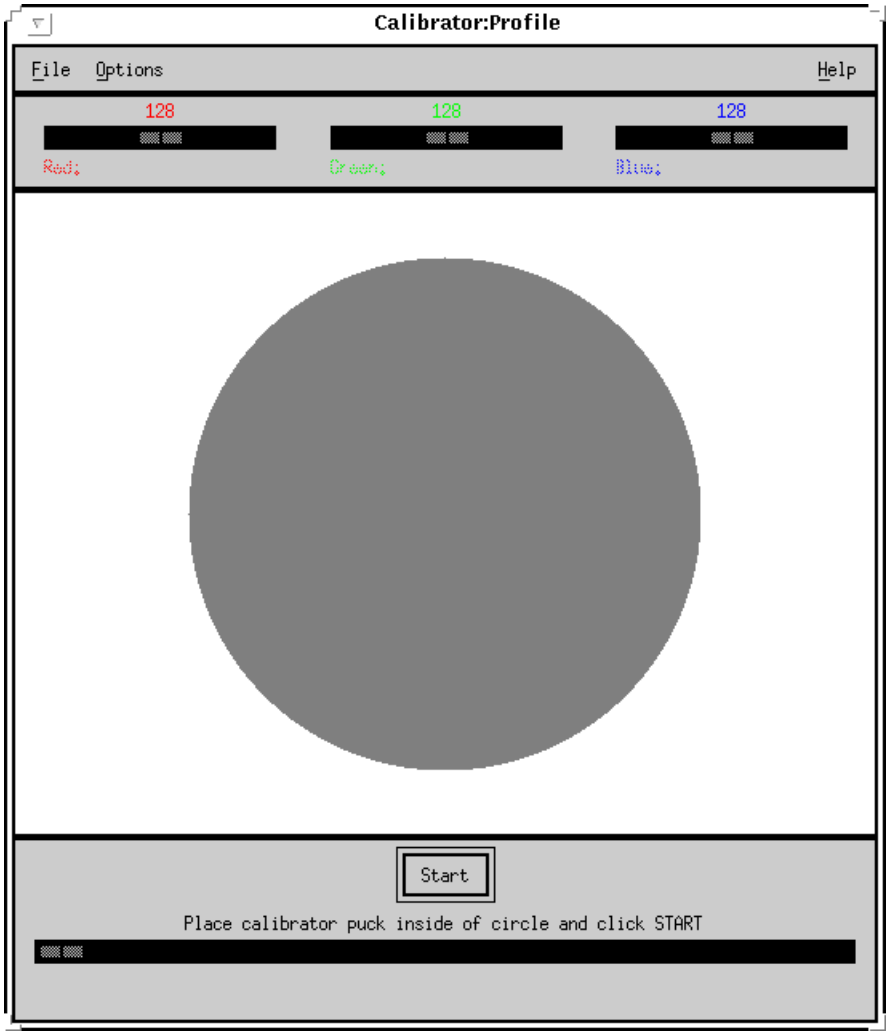

Figure 2–3 Interface to a Loadable Module

The significant point here is that, if your module makes use of a colorimeter, you need to consider a user interface design. The sample modules implement a color patch to obtain data from the monitor and, at the same time, to provide feedback to the user that calibration is actually taking place. Visibly each module provides feedback on its data gathering activity by displaying the color patch, which changes from cyan to magenta to yellow as the luminance values of each color are being read.

Updating Profiles

Calibrator Tool uses the data it collects from the loadable module to update the X Window System profiles.

Error and Status Messages

See Chapter 10, “Customizing Your Work Environment,” in the *OpenWindows Advanced User’s Guide*, for error and status messages that Calibrator Tool displays. For example, a message would be displayed if the loadable module did not finish collecting data. This situation would occur if, for example, the hardware failed to complete measurement or the end user cancelled the measurement. In such a case, the loadable module returns a nonzero status to Calibrator Tool and the entire measurement process must be restarted.

Note - It is up to the writer of the loadable module to provide the end user with any special status or error messages.

CHAPTER 3

OWconfig Database

In This Chapter

To provide for maximum flexibility in adding software modules, your module is dynamically loaded as a shared object at run time. The list of loadable objects (modules) is maintained in a system configuration database file called `OWconfig`.

By default, Calibrator Tool reads `OWconfig` from the `/usr/openwin/server/etc` directory. If it does not find the `OWconfig` file or the modules in it that it is looking for, it reads the `OWconfig` file from `/etc/openwin/server/etc`.

This chapter explains how to create, insert, and remove `OWconfig` file entries for calibration modules. The chapter also describes how the server uses the information in `OWconfig` entries to locate a dynamically loadable module.

Note - `OWconfig` is a database for various dynamically loadable objects. It not only can contain entries for calibration but also can include entries for other extensions and X Window System modules. The guidelines in this chapter pertain specifically to modules that interact with Calibrator Tool.

OWconfig Entry

Code Example 3–1 shows an `OWconfig` text file entry for a fictitious dynamically loadable calibration module.

CODE EXAMPLE 3–1 `OWconfig` File Entry

```
package="SUNWkcsmy"
class="KCMS_CALIBRATE" name="mydriver"
     kcmsCalDeviceType="monitor"
    kcmsCalLoadableModule="kcmscSUNWmydriver.so.1";
```

Table 3–1 describes the meaning of each attribute=value pair in the `OWconfig` file entry.

TABLE 3–1 Calibration Module Attributes and Meanings

Attribute=	Meaning
package=	This value is a unique name for your `OWconfig` entries.
class=	This value string is always KCMS_CALIBRATE, which uniquely identifies the class of loadable modules that interact with Calibrator Tool.
name=	This value is the name of the loadable module. It uniquely identifies the instance of the class object in the `OWconfig` file and must be different for each loadable module. This name string will appear in the Calibrator Tool Calibrator:Devices window. (See Figure 2–2 in Chapter 2.)
kcmsCalDeviceType=	This value is accessed when the end user chooses a particular device type to calibrate. It identifies the type of the device and is used to create a list of loadable modules for this device. The names of all the modules of the specified device type are displayed in the Calibrator:Devices window. (See Figure 2–1 in Chapter 2.) There can be several device type entries. Each is uniquely identified by the name= value.
kcmsCalLoadableModule=	This value is the name of the shared object that will be loaded when the end user clicks on Load. It has all the information necessary to start its own application to measure the device data. The module is expected to be installed in the `/usr/openwin/etc/devhandlers/` directory.

Updating the OWconfig File

Before your module can be loaded, you need to update the OWconfig file by inserting an entry for your module. To do this, you use the interactive OWconfig_calibrate program provided with the SDK.

To update the OWconfig file, you must be root. If you are not root or the /etc/openwin/server/etc path does not exist, the following error is generated:

```
OWconfile file not created/updated.
Check that you are root and /etc/openwin/server/etc exists.
```

Start the OWconfig_calibrate program as follows:

```
example% su
example# ./OWconfig_calibrate
```

Inserting An Entry

The following is an example of how to insert a calibrator module configuration entry into the OWconfig file. Sample user responses are enclosed in brackets ([]).

CODE EXAMPLE 3–2 Inserting A Calibrator Module Entry

```
ATTENTION: You must be root to update the OWconfig file.

Are you inserting an OWconfig entry? y/n

[y]

You will be asked to supply a name, a device type, and
a kcsLoadableModule to create an entry such as the
following:  name = 'mydriver'
        kcmsCalDeviceType = 'monitor'
        kcmscCalLoadableModule = 'kcmscSUNWmydriver.so.1'
Please see the KCMS Calibrator Tool LOadable Interface Guide for
information


The class for this entry will always be KCMS_CALIBRATE
Enter the unique name of the device

[mydriver]

Enter the device type  - normally monitor

[monitor]

Enter the name of your dynamically loadable module.

[kcmsSUNWmydriver.so.1]

This is your OWconfig entry.  OK? y/n
```

```
class = KCMS_CALIBRATE  name = mydriver
kmcsCalDeviceType = monitor
kmcsCalLoadableModule = kcmsSUNWmydriver.so.1
y

Do you have more entries to create? y/n
n
```

The `OWconfig_calibrate` program above appends the entry to the `OWconfig` file.

Try inserting the entry shown in the above example using the procedure outlined below:

1. **Run the `OWconfig_calibrate` program. Be sure you are root.**

   See "Updating the `OWconfig` File" on page 31.

2. **Insert the user responses shown in Code Example 3–2.**

3. **Check for the entry at the end of the `OWconfig` file.**

   The new configuration entry is appended to the `/usr/openwin/server/etc/OWconfig` file. For local machine use only, the `/etc/openwin/server/etc/OWconfig` file is updated. If you inserted the entry with the user responses in Code Example 3–2, the entry appears in `OWconfig` as shown below.

CODE EXAMPLE 3–3 `OWconfig` Entry For Calibrator Loadable Module

```
class="KCMS_CALIBRATE" name="mydriver"     kcmsCalDeviceType="monitor"
   kcmsCalLoadableModule="kcmsSUNWmydriver.so.1";
```

Removing Entries

The following is an example of how to remove a configuration entry from the `OWconfig` file. Sample user responses are enclosed in brackets ([]).

CODE EXAMPLE 3–4 Removing A Calibrator Module Entry

```
ATTENTION: You must be root to update the OWconfig file.

Are you inserting an OWconfig entry? y/n
[n]
To remove an OWconfig entry:

Enter the unique name for your driver for removal
from the OWconfig file.

[mydriver]

This is the OWconfig entry to remove.  OK? y/n

class = KCMS_CALIBRATE  name = mydriver
```

```
[y]

Do you have

more entries to remove? y/n
[n]
```

The `OWconfig_calibrate` program removes the last entry in the `OWconfig` file.

Try removing the entry you inserted following Code Example 3–4. Use the procedure outlined below:

1. **Run the `OWconfig_calibrate` program again. Be sure you are root.**
   See “Updating the `OWconfig` File” on page 31.

2. **Fill in the user responses shown in Code Example 3–4.**

3. **Check the `OWconfig` file.**
   The entry should no longer appear at the end of the file.

Locating and Opening the Module

Calibrator Tool supplies the `OWconfig` values as arguments to the `OWconfig` library routines. Using the values, the `OWconfig` routines find and dynamically open the loadable module. Then, using `dlsym`(3X), they execute the module functions.

CHAPTER 4

Measuring Monitor Response

In This Chapter

This chapter touches on the rationale Calibrator Tool uses to set up the structures for measuring data. In addition the chapter identifies precautions you can take in the design of your loadable module to be sure that it provides Calibrator Tool with accurate and meaningful measurement data.

Determining Displayable Visuals

When measuring monitor response, Calibrator Tool has to determine what X Window System visuals the frame buffer is capable of displaying. This is important because some frame buffers have their own gamma correction tables while others output directly to the monitor. Typically, however, frame buffers do not provide gamma correction, and gamma values are approximately 2.22 for SPARC systems. To identify the display capabilities of different visuals, Calibrator Tool calls the standard X Window System library routine `XGetVisualInfo(3)( )` and the Solaris library routine `XSolarisGetVisualGamma (3)( )`. This combination of routines returns a list of all the visual structures and their gamma values. (For details on `XGetVisualInfo( )`, see the *Xlib Reference Manual* listed in “Related Books” on page 9 in the Preface to this guide. For details on `XSolarisGetVisualGamma (3)( )`, see the man page description.)

Visuals Not Measured

Currently Calibrator Tool does not measure GrayScale or StaticGray visuals, because they are not color visuals.

Furthermore, it does not measure StaticColor visuals (24-bit TrueColor being the exception). This is because the loadable module has to display an *exact* color on the screen when it is measuring a color. To accomplish this, the loadable module must be able to access the writable color cell to modify the color at will. The loadable module has no direct control over the RGB values for StaticColor visuals, so it does not measure them.

Slow and Fast Mode Measurements

Calibrator Tool provides two modes of measurement: slow and fast. In fast mode, the Calibrator Tool main program requests the loadable module to return the minimum number of measurements to increase the speed of acquiring the measurement data. In slow mode, it requests that the loadable module return measurements for all the visuals whose color it has writable control over. These distinct modes provide support for frame buffers that might have different color responses at different depths.

In slow mode, visuals are considered to be *equivalent* if they have the same gamma and depth. In this case, measurements are required for one representative of a class of equivalent visuals. If, for example, 24-bit DirectColor and 24-bit TrueColor have the same visual gamma, they are considered to be equivalent, and measurement is performed for TrueColor only. Since measurement is a response of the monitor to a particular depth and gamma, the measurement is not repeated for DirectColor. Instead, the DirectColor visual profile is updated with the TrueColor measurement results.

In fast mode, visuals are considered to be equivalent if they have the same gamma, regardless of depth. This mode is provided for monitors whose response is believed to be the same for varying numbers of planes. If, for example, 8-bit PseudoColor and 24-bit TrueColor have the same gamma, the PseudoColor measurement is the only one performed, with the result being used to update the TrueColor as well as the PseudoColor profiles. In this case, it doesn’t matter which visual generates the color patches as long as the visuals represent the same color.

Precautions While Taking Measurements

The sample modules display a color patch that changes between cyan, magenta, and yellow as the end user takes measurements. When taking measurements from the monitor display using a hardware puck, it is important to ensure that the window displaying the patch is not obscured inadvertently by some other window tool. The sample modules prevent interference by using Motif's override shells, which are always on top of every other window. If by chance another window is moved into the color patch display region, that window passes under the override shell window and does not disturb the color patch display.

Recommended Number of Measurements

Although there is no specified number of measurements for visuals, it is recommended that the number be large enough for the KCMS API library to perform a reasonable interpolation of data for each color channel. The `XSolarisVisualGamma( )` (or `colorsense( )`) module determines the number of intensity levels based on the bits_per_rgb value of the visual (that is, 2 bits_per_rgb). Then it measures every fourth intensity level to speed up measurement. If, for example, it is measuring a 6-bit frame buffer with 64 intensity levels, it performs 17 measurements. For an 8-bit frame buffer, it performs 65 measurements (including intensity 0). For intensity levels that are less than 8 bits_per_rgb, you must skip fewer intensity levels to achieve a more accurate resulting response curve.

Allocating and Deallocating Structures For Measurement Data

Calibrator Tool handles allocation and deallocation of `KCMSCVisuals` and `KCMSCData` structures. Once the loadable module completes the shared data structure `KCMSCData` (by filling in the output portion of the `KcsCalibrationData` structure for each of the measured X visuals) it returns, leaving the data in the shared structure.

Handling the Measurement Data

Calibrator Tool uses the data provided by the loadable module to update the X Window System profiles using calls from the KCMS API library.

CHAPTER 5

Data Structures

This chapter describes the data structures used by Calibrator Tool.

KCMSCData

```
typedef struct _KCMSCData {
   int                  size;  /* number of data sets */
  KcsCalibrationData ** data;  /* pointers to data sets */ }
  KCMSCData;
```

KCMSCData is a structure that collects data. For monitor calibration, the size element must be equivalent in value to numVisuals in the KCMSCVisuals structure. (For more information on numVisuals, see "KCMSCVisuals" on page 40 in this chapter.)

The members in this structure are described below:

size	Is the number of sets of data. For monitors, the value must be equivalent to the value of numVisuals.
KcsCalibrationData	Is a variable array of pointers to KcsCalibrationData structures. KcsCalibrationData contains an array of input values for the profiles (supplied by Calibrator Tool) and an array of output values (results of measurements) that represent the color response of the device. (For details on the KcsCalibrationData type, see Chapter 3, "Data Structures" in the *KCMS Application Developer's Guide*.)

KCMSCVisuals

```
typedef struct _KCMSCVisuals {
    Display * display;
    int       screen;
    int       numVisuals;   /* dynamic array size */
    XVisualInfo ** visuals;  /* visuals to get data for */


} KCMSCVisuals;
```

`KCMSCVisuals` is a structure that identifies to the loadable module the display and visuals upon which to perform measurements. The members in this structure are described below:

`display`	Is a `Display` pointer. The `Display` pointer is for use when the loadable module makes an X library call (such as `XCreateWindow( )`) to connect to the local windowing system. (For details on `Display`, see the *XLib Programming Manual*.)
`screen`	Is the screen number to be used in Xll calls. Calibrator Tool fills in the value of this field.
`numVisuals`	Is the number of X visuals for which color measurement data is needed. Calibrator Tool fills in the value of this field based on the number of X visuals to be updated. (For details, see “Slow and Fast Mode Measurements” on page 36 in Chapter 4.)
`visuals`	Is an array of `XVisualInfo` pointers. (See “`XVisualInfo`” on page 40 below for the format of this structure.)

XVisualInfo

The `XVisualInfo` structure has the format shown below. For details on the members of this structure, see the *XLib Programming Manual.*

```
typedef struct {
  Visual* visual;
  VisualID visualid;
  int screen_num;
  unsigned int depth;
  int class;
  unsigned long red_mask;
  unsigned long green_mask;
  unsigned long blue_mask;
  int colormap_size; /* Same as map_entries member of Visual */
  int bits_per_rgb;
 } XVisualInfo;
```

CHAPTER 6

Functions

This chapter alphabetically presents each calibration function your loadable module calls to interface with KCMS Calibrator Tool. For each function, the chapter provides its purpose, arguments, and return values.

Once a module is dynamically opened, Calibrator Tool uses `dlsym`(3X) to access the module functions by their symbolic names. Then it adds the names to the process symbol space.

KCMSCMonClose( )

```
int
 KCMSCMonClose (void);
```

Purpose

`KCMSCMonClose( )`performs any cleaning necessary (for example, unlocking the serial port) for the module code.

Arguments

None.

Return Value

Returns 0 if successful; returns any other nonzero value if unsuccessful.

KCMSCMonInit( )

```
int
 KCMSCMonInit (KCMSCVisuals *vis_data);
```

Purpose

`KCMSCMonInit( )` accepts a pointer to a `KCMSCVisuals` structure passed to it from the Calibrator Tool main program and is used by the module for initialization. Examples of initialization tasks the module might perform include:

- Initializing the RS-232 serial port, if a hardware colorimeter is used to take color measurements
- Starting its own GUI-based application
- Setting everything necessary to measure monitor data
- Doing nothing, if the measurements were previously stored in a file by the end user

Arguments

vis_data	Points to a `KCMSCVisuals` structure. For details on the format of this structure, see "`KCMSCVisuals`" on page 40 in Chapter 5.

Return Value

Returns 0 if successful; returns any other nonzero value if the hardware fails to initialize.

KCMSCMonMeasure( )

```
int
 KCMSCMonMeasure (KCMSCData *measured_data);
```

Purpose

`KCMSCMonMeasure( )` accepts a pointer to a `KCMSCData` structure passed to it from the Calibrator Tool main program and performs the following functions:

- It performs the number of measurement sets specified by the value of the `size` field in the `KCMSCData` structure (alternately, it can be programmed to read all the measurements previously created).
- It uses the measurement values to fill in the appropriate fields in the `KCMSCData` structure.

Arguments

measured_data	Points to a `KCMSCData` structure. For details on the format of this structure, see“`KCMSCData`” on page 39 in Chapter 5.

Return Value

Returns 0 if successful; returns any other nonzero value if the collection of data was interrupted either by the user or by failure of the hardware to measure the data.

Glossary

channel Refers to a red, green, or blue intensity. See *RGB values*.

color lookup table A color map that establishes the relationship between pixel values and visible screen colors. A pixel value is an index into this table to arrive at an RGB value defining a displayable color.

colorimeter A hardware device used to measure luminance values for calibrating a monitor.

depth The number of planes (the number of bits per pixel value) for a window.

display A set of one or more screens driven by an X Window System server. The `Display` structure contains all the information about a particular display and its screens. It is filled when a program calls an X library routine to connect to a windowing system. Also see *screen*.

gamma correction Changing RGB values to ensure that the appropriate color appears on the screen.

KCMS framework The KCMS API library routines available with the Kodak Color Management System portion of the Solaris 2.5 SDK that are used to manipulate color profiles.

`OWconfig` A system configuration database file that maintains a list of the dynamically loadable objects (modules).

profile A file that provides information to the KCMS framework on how to convert input color data to the appropriate color-corrected output color data. See *KCMS framework*.

RGB values Red, green, and blue intensity values used to define a color.

screen

One or more screens can be associated with a single monitor, keyboard, and pointing device. A `Screen` structure is a member of a `Display` structure. See *display*.

visual

A pixel-value translation into colors that are displayed on the screen. X Window System visuals include PseudoColor, DirectColor, GrayScale, StaticColor, TrueColor, and StaticGray.

Index

N

O

P

R

S

T

U

V

X