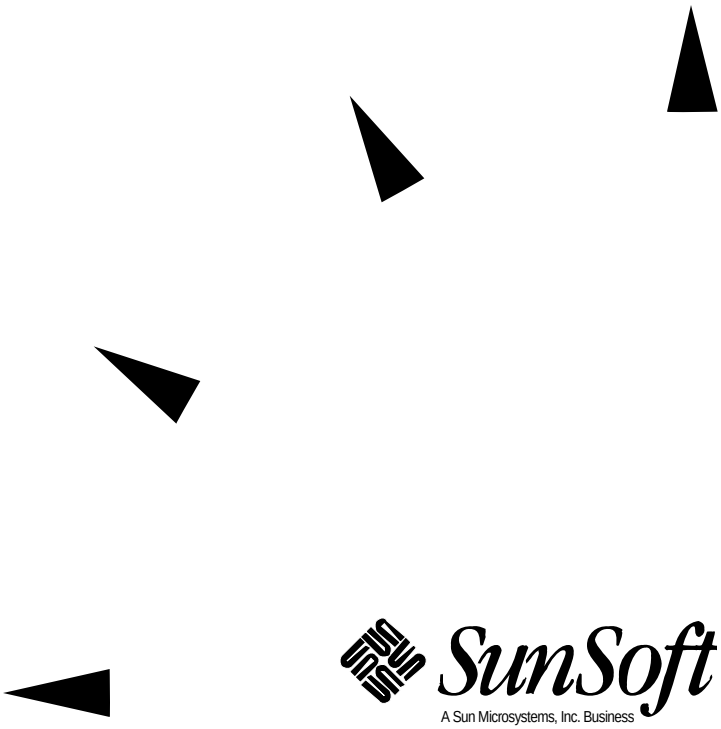
Solaris 2.4
Driver Developer Kit
Introduction



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

Contents

Preface

The *Driver Developer Kit Introduction* gives an overview to the Solaris™ 2.4 Driver Developer Kit (DDK). It also:

- Tells you how the DDK fits into a Solaris development environment.
- Lists new DDK features.
- Describes each component of the DDK.
- Lists and gives a brief description of DDK documentation.
- Tells you how to obtain hard copy documents, technical support, and training for the DDK.

Who Should Use This Book

If you are a driver developer who is interested in providing driver software for Solaris you should read this book. Typical driver developers are independent hardware vendors (IHVs) or original equipment manufacturers (OEMs) who want their hardware products to operate in a Solaris environment.

DDK users include:

- IHVs and OEMs who are interested in writing DDI/DKI-compliant device drivers for hardware devices.
- IHVs whose products include device drivers.
- IHVs interested in writing device handlers for the OpenWindows™ Server.
- IHVs who are writing device pipelines for the XGL™ graphics library.

- IHVs who are writing device handlers to port hardware devices to the XIL™ imaging library and technology providers who are writing additional device-independent acceleration code for XIL operators.
- IHVs writing FCode PROM programs for SBus cards.

This manual assumes that you are familiar with the Solaris 2.4 distributed computing environment and general UNIX® device driver principles. If you are new to writing device drivers, see the first three chapters of the *Writing Device Drivers* manual.

Before You Read this Book

Before you install the DDK, you must have one of the following versions of Solaris 2.4 installed on your x86 or SPARC platform:

- x86 Enterprise Server
- x86 Workgroup Server
- x86 DeskTop
- SPARC DeskTop
- SPARC Workgroup Server
- SPARC Enterprise Server

You should also have the AnswerBook® product (the on-line version of the documentation) installed. The AnswerBook product is provided with any version of Solaris 2.4. Once you have a version of Solaris 2.4 installed, read the manuals that are provided in the DDK box. These include:

- *Driver Developer Kit Introduction* (this manual)
- *Driver Developer Kit Open Issues and Late-Breaking News*
- *Driver Developer Kit Installation Guide*

You can then install the DDK using the *Driver Developer Kit Installation Guide.*

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book.

Table P-1 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. system% You have mail.
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% 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.
Code samples are included in boxes and may display the following:		
%	UNIX C shell prompt	`system%`
$	UNIX Bourne and Korn shell prompt	`system$`
#	Superuser prompt, all shells	`system#`

Introduction 1

Welcome to the *Solaris* 2.4 *Driver Developer Kit* (DDK). This chapter introduces you to the DDK and tells you how it fits into a Solaris development environment. It also lists features that are new to the DDK in the Solaris 2.4 release.

Driver Developer Kit Overview

The DDK helps you develop dynamically loadable *device drivers* and *graphics device handlers* for Solaris 2.4 by providing you with the necessary software tools, technical assistance, on-line documentation, and technical training information.

Device drivers present the kernel with a consistent interface to diverse devices. Solaris supports a set of source-level interfaces between drivers and the kernel called the *device driver interface/driver-kernel interface* (DDI / DKI). Device drivers are dynamically loaded by the SunOS™ kernel. Device driver code runs as kernel-level code.

Graphics device handlers (or *device handlers*) are software modules that add device-specific support for a *VISUAL™ for Solaris* graphics foundation library. Each *VISUAL for Solaris* foundation library defines a device porting interface, called a *graphics porting interface* (GPI). With the help of the DDK, you can write a device handler for a specific foundation library that is compliant with the GPI for that foundation library and is dynamically loaded by that foundation library. Device handler code runs as user-level code.

1

For graphics devices, you generally need to write both a device driver and a graphics device handler for one or more VISUAL libraries.

Also included in the DDK are FCode development tools you need to help you write OpenBoot PROM code for SBus cards.

DDK *components* are the software tools, libraries, server, and on-line documentation the make up the DDK. Except for the OpenWindows X Server, which is delivered on the Solaris CD-ROM disc, the following DDK components are provided on the DDK CD-ROM disc:

- Sample drivers and driver development tools
- Device driver handler support for *VISUAL for Solaris*, which includes:
  - OpenWindows X Server
  - XGL graphics library
  - XIL imaging library
- FCode development tools
- On-line documentation

These DDK components are explained further in Chapter 2, “DDK Components.”

How the DDK Fits Into a Solaris Development Environment

Solaris developers produce applications, drivers, and graphics handlers that are ready for the end-user Solaris runtime environments. A Solaris development environment may be constructed using the Solaris runtime environments (available with any version of Solaris 2.4), developer kits (the Solaris 2.4 Driver Developer Kit and Software Developer Kit), and compilers (the ProCompilers and SPARCompilers C and C++).

The DDK contributes to this environment by providing the background information, requirements, and testing tools that you need to create software support for specific hardware devices in the Solaris runtime environments. The DDK provides the information you need to create a wide array of hardware drivers. In some cases, the DDK reduces direct coding efforts by providing sample driver code as starting points for driver development.

For more information on the Solaris 2.4 release, see the *Solaris 2.4 Introduction* manual.

New DDK Features

The following features are new to the Solaris 2.4 DDK:

- The DDK software and documentation are now merged so that there is support for both the x86 and SPARC architectures.

New features for each component (for example, new *Sample Drivers and Driver Development Tools* features) are described in Chapter 2, “DDK Components.”

1

DDK Components 2

The DDK is made up of individual software technologies and supporting documentation. The software technologies and the supporting documentation are called the *components* of the DDK. This chapter describes each DDK component and tells you about the documentation associated with that component. If a component has features that are new in the Solaris 2.4 release, they are listed.

Sample Drivers and Driver Development Tools

The *Sample Drivers and Driver Development Tools* component of the DDK provides materials to assist you in developing device drivers for the Solaris release. These materials include:

- Sample device drivers
- Driver development tools
- Documentation

These materials cover:

- How to use Solaris DDI/DKI interfaces to ensure forward compatibility with future Solaris releases.
- How to implement drivers for different types of devices.

2

New Features

The following new features are in this release of the DDK *Sample Drivers and Driver Development Tools* component:

- Updated and merged *Writing Device Drivers* manual—This manual describes writing device drivers for both x86 and SPARC architectures.
- DDI compliance tool (DDICT)—This tool checks source code for noncompliance with the DDI/DKI.
- Frequently asked questions (FAQ) file—This file is a list of frequently asked questions about device drivers and DDI/DKI topics and answers to those questions.

Sample Device Drivers

The DDK includes a variety of sample device drivers. In some cases, the device drivers provided with the DDK are complete functioning modules; in other cases, hardware details are omitted or generalized to provide a template from which you can generate a functioning driver.

You can use these sample device drivers, together with hardware-specific documentation and the *Writing Device Drivers* manual, as a starting point for developing DDI/DKI-compliant drivers.

The DDK provides sample device drivers for the following:

- Simple SCSI target driver
- Block SCSI target driver
- Graphics (frame buffer) device driver
- Data link provider interface (DLPI) network device driver
- Simple programmed I/O driver
- Simple DMA character device driver
- Simple RAM disc driver

Driver Development Tools

The DDI compliance tool (DDICT) checks device driver C source code for non-DDI/DKI compliance. Non-DDI/DKI compliance is the use of features that are not part of the Solaris 2.x DDI/DKI. DDICT issues error and warning messages when it finds noncompliant code.

Documentation

The following driver development documentation is available in the DDK. See "A Complete List of DDK Documentation" on page 15 for the locations of these documents.

- *man Pages(9): DDI and DKI Overview*—Section 9 of the *Solaris 2.4 Reference Manual AnswerBook*. These manual pages document DDI/DKI source-level interfaces. Subsections cover:
  - 9E—required driver entry points
  - 9F—kernel functions drivers may call
  - 9S—kernel structures used by drivers
- *Writing Device Drivers*—This manual describes device driver development for character-oriented devices, block-oriented devices, and SCSI target devices. It also covers general device driver topics.
- FAQ—The FAQ is an ASCII file containing frequently asked questions and the answers to those questions for DDI/DKI and device driver development topics.
- `ddict.1`—This is the `man` page for DDICT.
- *Data Link Provider Interface Specification*—This document specifies a STREAMS kernel-level instantiation of the ISO Data Link Service Definition (DIS 8886) and Logical Link Control (DIS 8802/2).
- *SCSI Host Bus Adapter Drivers*—This PostScript document describes how to write a SCSI HBA nexus driver using the new DDI/DKI SCSI interfaces. This document is a work in progress and will become a chapter in the *Writing Device Drivers* manual, called "SCSI HBA Target Drivers." Although not yet complete, this chapter contains valuable information on creating a SCSI HBA driver and provides sample code illustrating the use of the HBA driver interfaces.

VISUAL for Solaris

The *VISUAL for Solaris* environment is the windows and graphics hardware developer environment. It includes support for the:

- XGL graphics library
- XIL imaging library
- X11R5-based window-system server (OpenWindows X server)

2

The *VISUAL for Solaris* environment includes application-programming interfaces (APIs) for a wide variety of graphics functionality, including:

- 2-D and 3-D geometric graphics (from the XGL graphics library).
- Imaging and digital video (from the XIL imaging library).
- Stencil-paint style graphics (from Display PostScript™ library).
- Basic pixel graphics (from the X11 library).

You may build applications by using a set of application libraries, including libraries from SunSoft™ and from third parties. These application libraries are built on a set of foundation libraries that are part of the Solaris development environment—one for each major area of graphics functionality. Some of these foundation libraries are also available directly to application developers in the *Solaris* 2.4 *Software Developer Kit*. Each foundation library defines a graphics porting interface that is the interface for porting the library to hardware devices.

You may port your device to the *VISUAL for Solaris* environment by porting one or more of the Solaris *graphics porting interfaces* (GPIs) to a device. The DDK provides information enabling you to do this. A device might be a:

- Frame buffer
- Graphics accelerator
- Input device
- Frame grabber
- Image compression device

A device might also be a software component, for example, an optimized version of a compression algorithm or rendering pipeline. You can use the Solaris GPIs to support such “software devices.”

For further information on the *VISUAL for Solaris* environment see the following documents:

- *Solaris VISUAL Overview for Driver Developers*—This manual gives the overview and philosophy of the *VISUAL for Solaris* environment. It includes detailed discussions of:
  - XGL graphics library (geometry)
  - XIL imaging library (imaging and video)
  - Display PostScript (stencil-paint)
  - X Window System (pixel-based)
- *Solaris VISUAL White Paper*—This paper gives the overview and philosophy of the *VISUAL for Solaris* environment.

Solaris XGL 3.1 Graphics Library

The XGL graphics library is a foundation graphics library that provides geometry graphics support for Solaris-based applications. The XGL library includes a device-level interface that defines the mapping of XGL code to the underlying hardware. If you write XGL loadable device pipelines (device handlers), which provide this mapping, you can build graphics devices that support any binary XGL application.

Because the Solaris environment provides mechanisms to dynamically load kernel device drivers and user process shared libraries, you can incorporate a new graphics accelerator into the Solaris environment by providing a dynamically loadable kernel device driver and an XGL device pipeline.

The XGL architecture provides open, well-defined interfaces that facilitate the task of implementing loadable device pipelines. The geometry-rendering interfaces are organized into a set of porting layers, which enable you to map underlying hardware capabilities to the XGL application-programming API. You can choose a layer for the device pipeline based on the functionality of the device and let XGL handle the rendering of functionality not accelerated by the device. For further information on the XGL graphics library, see the following documents:

- *XGL Device Pipeline Porting Guide*—This guide describes how to write an XGL graphics handler. It provides information on XGL internal interfaces and utilities, and on the mechanisms that enable the device code to work with the XGL device-independent code.
- *XGL Architecture Guide*—This guide provides information on the XGL architecture and presents details on the implementation of key aspects of that architecture. It also provides information on the design of the XGL loadable pipelines and describes XGL object-oriented internal design and coding conventions.
- *XGL Test Suite User's Guide*—This guide describes the installation and use of a set of graphics verification programs used to test the accuracy of a particular XGL implementation.
- *The XGL White Paper*—This white paper describes the purpose, structure, and features of functions in the XGL graphics library.

2

Solaris XIL 1.2 Imaging Library

The Solaris XIL imaging library is a foundation library for image processing and digital video applications. The XIL imaging library provides a single interface to the hardware with which it interacts. The XIL library has two public interfaces:

- ISV interface—documented in the Solaris 2.4 Software Developer Kit (SDK).
- Technology provider interface—provided with the Solaris 2.4 DDK. It enables you to:
  - Port new hardware devices to the XIL imaging library.
  - Accelerate existing XIL functions.
  - Add video compressors and decompressors to the XIL imaging library.

For further information on the XIL imaging library see the following documents:

- *XIL Device Porting and Extensibility Guide*—This guide describes the architecture and internal interfaces of the XIL library. It also describes the XIL library C++ classes and the mechanism for acceleration and porting of new hardware. If you are porting hardware to use the XIL imaging library or if you are writing device-independent acceleration code for XIL operations, you should read this guide.
- *XIL Test Suite User's Guide*—This guide describes how to run the Xilch test suite to verify the XIL imaging library. It also describes how to create new Xilch tests and benchmarking.
- *The XIL White Paper*—This white paper describes the purpose, structure, and features of functions in the XIL imaging library.

OpenWindows X Server 3.4

The OpenWindows X Server is based on the MIT X Consortium X11R5 server. The OpenWindows X Server provides basic window system and pixel graphics support. It also provides stencil-paint style graphics through the Display PostScript (DPS) extension.

The OpenWindows X Server provides a device-level GPI based on the standard Device Dependent X (DDX) interface and the XInput Extension for input devices. You can incorporate support for new graphics accelerators or input devices by providing dynamically loadable device handlers that implement the GPI for the OpenWindows X Server.

While the OpenWindows X Server defines a basic GPI based on DDX, it provides several utility porting layers that help you implement the GPI on various types of graphics accelerators and input devices. Some of these utility porting layers are common to the X11R5 server, such as:

- Color frame buffer (cfb)
- Monochrome frame buffer (mfb)
- Machine independent (mi)

Other layers, such as direct graphics access (DGA) and multiple plane group (MPG), enable you to use SunSoft features that provide enhanced graphics performance or support for advanced frame buffer architectures. The porting interface enables phased porting. This means that, in the beginning of porting, you can provide a basic port with limited acceleration, and later you can optimize the device port to use the full graphics accelerator functionality.

The OpenWindows X Server includes:

- Sample device handler source code for many devices.
- Sample source code for some of the utility layers to aid debugging.
- Server header files required to compile device handlers.
- Sample directory hierarchy, including `Imakefiles`, to help build device handlers.

New Features

The following features are new for this release of the DDK:

- Support for x86 and SPARC architectures
- Support transparent overlays
- Enhanced DGA drawable interface
- `OWconfig` access method
- Debug server to aid in debugging your DDX handlers
- Support for visual gamma corrections

For more information on the OpenWindows X Server, including detailed information on these new features, see the *OpenWindows Server Device Developer's Guide.* This guide provides detailed information for writing device handlers for the OpenWindows X server.

FCode Development Tools

FCode is a Forth-like language used to write to OpenBoot PROM code for SBus cards. When used with the other standard programming tools provided with the Solaris release, the FCode *tokenizer, detokenizer,* and *fakeboot* are important FCode development tools.

The FCode tokenizer converts FCode source into FCode binary that is suitable to reside on PROM. Use the FCode tokenizer if you want to design new SBus interface cards for Sun SPARC systems. The detokenizer converts the FCode executable file into a source file. For testing, fakeboot encloses an executable in a file suitable for loading into memory with the boot program.

For more information on FCode development of OpenBoot PROM and SBus cards, see the following documentation:

- *Writing FCode Programs* manual—describes how to write, debug, and test FCode programs for SPARC-based systems and interface card devices.

For additional information on FCode development of OpenBoot PROM and SBus cards, you may want to see the following documentation although it is not provided with the DDK:

- *OpenBoot Command Reference Manual*—This manual is available in the *Solaris 2.4 System Administrator AnswerBook* on-line documentation.
- *OpenBoot Quick Reference Card*—This card provides a quick reference for OpenBoot commands. It is available in the *Solaris 2.4 System Administrator AnswerBook* on-line documentation.
- *IEEE Standard for a Chip and Module Interconnect Bus: SBus* (IEEE Standard 1496-1993)—Please write IEEE to obtain a copy of this specification at:

  Institute of Electrical and Electronic Engineers, Inc.
  345 East 47th Street
  New York, NY 10017, USA

Technical Support, Training, and Documentation 3

This chapter tells you how to obtain technical assistance when you are installing or using the DDK. It also lists technical classes offered through Sun Educational Services and tells you how to obtain a class location list. In addition, a complete list of the DDK documentation is provided for easy reference.

Technical Support

If you need help with the installation or use of the DDK and you're calling from the United States or Canada, call 1-800-SOFTSPT (1-800-763-8778) for an Authorized Service Provider near you. Customers outside the United States or Canada, please call 1-510-460-3267. Also refer to your Support Addendum card.

The DDK is also supported through the *SunSoft Catalyst Developer's Program* (Catalyst[SM]). The Catalyst program offers a variety of technical support services to assist you in bringing your Solaris-ready software applications to market. For information about the Catalyst program, contact the Catalyst Information Center at 510-460-3267.

3

Sun Educational Services

In partnership with Sun Educational Services, SunSoft Authorized Education Centers provide the following training on Solaris Developer products. These course are offered at many locations in the United States and throughout the world. For a more current listing of classes and a class location list call Sun Educational Services at 1-800-422-8020.

Lecture and Lab Courses

- Fundamentals of Solaris 2.x
- Solaris 2.X for Advanced Users
- Solaris for Programmers
- Solaris 2.X Concepts and Tuning
- Solaris 2.X Internals
- Programming Transition to Solaris
- Shell Programming
- C Shell Programming
- Korn Shell Programming
- Multithreaded Interapplication Programming
- Network Programming
- C++ Programming
- Object-Oriented Analysis and Design Using C++
- Introduction to Writing Device Drivers Solaris 2.X
- `Grep`, `Sed`, and `Awk` Programming
- System Interface Programming
- ANSI C for Nonprogrammers
- ANSI C Programming
- Advanced ANSI C Programming
- SPARCworks Programming Environment

Self-Paced Courses

- Fundamentals of Solaris
- UNIX Software Tools for Programmers
- UNIX System V Release 4 Internals
- Shell Command Language for Programmers
- ANSI C Language for Programmers
- Using C++
- Advanced C++
- Object-Oriented Design

A Complete List of DDK Documentation

This section lists all of the DDK documentation and tells you where to find it. The DDK documentation is provided in the following ways:

- On-line through the AnswerBook product and in PostScript files.
- In hard copy in the DDK box.
- In hard copy through SunExpress.

Documents Available Through the AnswerBook Product

The AnswerBook product provides a way for you to view the DDK documentation on-line. The DDK includes several sets of on-line documentation. These sets of documentation are called:

- *Solaris 2.4 Driver Developer AnswerBook*
- *Solaris 2.4 Supplemental Developer AnswerBook*
- *Solaris 2.4 Reference Manual AnswerBook*

For information on installing the DDK on-line documentation, see the *Driver Developer Kit Installation Guide*.

Solaris 2.4 Driver Developer AnswerBook

- *Driver Developer Kit Introduction* (this manual)
- *OpenWindows Server Device Developer's Guide*
- *Writing Device Drivers*
- *Writing FCode Programs*
- *XGL Architecture Guide*
- *XGL Device Pipeline Porting Guide*
- *XGL Test Suite User's Guide*
- *XIL Device Porting and Extensibility Guide*
- *XIL Test Suite User's Guide*
- *Solaris VISUAL Overview for Driver Developers*

Solaris 2.4 Supplemental Developer AnswerBook

- *Application Packaging Developer's Guide*
- *STREAMS Programmer's Guide*

Solaris 2.4 Reference Manual AnswerBook

- *SunOS Reference Manual*
  - *man Pages(1): User Commands*
  - *man Pages(1M): System Administration Commands*
  - *man Pages(2): System Calls*
  - *man Pages(3): Library Routines*
  - *man Pages(4): File Formats*
  - *man Pages(5): Headers, Tables and Macros*
  - *man Pages(6): Demos*
  - *man Pages(7): Special Files*
  - *man Pages(9): DDI and DKI Overview*
  - *man Pages(9E): DDI and DKI Driver Entry Points*
  - *man Pages(9F): DDI and DKI Kernel Functions*
  - *man Pages(9S): DDI and DKI Data Structures*

Documents Available Through PostScript Files

These files are installed in `/opt/SUNWddk/doc`. They are also on the DDK CD-ROM disc in `/cdrom/ddk_2_4/PostScript_files`.

Title	**File Name**
Multithreading and Real-Time in Solaris: Terms and Concepts White Paper	`OSMT-WP.PS`
Data Link Provider Interface Specification	`DLPISPEC.PS`
Solaris Visual White Paper	`SVIS.PS`
Solaris VISUAL Overview for Driver Developers	`VISOVRVW.PS`
The XGL White Paper	`XGL-WP.PS`
The XIL White Paper	`XIL-WP.PS`
SCSI Host Bus Adapter Drivers	`scsihba.ps`

These files are on the CD-ROM disc in `/cdrom/ddk_2_4/PostScript_files`. You can also find these documents in the AnswerBook on-line documentation.

Title	File Name
Writing Device Drivers	`wdd.ps`
OpenWindows Server Device Developer's Guide	`OWSERVER.PS`
XGL Architecture Guide	`XGLARCH.PS`
XGL Test Suite User's Guide	`DENIZEN.PS`
XGL Device Pipeline Porting Guide	`XGLDEVPG.PS`
XIL Device Porting and Extensibility Guide	`XILSYSPG.PS`
XIL Test Suite User's Guide	`XILCH.PS`

Documents Available in Hard Copy Only

- *Driver Developer Kit Installation Guide*
- *Driver Developer Kit Open Issues and Late-Breaking News*

AnswerBook Documents Also Available in Hard Copy

To get hardcopy versions of the DDK AnswerBook on-line documentation contact SunExpress™ (1-800-873-7869) or an authorized Sun reseller.

Suggested Reading Beyond the DDK

- *OpenBoot Command Reference Manual*—This manual is available in the *Solaris 2.4 System Administrator AnswerBook* on-line documentation.
- *OpenBoot Quick Reference Card*—This manual is available in the *Solaris 2.4 System Administrator AnswerBook* on-line documentation.
- *IEEE Standard for a Chip and Module Interconnect Bus: SBus* (IEEE Standard 1496-1993)—Please write IEEE to obtain a copy of this specification at:

  Institute of Electrical and Electronic Engineers, Inc.
  345 East 47th Street
  New York, NY 10017, USA

3