

Book 1—Getting Started with Philips TriMedia



Table of Contents

About These Manuals

Conventions Used in These Manuals	vii
Fonts and Colors	vii
Types of Notes	viii
How to Get Help	viii

Chapter 1 Overview of the TriMedia System

TriMedia VLIW Architecture	10
Overview of The TriMedia VLIW Architecture	10
Functional Unit Assignment	10
What Makes TriMedia's VLIW Architecture Better	11
Performance	12
TriMedia Software Streaming Architecture	12
TriMedia Hardware Components	13
The TriMedia Processor	13
The TriMedia (TM-1xxx/TM-2xxx) IREF Board	14
TriMedia Software Components	15
TriMedia Tools	16
Execution Tools.....	17
System Tools.....	17
Performance Analysis and Enhancement Tools	18
Machine-Level Simulator	18
SDE Documentation	19

Chapter 2 Installing the TriMedia System

Introduction	22
Installing TriMedia Software and Hardware on the PC	23
Supported PC Build Hosts	23
Supported PC Execution Hosts	23

Installing the TriMedia SDE	23
Using the Setup Program	24
Entering Personal Information	24
Choosing the Location of the Installation Folder	24
Specifying the Installation Type	25
Finishing the Installation	27
A Note on Installation Requirements For the TriMedia Debugger	27
Installing the TriMedia Reference Board	28
JTAG-Hosted Operation	28
Installing JTAG.SYS on Windows NT	28
Installation Requirements for Hosted Operation	29
Installation Procedure	29
Installing the TCS on UNIX-Based Platforms	34
Installing TriMedia on Solaris™	34
Installing TriMedia on SunOS	35
Installing TriMedia on HP-UX	36

Chapter 3 Running Demonstration Programs

Introduction	38
Running the fplay Program	39
Running the Files Program	41
Running the icptest Program	42
Running the exalVrendVO Program	44

Chapter 4 Using TriMedia SDE Tools

Introduction	46
Sample Program: getIt.c	46
Compiling TriMedia Applications	47
Simulating TriMedia Applications	48
Running TriMedia Applications on the TriMedia Processor	49
Debugging TriMedia Applications	50
Where to Go from Here	54

Chapter 5 Migration Issues

Application Libraries	56
Audio Renderer (ArendA0)	56
Audio Digitizer (AdigAI)	56
Video In/Out	56
Multiple Hardware Units	57
Video Digitizer (VdigVI)	57
API Changes	57
Other Changes	57
Video Renderer (VrendVO)	58
Clock	58
icptest	58
Examples	59
Device Libraries and Initialization	59
Registry	59
Component Manager	59
Specifying Board Support Packages	59
Migration Notes	60
tmsim—The Simulator	60
Benchmarking	60
tmman—Execution Hosts	60
TSSA	62
In-Place Components	62
tmcc/tmccom—C/C++ Compiler	62
General Information	62
Long Double Floating Point	64
C++ Support	65
C++ Compiler	65
C++ Standard Library	65
Standard C Library	66
tmdbg—Debugger	66
Debugging Multiprocessor and Multitasking Applications	66
Downloading Support	66
Enhanced GUI Features	67
tmprof—Profiler	67
tmsched—Instruction Scheduler	68

Documentation	68
New Components Available as of SDE v2.0.....	68
New Code Compression Library	68
New Flash File System Manager	69
Memspace library	69
New Serial I/O Host	69
Migrating from SDE Version 2.0 to SDE Version 2.1	70
tmcc—the Compiler	70
C++ Support	71
tmdbg—Debugger	71
tmman—Execution Hosts	72
Device Libraries	72
Application Libraries	72

Chapter 6 Troubleshooting

Important Points to Remember	74
Application libraries	74
Device Libraries and Board Support Packages	74
tmcc/tmccom—The C/C++ Compiler	74
PSOS+ and C++	75
Interrupt Service Routines	76
tmdbg—The Debugger	76
tmprof—The Profiler	76
tmsim—The Simulator	77
tmman—The Execution Hosts	77
Troubleshooting Common Problems Under Windows.....	80
Frequently Asked Questions.....	84

About These Manuals

Conventions Used in These Manuals

These manuals use various conventions to present information. Words that require special treatment appear in special type faces. Certain information (program code, for example) appears in special formats so that you can scan it quickly.

Fonts and Colors

When standing alone, code listings, structure members, enumeration fields, function parameters, and return codes are shown in **monospaced type**.

When embedded in a paragraph, command line options and the names of structures, functions, macros, etc., are shown in **sans-serif bold type**.

Filenames and filename extensions (for example, .c or .o) are shown in normal type.

All TriMedia commands are shown in **bold type** (for example, **tmcc** and **tmdbg**).

These type conventions do not apply to titles and headers. All text in titles appears in the typeface normal for the title or header.

Certain code listings and command lines are presented in gray boxes.

Types of Notes

These manuals contains three types of notes.

Note

A note such as this contains information that is interesting, but possibly not essential to an understanding of the text. Notes can also tell you where to look for a more detailed discussion of a particular topic. ◆

IMPORTANT

A note such as this contains information that is essential for an understanding of the text. ▲

▲ WARNING

Warnings such as this indicate potential problems you should be aware of as you design your applications. Failure to heed these warnings could result in system crashes or loss of data. ▲

How to Get Help

Visit the TriMedia home page at www.semiconductors.philips.com/trimedia/ for more information.

Chapter 1

Overview of the TriMedia System

Topic	Page
TriMedia VLIW Architecture	10
TriMedia Software Streaming Architecture	12
TriMedia Hardware Components	13
TriMedia Software Components	15
SDE Documentation	19

TriMedia VLIW Architecture

TriMedia's DSPCPU family delivers exceptional performance and high-level language programmability for multimedia applications through the use of its VLIW (very long instruction word) architecture. TriMedia's VLIW architecture combines innovations in compiler and software design with advances in logic design.

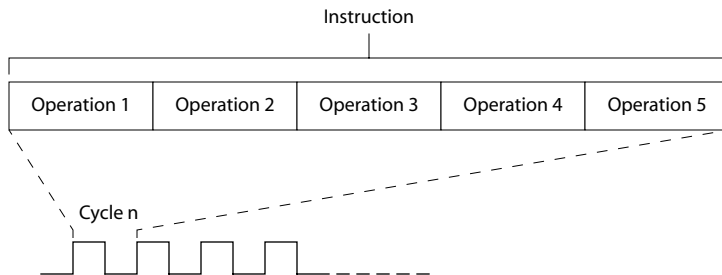
Rather than supporting only general purpose code as other vendors do, TriMedia's VLIW architecture supports multimedia-specific code that takes full advantage of the architecture. This code (along with a C/C++ compiler, Debugger, and other tools) is supplied to you in the form of application libraries as part of the TriMedia Software Development Environment (SDE).

Overview of The TriMedia VLIW Architecture

The TriMedia architecture is based on a three-level hierarchy of operators:

- Instructions
- Operations
- RISC operations

One instruction may contain five operations. Each operation may execute multiple arithmetic operations.



An example of one such operation is the command `ifir(a,b)`. This command contains a total of three arithmetic operations: *two* multiplications and *one* addition ($a_{HI} \times b_{HI} + a_{LO} \times b_{LO}$). Up to five operations including two `ifir` commands can be issued in each machine cycle.

The ability of TriMedia's VLIW architecture to execute multiple operations in parallel gives it a big advantage over traditional RISC and CISC architectures found in current mass-market microprocessors.

Functional Unit Assignment

Although the VLIW architecture allows for five operations to be executed per instruction, most operations cannot use just any slot because each functional unit of the Tri-

Media CPU is assigned to specific issue slots in a TriMedia VLIW instruction. Table 1 gives the functional unit assignments.

Table 1 Functional Unit Assignments

Functional Unit	Quantity	Latency ^A /Delay ^B	Recovery ^C	Slot Assignment				
				1	2	3	4	5
Constant	5	1	1	✓	✓	✓	✓	✓
Integer ALU	5	1	1	✓	✓	✓	✓	✓
Load/Store	2	3	1				✓	✓
DSP ALU	2	2	1	✓		✓		
DSP MUL	2	3	1		✓	✓		
Shifter	2	1	1	✓	✓			
Branch	3	3	1		✓	✓	✓	
Int/Float MUL	2	3	1		✓	✓		
Float ALU	2	3	1	✓			✓	
Float Compare	1	1	1			✓		
Float sqrt/div	1	17	16		✓			

A. Clock cycles between the issuance of an operation and availability of its results

B. Clock cycles between the execution of a branch instruction and the branching taking place

C. Minimum number of clock cycles between successive operations

Because of the number of available functional units and their assignment, some operations may have to wait for one or more cycles before they are executed. This means that in some cases not all issue slots are used. Good use of issue slots is one of the purposes for software code optimization.

What Makes TriMedia's VLIW Architecture Better

The beginning instruction set architecture (the processor programming model) must be distinguished from implementation (the physical chip and its characteristics). VLIW microprocessors and superscalar implementations of traditional instruction sets share some characteristics such as multiple execution units and the ability to execute multiple operations simultaneously.

The techniques used to achieve high performance are different because the parallelism is explicit in VLIW instructions, but must be discovered by hardware at run time by superscalar processors. VLIW implementations are simpler for very high performance. Just as RISC architectures permit simpler, cheaper high-performance implementations than do CISC architectures, VLIW architectures are simpler and cheaper than RISC architectures because of hardware simplifications. However, they require more compiler support.

The TriMedia VLIW architecture optimizes parallelism at compile time using its sophisticated compilation system. This makes the core less complicated (and feasible) and does not require specialized scheduling hardware at run-time. As a result, the TriMedia processors have much simpler control logic than other parallel designs and can run at higher clock speeds.

Performance

The best performance measurement is actual application measurement. Experience shows that complex DSP algorithms ported to TriMedia take between 1.0 and 0.5 times the number of instruction cycles as on other DSPs. However, while these algorithms are optimized in C for TriMedia, DSPs are traditionally optimized in assembly language.

In some algorithms, very high peak performance can be achieved. This is one of the keys to the optimization of the MPEG decoding algorithm. Because as many as 4 “RISC” operations can be performed in one TM operation, and because up to five TM operations can be executed at the cycle rate of 100 MHz, peak execution rates of 500 million operations per second are possible.

The mixture of programmability in C and efficient signal processing loops makes TriMedia particularly suited for the complex signal processing required by today's multimedia.

TriMedia Software Streaming Architecture

In a software driven system like TriMedia, it is important that the authors of diverse components agree upon some ground rules governing the connections between components.

Philips has made available this set of rules in the TriMedia Software Streaming Architecture (TSSA). If you are familiar with other streaming architectures (such as Microsoft's ActiveMovie™ and Apple's QuickTime™), you will see many similarities.

The TSSA:

- Defines formats and data structures to describe common multimedia data types.
- Defines procedures to start up and shut down components.
- Defines a method of connection between data sources and data sinks.
- Describes rules to determine who owns memory allocated by components.

Detailed information is contained in *Book 3, Software Architecture*.

TriMedia Hardware Components

This section describes the hardware components of the TriMedia system:

- The TriMedia Processor
- The TriMedia Board

The TriMedia Processor

The TriMedia processor can be used as a stand-alone processor, as a coprocessor to a more traditional CPU, or as one of a group of TriMedia chips arranged in a multi-processing configuration.

The 32-bit variants of the TriMedia processor (see Figure 1) have a number of special features that help to accelerate target applications. Their DMA driven I/O units operate independently to format data. Peripheral units are included for video in/out and audio in/out. Additionally, independent DMA driven coprocessors are available to perform key multimedia operations.

For example, the image coprocessor copies images from the synchronous DRAM (SDRAM) to the host's video frame buffer while simultaneously performing scaling and color space conversion. A peripheral block for Variable Length Decoding (VLD) assists in the decompression of MPEG-1 and MPEG-2 data streams.

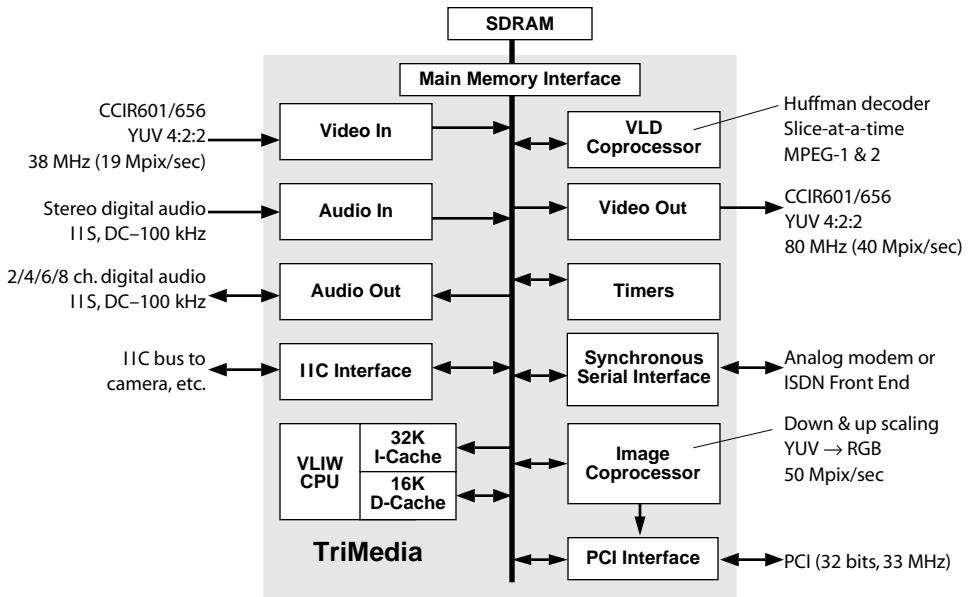


Figure 1 TriMedia Block Diagram

The TriMedia (TM-1xxx/TM-2xxx) IREF Board

The TriMedia board allows Windows 95, Windows NT, and Macintosh platforms to take advantage of the TriMedia processor. The TriMedia board is installed in an available PCI slot in the target platform.

The TriMedia SDE includes a board support package (BSP) that functions as a driver interface for the TriMedia board. If you create your own board, you will have to create your own BSP.

The BSP enables you to run all the TriMedia examples that come with the TriMedia SDE. For more information, see [Chapter 19, *TMBoard API*](#), of Book 5, *System Utilities*, Part C.

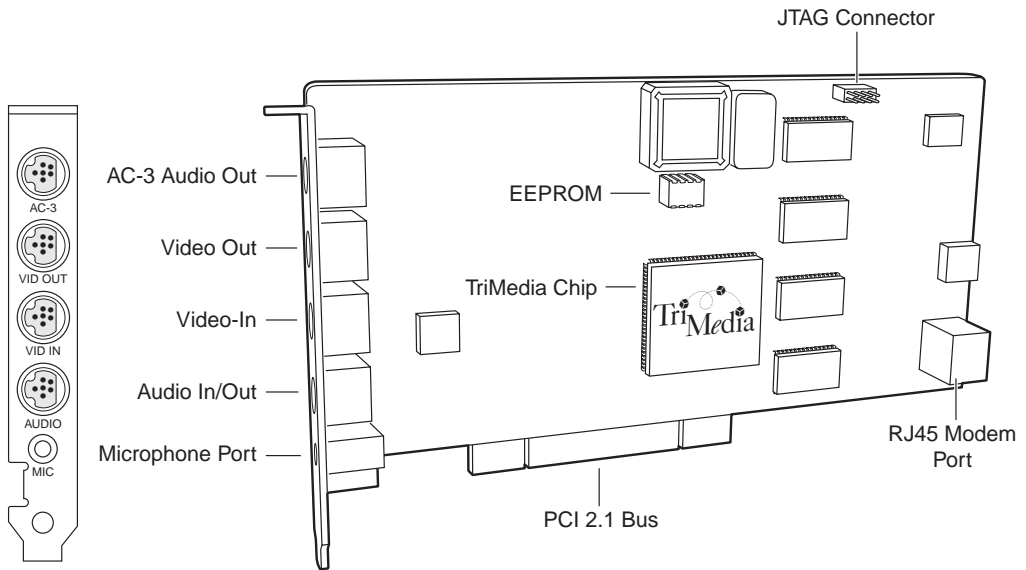


Figure 2 The TriMedia IREF Board and Ports

TriMedia Software Components

The TriMedia processor is supported by a robust, open TriMedia Software Development Environment (SDE) that speeds creation of highly optimized multimedia applications entirely in the C and C++ programming languages. The TriMedia SDE provides a comprehensive suite of multimedia libraries and system software tools to compile and debug multimedia applications, analyze and optimize performance, and simulate execution on the TriMedia processor.

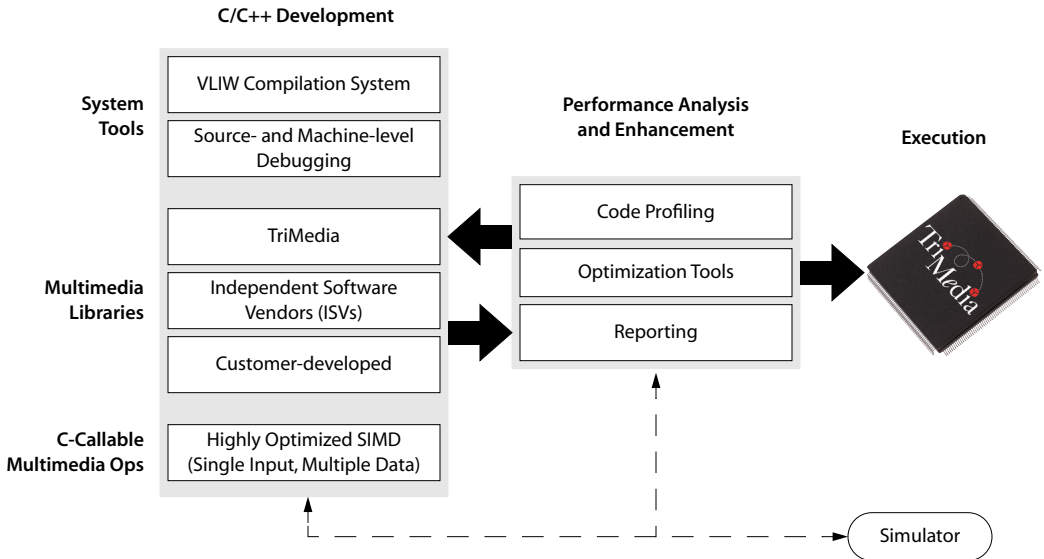


Figure 3 TriMedia Software Components

The TriMedia SDE features the following:

- ANSI-compliant C/C++ compilation system
- Source- and machine-level debugging
- Performance analysis and enhancement tools
- Cycle-accurate machine-level simulator
- TriMedia device and application libraries
- Example programs for application libraries and all on-chip peripherals
- Comprehensive online documentation
- Includes pSOS operating systems from Integrated Systems, Inc. (ISI)

TriMedia Tools

Figure 4 shows the TriMedia SDE tools.

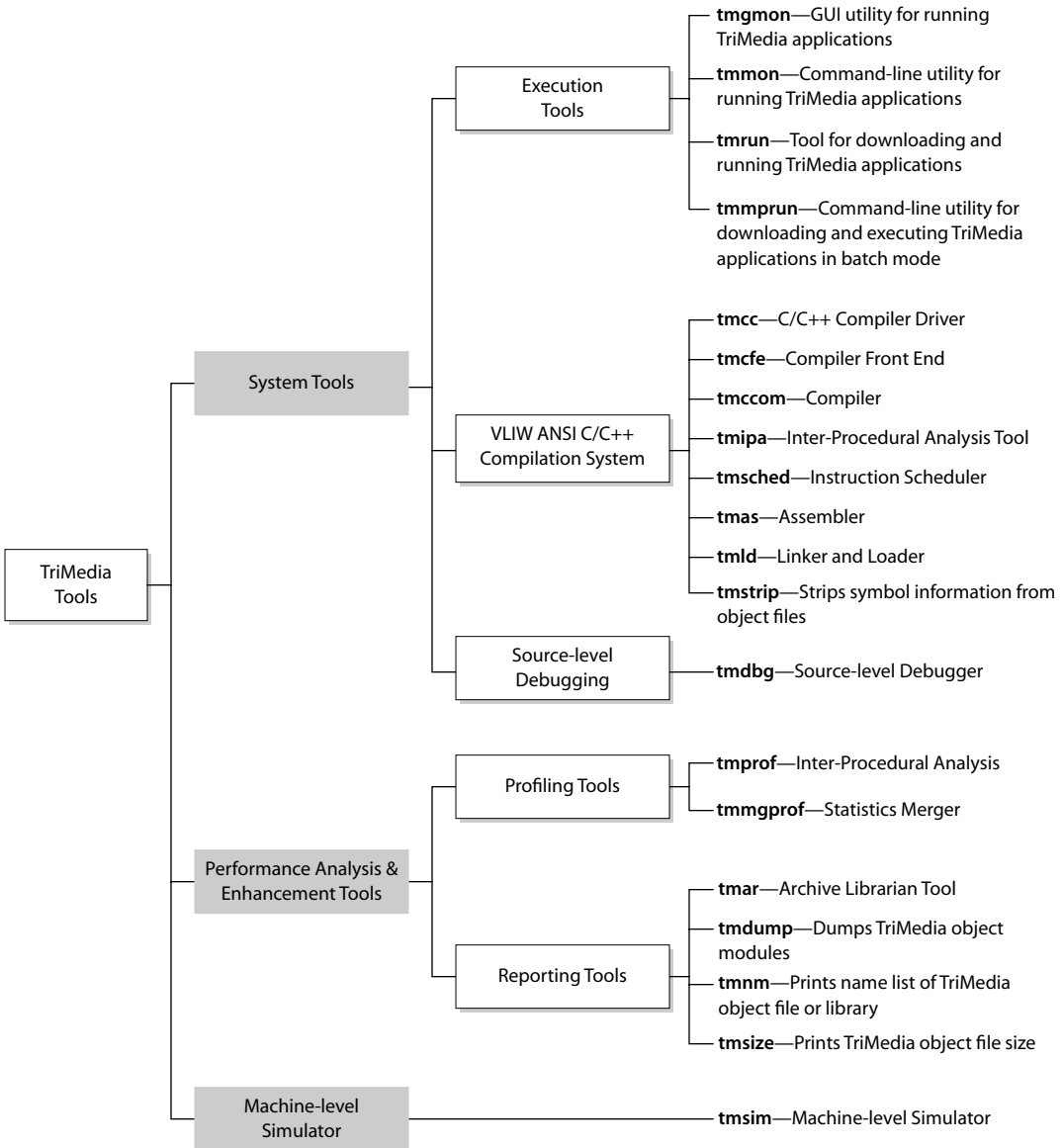


Figure 4 TriMedia SDE Tools

Execution Tools

Table 2 describes the TriMedia Execution tools.

Table 2 Execution Tools

Tool	Description
tmgmon	GUI utility for running TriMedia applications.
tmmon	Command-line utility for running TriMedia applications.
tmsrun	Tool for downloading and running TriMedia applications.
tmmprun	Command-line utility for downloading and executing TriMedia applications in batch mode.

System Tools

Table 3 describes the TriMedia VLIW ANSI C/C++ Compilation System Tools.

Table 3 VLIW ANSI C/C++ Compilation System

Tool	Description
tmcc	C/C++ Compiler Driver.
tmcfe	Processes a C/C++ source file and produces input for tmccom.
tmipa	Inter-Procedural Analysis Tool.
tmccom	The C/C++ compiler module of the TriMedia Compilation System.
tmsched	The TriMedia instruction scheduler. It schedules the TriMedia intermediate code input file (.t) for the target machine specified by the machine description file (.md) and produces the TriMedia assembly code file (.s).
tmas	The TriMedia assembler.
tmlld	TriMedia linker and loader. Links files together to produce a linked object module or executable, loading necessary modules from specified libraries
tmstrip	Strips symbol information from object files.

Table 4 describes the TriMedia Source-level Debugging Tool.

Table 4 Source-level Debugging

Tool	Description
tmdbg	A utility for source-level debugging and execution of programs written in ANSI C for a TriMedia VLIW processor.

Performance Analysis and Enhancement Tools

Table 5 describes the TriMedia Profiling Tools.

Table 5 Profiling Tools

Tool	Description
tmprof	Generate estimated execution profile.
tmmgprof	Merges the profiling statistics from specified files.

Table 6 describes the TriMedia Reporting Tools.

Table 6 Reporting Tools

Tool	Description
tmar	Allows the user to build a library, print the contents of a library, delete or replace the modules in a library, extract modules from a library, or print modules from a library
tmdump	Dumps a TriMedia format object file or library in readable form.
tmmnm	Prints the name list (symbol table) of a TriMedia format object file or library table of contents (__.SYMDEF module).
tmsize	Prints the sizes of the sections (segments) in a TriMedia format object file or library.
tmcc++filt	TriMedia C++ stand-alone name demangler.

Machine-Level Simulator

Table 7 describes the TriMedia Simulation Tool.

Table 7 Simulation Tool

Tool	Description
tmsim	The TriMedia machine-level simulator. It simulates the execution of a program on a specific TriMedia machine.

SDE Documentation

The Philips TriMedia SDE documentation is structured as follows:

Table 8 Philips TriMedia SDE Documentation Set

Book Name	Description		Format
Book 1: Getting Started	Contains a broad-brush overview of the TriMedia system hardware and software components as well as instructions for installing the TriMedia system. In addition, this manual guides you to finding information in the document chain.		Hardcopy Online (pdf)
Book 2: Cookbook	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	Developing TriMedia Applications	
	B	Programming with Peripherals	
	C	Bootstrapping TriMedia	
	D	Optimizing TriMedia Applications	
The four parts of the SDE Cookbook emphasize the how-to's for all stages of applications development. Use the sections of the Cookbook as an extended, and somewhat sophisticated, tutorial.			
Book 3: Software Architecture	This book discusses the concepts and reasoning behind the TriMedia Software Architecture (TSA).		Online (pdf)
	Part	Name	
	A	Foundation	
	B	The Streaming Architecture	
Book 4: Software Tools	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	C Language Users Guide	
	B	Program Development Tools	
	C	TriMedia Debugger	
The three parts of Book 4 serve as a detailed reference to all software components of the TriMedia system. For more information on TriMedia's VLIW architecture, refer to the appropriate TriMedia Data Book.			

Table 8 Philips TriMedia SDE Documentation Set (Continued)

Book Name	Description		Format
Book 5: System Utilities	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	Support Libraries	
	B	Examples of TSSA Components	
	C	System Device Libraries	
	D	MPEG System Components	
Book 5 provides reference material for system libraries.			
Book 6: Audio Support Libraries	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	I/O and Control	
	B	Codecs	
Book 6 presents the audio libraries.			
Book 7: Video Support Libraries	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	Video I/O	
	B	Video Processing and Coding	
Book 7 presents the video libraries.			
Book 8: Graphics Libraries	Book 8 presents the graphics libraries, some of which address closed captioning and some of which address HTML.		Online (pdf)
Book 9: Communications Libraries	Book 9 presents the communications libraries.		Online (pdf)
Book 10: ^A DTV Libraries	Consists of the following parts:		Online (pdf)
	Part	Name	
	A	DTV Application	
	B	Nx2x00 Peripherals and TSSA Components	
Book 10 contains reference material specific to DTV developers.			

A. The DTV book is not available in certain SDE releases.

Chapter 2

Installing the TriMedia System

Topic	Page
Introduction	22
Installing TriMedia Software and Hardware on the PC	23
Installing the TCS on UNIX-Based Platforms	34

Introduction

This chapter provides detailed information for installing the TriMedia SDE, also referred to as the TriMedia Compilation System (TCS), and TriMedia development boards, like the TriMedia IREF Board on the PC. It also describes how to install the TriMedia SDE on UNIX platforms.

SDE v2.1 ships with 2 CDs (compact discs). Disc 1 contains the installation for UNIX platforms and disc 2 contains the installation for PC platforms. The Metrowerks CodeWarrior plug-ins for MacOS 7.0 and above, Windows 95, Windows 98 and Windows NT are also on disc 2. For information on how to install the Metrowerks CodeWarrior plug-ins, refer to the document Metrowerks/ReleaseNotes on disc 2.

IMPORTANT

You can use UNIX-based platforms as TriMedia compilation (build) hosts only.

Installing TriMedia Software and Hardware on the PC

The TriMedia development system (Software Development Environment, or SDE) can be installed on a PC to create a complete development and test environment. You can also install TriMedia hardware on a PC with or without the complete SDE. Portions of the SDE are required to use the reference board. In the same way, it is possible to install the SDE on a machine that does not include TriMedia hardware.

Supported PC Build Hosts

The following are the supported operating system levels for PC build hosts:

- Windows 95 4.0 SR2
- Windows NT 4.0

Supported PC Execution Hosts

The v2.1 final release supports the following execution hosts:

- Windows 95
- Windows NT 4.0 (SP3)
- Windows 98
- Windows 2000 (Beta 3.0)
- Windows CE 2.11

Installing the TriMedia SDE

The PC can function both as a build host and as an execution host. The TriMedia compiler runs on build hosts such as SunOS, Solaris, Windows, and HP-UX. The TriMedia reference board (IREF or DTV) plugs into an execution host, allowing you to execute TriMedia code on the actual hardware. The Windows platforms are the only execution hosts currently available.

Although an execution host can also be used as a build host, it is a common practice to use different hosts for building and running. For example, you might use two machines connected by a network, where you build on one machine but run on the other. This keeps the build environment active if your program crashes the execution host.

The early chapters of the TriMedia cookbook provide more information on building and running your programs.

IMPORTANT

For an execution host, it is important that no previous version of the TriMedia SDE is installed on your PC. It is possible, but not recommended, to keep two versions of the build environment on one PC:

—Remove versions of `tmman32.dll`, `vtmman.vxd`, `vtmman.sys`, and `tmman.sys` from the path. These are likely to be found in the Windows directory.

—Remove references to `vtmman.vxd` from `system.ini`. s

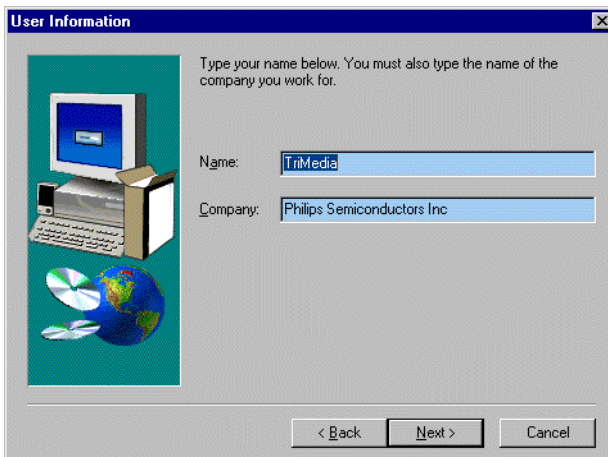
To install the TriMedia SDE on your Windows computer, perform the procedures described in the following sections.

Using the Setup Program

The TriMedia SDE CD includes `setup.exe` in the `WINDOWS` directory under the root directory of disc 2. This program automates much of the installation process. To install the complete SDE, run that program and follow its directions as outlined below.

Entering Personal Information

Fill in your name and your company's name:



Choosing the Location of the Installation Folder

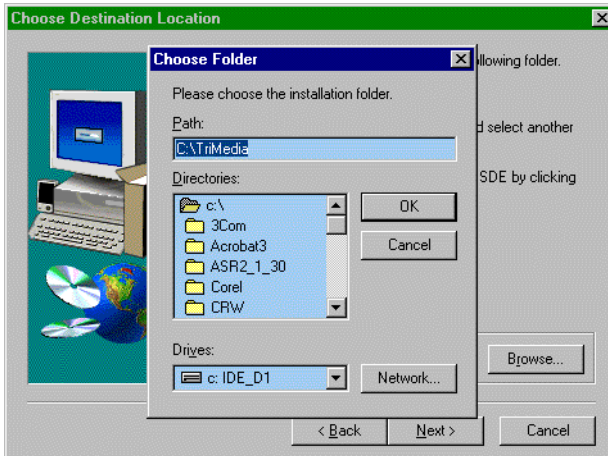
Next you are asked to choose the location of the installation.

1. Click the Browse button if you want to specify an installation directory.
2. By default, the wizard installs the TriMedia SDE in the `C:\TriMedia` directory. When you click the Browse button, the Choose Folder window appears. We recommend that you use a standard place.

- Choose the directory in which you want to install the TriMedia SDE from the Directories panel.

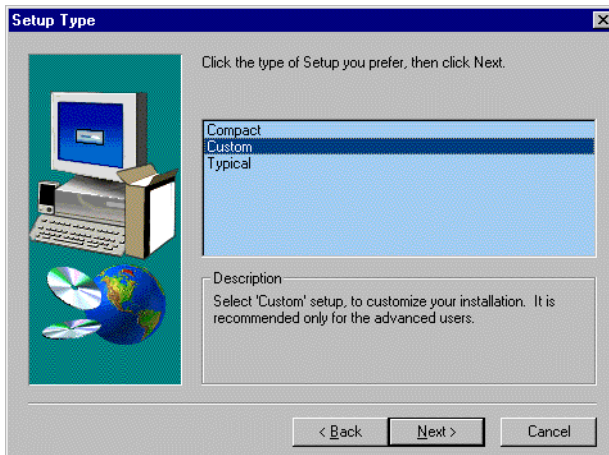
The installation wizard asks you whether you want a new folder to be created to use as the installation folder.

- Type the name of the installation folder in the Path field.



- Click OK.
- Click Next.

The Setup window appears.



Specifying the Installation Type

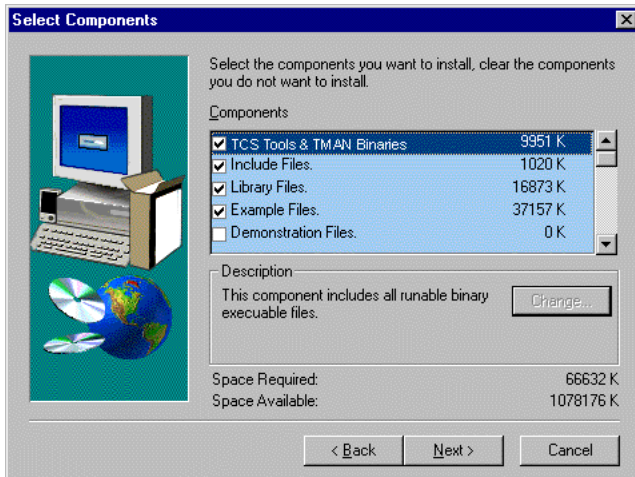
The Setup window allows you to choose from one of the following installation types:

- Typical

Installs all files.

- Custom

Allows you to choose the components you want to install. The Executable Binary File option must always be selected.



- Compact

Installs the following items only:

- TCS Tools & TMAN Binaries
- Include Files
- Library Files
- Example Files
- pSOS Files

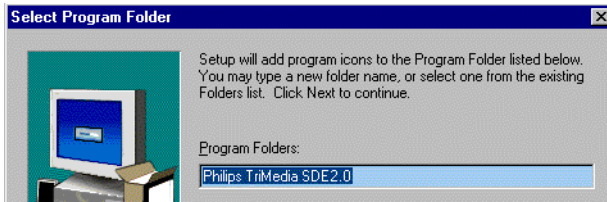
Omitted items include:

- Demos
- Documentation
- man pages
- Acrobat

7. Choose the desired installation type.

8. Click Next.

The wizard asks you (under Windows 95 or Windows NT) for the name of the program folder where `tmgmon.exe` and a readme file will be stored.



The default name for program folder is Philips TriMedia SDE 2.0.

9. Click Next.

The wizard copies the necessary files into the installation and program folders. In addition, the wizard creates the `TriMedia.bat` file and asks you to make the necessary changes to the `Autoexec.bat` file on your system. The wizard does not automatically update your `Autoexec.bat`. In addition, the wizard asks you to review the readme file and launch `tmgmon`.

Note

Use the WordPad program (Start\Programs\Accessories) to view the Windows 95 and Windows NT readme files.

Finishing the Installation

10. Click Finish to exit the installation wizard.

A Note on Installation Requirements For the TriMedia Debugger

The TriMedia debugger `tmdbg` uses sockets for interprocess communication when used with a `tmsim` target. Consequently, the host operating system must support sockets.

On a Windows 95 system, you must install an Ethernet card and TCP/IP networking (even if the host machine is not attached to a network) and make sure the machine has an assigned IP address.

Select: **Control Panels>>Network**. If TCP/IP is not installed, click Add and follow directions to install TCP/IP. Once it is installed, select: **Control Panels>>Network>>TCP/IP>>Properties>>IP Address** and make sure the machine has an assigned IP address (not "Obtain an IP address automatically"). `tmdbg` will report a `gethostbyname` failure if it cannot find an IP address for the host machine.

Installing the TriMedia Reference Board

This section describes how to install a TriMedia reference board, such as the IREF board into a Windows PC for hosted operation. The drivers for the various flavors of Windows now use very similar mechanisms.

JTAG-Hosted Operation

While it is very convenient to develop with TriMedia hosted by a Windows PC, it is also possible to develop with TriMedia configured to boot from ROM on a stand-alone board. In this case, the debugger connected to TriMedia over the JTAG bus is a very powerful tool.

1. Turn off your PC.
2. Install the JTAG board in an available ISA slot.
3. Boot the PC.
4. Install the debugger's JTAG kernel-mode driver, JTAG.SYS.

Installing JTAG.SYS on Windows NT

1. Log in as an administrator or as a user with administrative privileges.
2. If an old jtag.sys driver is already installed, you need not run the driver.exe program again. Simply copy jtag.sys from the Windows\data\bin\bin directory of the disc 2 CD into WinNT\system32\drivers directory and, at the Command Prompt, enter the following two commands:

```
C:\> net stop jtag  
C:\> net start jtag
```

3. For a new installation, copy jtag.sys from the Windows\data\bin\bin directory of the CD into WinNT\system32\drivers directory and reboot.
4. Run the driver.exe program (located in the Windows\data\bin\WinNT directory on the TriMedia CD) with the following command options.

```
C:\trimedia\bin> driver -oi -sa -njtag
```

If for any reason the user needs to remove the kernel mode driver entry from the registry the driver utility can be used with the following parameters,

```
C:\trimedia\bin> driver -or -njtag
```

To reinstall the driver so it does not start automatically, use the following command:

```
C:\trimedia\bin> driver -oi -sm -njtag
```

This will make the start type of the driver a manual start instead of auto start. To start the driver manually type the following at the command prompt:

```
C:\> net start jtag
```

To stop the driver, manually type the following at the command prompt:

```
C:\> net stop jtag
```

The kernel mode driver `jtag.sys` should be present in the `WinNT\system32\drivers` sub-directory.

For more information, see [Book 4, *Software Tools, Part C*](#).

Installation Requirements for Hosted Operation

Before installing the TriMedia reference board and running programs, you should install the TriMedia SDE as described earlier. In addition, make sure that your system meets the requirements shown in Table 9. The limitations on graphics adapters are particularly important if you plan to use the image coprocessor (ICP) to display video on the PC's screen.

Table 9 System Requirements

CPU	Pentium ^A or Pentium Pro.
Motherboard	TriMedia generally complies with PCI specifications, so any reputable motherboard should work. If you are concerned, stick to Intel chipsets.
Graphics Adapter	<p>Following is a list of some of the adapters that have been tested against the TriMedia IREF board:</p> <ul style="list-style-type: none"> — ATI graphics adapters (strongly recommended) — Diamond graphics card <p>Some Diamond graphics adapters claim PCI memory without registering it with the PC's operating system. This makes it possible for TriMedia to conflict with the undeclared memory.</p> <p>WARNING</p> <p>Do not use Matrox boards. They have been tested against the 32-bit variants of the TriMedia IREF board and they don't work. Matrox Millennium boards place more restrictions on the alignment of a graphics window managed by an external device like the ICP. This will lead to unacceptable performance in graphics demos.</p>

A. Pentium II with at least 64MB of RAM is recommended

Installation Procedure

To install a TriMedia reference board, do the following:

WARNING

Do not abort this procedure. It is required for plug-and-play to assign resources to your board.

1. Turn off your PC.
2. Install the TriMedia reference board in an available PCI slot.
3. Boot the PC.

4. Follow the procedure below that applies to your operating system.
5. Update the registry to reflect your processor's speed (information follows).
6. Verify installation of the TriMedia SDE and the TriMedia IREF reference board by running demonstration programs as described in Chapter 3, *Running Demonstration Programs.*"

Installation of Updates

If you have been using TriMedia with earlier releases, your installation will require an extra step to remove the old drivers. Because new features are added, the drivers from a new release are sometimes not compatible with earlier releases. This information will be clearly noted in the release notes. Since the drivers for a Windows system are closely tied to the kernel of the OS, you must reboot to install new drivers. This makes it hard to use two versions of the drivers on the same machine. One possible solution is to set the machine up with multiple operating systems. That way, you are asked to choose the OS at boot time, and you can install one set of drivers under Win95 and another under WinNT. As of the SDE v2.0 beta release, Windows NT became the preferred operating system.

Windows NT Setup

If you have installed the complete SDE, then all files are already on your hard disk. If you want only to install the minimal set of drivers, you will need to copy most of the contents of the TriMedia\bin directory from disc 2 to your local hard disk.

If installing on Windows NT, access BIOS setup at startup and make sure that an interrupt is available for the PCI slot. In addition, to find out what resources TriMedia uses under Windows NT, go to Programs/Administration/NT Diagnostics panel and click on resources. Choose TriMedia and click on Devices.

1. Log in as an administrator or as a user with administrative privileges.
2. If an old tmman.sys driver is already installed, simply copy tmman.sys from the Windows\data/drivers\WinNT directory of the disc 2 CD into WinNT\system32\drivers directory and reboot. You need not run the driver.exe program again.
3. For new installation, copy tmman.sys from the Windows\data/drivers\WinNT directory of the disc 2 into WinNT\system32\drivers directory and reboot.
4. Run the driver.exe program (located in the Windows\data\bin\WinNT directory on the TriMedia SDE disc 2) with the appropriate flags.

```
C:\trimedia\bin> driver -oi -sa -ntmman
```

5. Update the registry as described below.

If for any reason the user needs to remove the kernel mode driver entry from the registry the driver utility can be used with the following parameters,

```
C:\trimedia\nt> driver -or -ntmman
```

To reinstall the driver so that it does not start automatically, use

```
C:\trimedia\nt> driver -oi -sm -ntmman
```

This will make the start type of the driver a manual start instead of auto start.

After the NT Kernel Mode driver has been successfully installed in the system, the registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\tmman have the following or similar values:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\tmman]
"DisplayName"="tmman"
"ImagePath"="\SystemRoot\System32\drivers\tmman.sys"
"TYPE"=dword:00000001
"Start"=dword:00000002
"ErrorControl"=dword:00000001
```

The driver can also be configured to start in manual mode by changing the value of **Start** to **dword:00000003**. To start the driver manually type the following at the command prompt

```
C:\> net start tmman
```

To stop the driver manually type the following at the command prompt

```
C:\> net stop tmman
```

The kernel mode driver tmman.sys should be present in the %system-root%\system32\drivers sub-directory.

All other files have to be in the current directory or in the executable search path.

Windows 95

For existing Win95 installations, the drivers that are currently installed have to be removed and the new drivers have to be installed, as described below.

1. The utility tmmanins.exe can be used to uninstall any existing drivers. This program automates the following three steps:
2. If the previous installation was not a PnP installation, then the TriMedia reference board will show up in the device manager as an "other device." Go to My Computer -> Properties -> Device Manager -> Other Devices. Double click on "Other Devices," highlight "PCI Multimedia Device" corresponding to TriMedia reference board, and select Remove. If the previous installation was a PnP installation, go to My Computer -> Properties -> Device Manager -> Sound, video and game controllers. Double click on "Sound, video and game controllers," highlight "TriMedia IREF board," and select Remove.
3. In the Windows directory c:\windows\inf\other, delete all files containing the string "tmman" (without quotes) in them. It is usually "Philips SemiconductorsTM-MAN.INF" or "Philips SemiconductorsVTMMAN.INF."
4. Delete c:\windows\system\vtmman.vxd.
5. Reboot.

6. Install the new TriMedia driver when the system prompts you. The driver information file `tmman95.inf` and the driver `tmman.vxd` are in the directory `Windows\data\drivers\Win95` on the SDE disc 2.

Windows 98

1. Delete any old drivers installed using the steps mentioned for Windows 95.
2. After rebooting, install the new TriMedia driver when the system prompts you to do so. The driver information file `tmman98.inf` and the driver `tmman.sys` are in the directory `Windows\data\drivers\Win98` on the SDE disc 2.

Windows CE 2.11

Please read the `readme.txt` file in the `Windows\data\drivers\WinCE` directory to see changes required in the `config.bib`, `platform.bib`, `platform.reg` and Kernel Mode OEM IOCTL code.

Updating the Registry

Run `tmgmon.exe`. The first time that you run this, the simple loader program, a number of important registry keys are created.

Before you can use the multimedia programs, you will need to set the processor speed registry entry. Click on Start -> Run and enter "regedit" as the name of the program you want to run. Be careful when using regedit.

Under `HKEY_LOCAL_MACHINE`, successively click on `SOFTWARE` -> `PhilipsSemiconductors` -> `TriMedia` -> `TMMAN` -> `Device0`. You should see a list of keys with their value. You must create a `clockspeed` entry. Select `Edit` -> `New` -> `DWORD Value`. An entry should appear. On the right, you should see a new key with a value of 0. Type in "clockspeed" and then click on the entry.

Enter the processor clockspeed in the data field (e.g. 150000000 for 150 MHz) and hit return. You must select 'decimal'.

To determine the clock speed, you need to know the oscillator frequency, and the memory and CPU multipliers. The oscillator is a silver, rectangular 4-pin package, physically close to the processor. The frequency is written on the top.

To determine the multiplier, you must run the `iicstest.out` program. It is situated under the `examples/bin/tm1` directory. Type the command "ps" to get the multipliers.

You should see something like the following:

```
Processor Speed variable reports TM running at 0.0 Mhz.  
EEPROM reports Memory to External ratio is 2:1  
EEPROM reports CPU to Memory Clock ratio is 3:2.
```

```
Your memory is running at 2 times the crystal frequency.  
Your CPU is running at 1.5 times the memory frequency.
```


For example, imagine “50.000M” is written on the oscillator package. Then the clock frequency is 150 Mhz (or $50 \times 2 \times 1.5$).

If you want to change the clock speed, just double-click on it. If you need to delete an entry you have created, click on that and type Edit->Delete.

```
HKEY_LOCAL_MACHINE\SOFTWARE\PhilipsSemiconductors\TriMedia
  tmman
    device0
      clockspeed = 125000000
```

Installing the TCS on UNIX-Based Platforms

Install TriMedia from the CDs, using an installation procedure based on your particular UNIX platform (see the procedures following). All UNIX installations use SDE disc 1.

WARNING

UNIX-based platforms function as TriMedia compilation hosts only. Do not plug a TriMedia reference board into a UNIX host.

Installing TriMedia on Solaris™

To install TriMedia on systems running the Solaris™ operating system, do the following:

1. Choose a directory to install the software.

Note

We recommend that you use a standard place because the remaining instructions assume the following:

- The CD-ROM is /dev/dsk/c0t6d0.
- The software is to be installed in /usr/local/tcs.
- There is a /cdrom mounting point.

2. Eject any mounted CD by typing the following:

```
eject /cdrom
```

Solaris™ will not allow you to open the CD-ROM drive if a CD is mounted.

3. Insert the TriMedia Software Distribution CD in the CD-ROM drive (You should be running openwin).

A dialogue box appears and the CD is mounted automatically under /cdrom/tcs11 (tcs11 is the label of the CD).

In addition, the volume manager (**vol**d) creates a directory under /cdrom corresponding to the label of the CD. You should change your working directory to that of the CD-ROM (cd /cdrom/tcs11).

4. To complete the installation of the software, type the following:

(if your system is Solaris 2.6)

```
setupsol /usr/local/tcs
```

(otherwise)

```
setupsun /usr/local/tcs
```

When this command completes, the software will be installed in /usr/local/tcs. The directory will be created automatically.

5. To remove the CD-ROM, click the eject button in the dialogue box.

These instructions assume **vold** is running. If this is not the case, you will have to mount the CD-ROM manually. You must be a super user to do this. Type the following:

```
su root
<enter password>
mount /dev/dsk/c0t6d0 /cdrom
```

The environmental variables **PATH** and **TMPDR** must be initialized in your profile in order to use the SDE. See the README file for more information.

Installing TriMedia on SunOS

To install TriMedia on systems running the SunOS operating system, do the following:

1. Choose a directory to install the software.

Note

We recommend that you use a standard place. The remaining instructions assume the following:

- The CD-ROM is `/dev/sr0`.
- The software is to be installed in `/usr/local/tcs`.
- There is a `/cdrom` mounting point.

2. Insert the TriMedia Software Distribution Disc 1 in the CD-ROM drive.
3. Mount the CD-ROM by typing the following (you must be a super user to do this):

```
su root
<enter password>
mount -ro /dev/sr0 /cdrom
```

4. Change your working directory to that of the CD-ROM by typing the following:

```
cd /cdrom
```

5. To complete the installation of the software, type the following:

```
setupsun /usr/local/tcs
```

It is preferable to execute the installation procedure under a login name other than **root**.

When this command completes, the software will be installed in `/usr/local/tcs`. The directory will be created automatically.

6. To remove the CD-ROM, type the following:

```
umount /dev/sr0
eject /cdrom
<control D>
```

The environmental variables **PATH** and **TMPDR** must be initialized in your profile in order to use the SDE. See the README file for more information.

Installing TriMedia on HP-UX

To install TriMedia on systems running the HP-UX operating system (version 10.x), do the following:

1. Choose a directory to install the software.

Note

We recommend that you use a standard place. The remaining instructions assume the following:

- The CD-ROM is /dev/dsk/c2d2t0.
- The software is to be installed in /usr/local/tcs.
- There is a /cdrom mounting point.

2. Insert the TriMedia Software Distribution Disc 1 in the CD-ROM drive.
3. Mount the CD-ROM by typing the following (you must be a super user to do this):

```
su root
<enter password>
mount -F cdfs /dev/dsk/c2d2t0 /cdrom
```

4. Change your working directory to that of the CD-ROM by typing the following:

```
cd /cdrom
```

5. To complete the installation of the software, type the following:

```
setuphp /usr/local/tcs
```

It is preferable to execute the installation procedure under a login name other than root.

When this command completes, the software will be installed in /usr/local/tcs. The directory will be created automatically.

6. To remove the CD-ROM, type the following:

```
umount /dev/dsk/c2d2t0
```

You can now remove the CD-ROM from the drive.

The environmental variables PATH and TMPDR must be initialized in your profile in order to use the SDE. See the README file for more information.

Chapter 3

Running Demonstration Programs

Topic	Page
Introduction	38
Running the fplay Program	39
Running the Files Program	41
Running the icptest Program	42
Running the exalVrendVO Program	44

Introduction

The installed version of the TriMedia SDE includes a set of demonstration programs that you can run on the 32-bit variants of the TriMedia chip.

Note

If you chose not to install the Examples Files and Demonstration Files, the examples and demo folders will not be available on your hard drive. However, these folders remain available on the CD in the Windows\Data folder.

The demonstration programs are designed to give you a flavor of the capabilities of the TriMedia processor. In addition, by studying how these programs work, you can learn how to write TriMedia programs. Refer to **Book 2, Cookbook, Part B**, for more information.

This chapter describes how to run the following demonstration programs:

- fplay
- Files
- icptest
- exalVrendVO

The programs **fplay**, **Files**, and **icptest** are stored in

`<installation_folder>\examples\bin\Windows\el\.`

The program **exalVrendVO** is stored in

`<installfolder>\apps\examples\bin\tml_WinNT_el\.`

The name of the installation folder is **TriMedia** by default.

You are encouraged to try the other demonstration programs on the CD. The supplied readme file provides the necessary information for running these programs.

IMPORTANT

Make sure that the directory which contains **tmgmon** and **tmrn.exe** is included in the path.

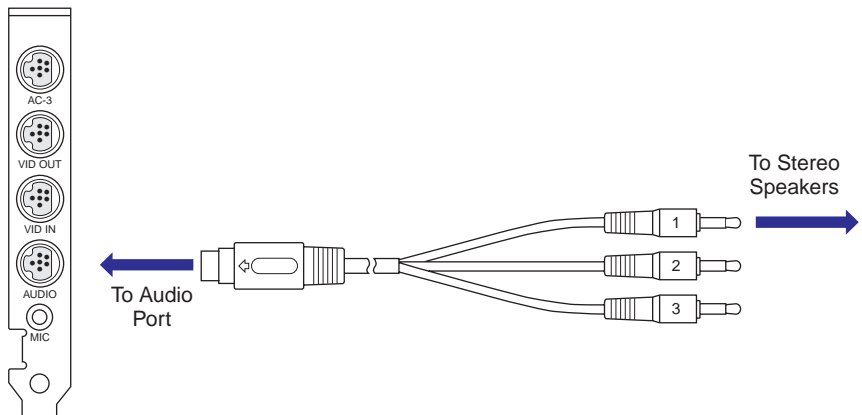
Running the fplay Program

The **fplay** program is useful for verifying installation of the TriMedia system. It loads and plays a 16-bit stereo sound file on the TriMedia chip. To run **fplay**, do the following:

1. Open the `<installation_folder>\examples\bin\Windows\el\` folder. If you use a name other than TriMedia for the installation folder, use it instead.
2. Connect the TriMedia IREF board to a set of powered stereo speakers.
3. Use the Audio cable that comes with your TriMedia package to connect the TriMedia IREF board to a set of powered stereo speakers through its Audio port.

Note

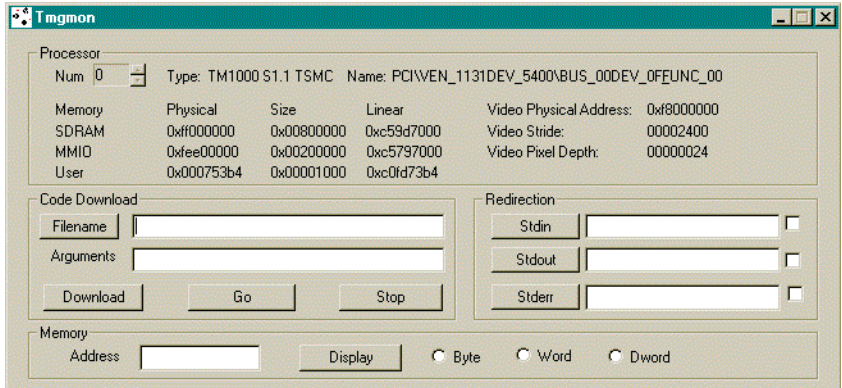
Plug the 9-pin miniDIN end of the Audio cable into the board's audio port and the audio jack 1 (audio out) into the stereo speakers. ◆



4. Make sure that `tmgmon.exe` and `tmrn.exe` are in the same folder as **fplay**.

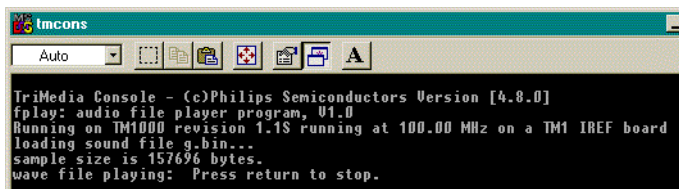
- Double-click `tmgmon.exe` (**tmgmon**).

The TriMedia Monitor window appears.



- Click the Filename button.
- Select `fplay.out`.
- Click Go.

The `tmrun` window appears describing the status of the execution. At the same time, you should be able to hear a looping sound of a guitar chord being strummed. If this does not happen, refer to Chapter 6, *Troubleshooting*, for possible causes and solutions. The data file “g.bin” is required in the same directory as the example executable file for this to work.



- Press any key to terminate the `tmrun.exe` program and go back to **tmgmon**. You can also use the Control-C key combination.

Running the Files Program

This program is useful for verifying TriMedia IREF board installation. It uses the `fread` and `fwrite` calls to transfer data back and forth across the PCI bus. The files program should run for hours without errors. Errors will be consistently generated (error messages are displayed in the `tmr` window) if there is a hardware compatibility problem. Refer to Chapter 6, *Troubleshooting*, for more information.

To run the files program, do the following:

1. Open the `installation_folder\examples\bin\Windows\el\` folder. If you use a name other than TriMedia for the installation folder, use it instead.

2. Double-click `tmgmon.exe`.

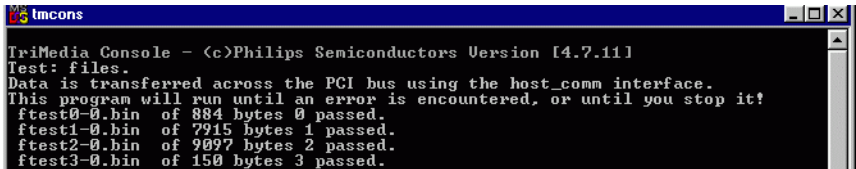
The TriMedia Monitor window appears.

3. Click the Filename button.

4. Select `files.out`.

5. Click Go.

The `tmr` window appears displaying.



```

tmcons
TriMedia Console - (c)Philips Semiconductors Version [4.7.11]
Test: files.
Data is transferred across the PCI bus using the host_comm interface.
This program will run until an error is encountered, or until you stop it!
ftest0-0.bin of 884 bytes 0 passed.
ftest1-0.bin of 7915 bytes 1 passed.
ftest2-0.bin of 9097 bytes 2 passed.
ftest3-0.bin of 150 bytes 3 passed.
  
```

6. Press Control-C to quit and go back to `tmgmon`.

Running the icptest Program

The icptest program shows you how to use TriMedia's ICP coprocessor. It writes directly to the PC's video memory (not standard Windows behavior). To run this program, you need to know the characteristics of your system's frame buffer. This information is available in the Device Manager under Windows 95 and through the diagnostics tool under Windows NT.

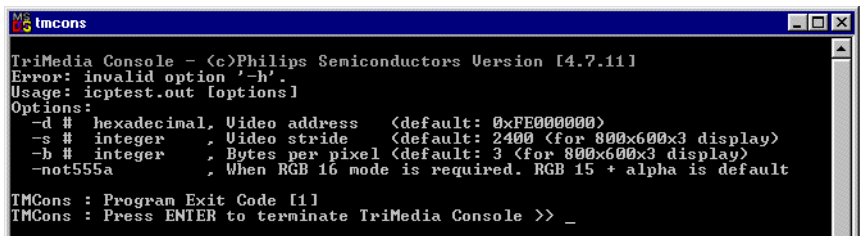
To run this program, do the following:

1. Open the `installation_folder\examples\bin\Windows\el\` folder. If you use a name other than TriMedia for the installation folder, use it instead.
2. Make sure the data files "tv.[yuv]" and "clown210x100.[yuv]" are in the same directory as the example executable.
3. Double-click `tmgmon.exe`.

The TriMedia Monitor window appears.

4. Click the Filename button.
5. Select `icptest.out`.
6. Type `-h` into the Arguments field.
7. Click Go.

The `tmrun` window appears listing the correct options.



```

TriMedia Console - (c)\Philips Semiconductors Version [4.7.11]
Error: invalid option '-h'.
Usage: icptest.out [options]
Options:
  -d # hexadecimal, Video address (default: 0xFE000000)
  -s # integer      , Video stride (default: 2400 (for 800x600x3 display)
  -b # integer      , Bytes per pixel (default: 3 (for 800x600x3 display)
  -not555a         , When RGB 16 mode is required. RGB 15 + alpha is default

TMCons : Program Exit Code [1]
TMCons : Press ENTER to terminate TriMedia Console >> _

```

8. Enter the appropriate arguments (for example `-not555a`).

To find out your video base board address to use with the `-d` option, do the following:

- Open the System control panel.
- Click the Device Manager tab.
- Select your display adapter.
- Click the Properties button.
- Click the Resources tab in the Properties window of the display adapter.

The video board base address is the last item in the Resource settings list.

9. Click Go.

The `tmrun` window appears listing five menu options.

10. Type 2 and press Enter.

A picture of a clown moves diagonally across a background picture.



11. Type 4 and press Enter.

The picture of the clown appears at the middle of the background picture and moves down while being scaled.

12. Try options 1 and 2.

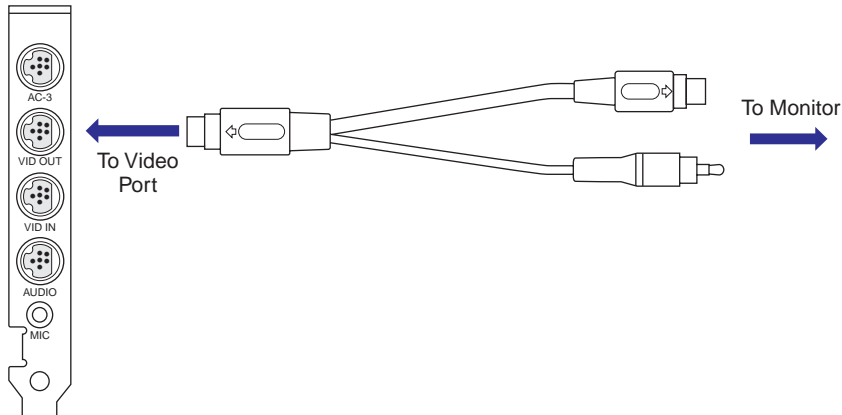
13. When done, use option 5 to quit.

14. Press Enter to return to **tmgmon**.

Running the exalVrendVO Program

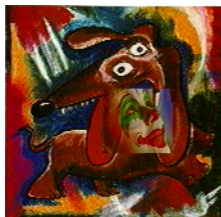
The **exalVrendVO** program (in `<installfolder>\apps\examples\bin\tml_WinNT_el\`) demonstrates the overlay and alpha blend functions of the TriMedia Video Out unit. It is an example of an application library that is built on top of the Video Out library. To run this program, do the following:

1. Connect the TriMedia IREF board to a video monitor using one of the video cables supplied with the board.



2. Open the `<installation_folder>\examples\bin\Windows\app_el\` folder. If you use a name other than TriMedia for the installation folder, use it instead.
3. Double-click `tmgmon.exe`.
The TriMedia Monitor window appears.
4. Click the Filename button.
5. Select `exalVrendVO.app`.
6. Click go.

The console window appears and the program runs through four 10-second long demonstration modes. The fourth mode performs video input to video output transfer that you can test by connecting a camcorder to the board's video in port.



7. Press Control-C to go back to **tmgmon**.

Chapter 4

Using TriMedia SDE Tools

Topic	Page
Introduction	46
Sample Program: getIt.c	46
Compiling TriMedia Applications	47
Simulating TriMedia Applications	48
Running TriMedia Applications on the TriMedia Processor	49
Debugging TriMedia Applications	50
Where to Go from Here	54

Introduction

This chapter shows, step by step, how to compile, simulate, process, and debug `getIt.c`, a TriMedia sample program, using `tmcc`, `tmsim`, `tmgmon/tmrun`, and `tmdbg` respectively.

Sample Program: `getIt.c`

The step-by-step examples covered in the remaining sections of this chapter are based on the `getIt.c` program:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <tmLib/tmTypes.h>

#define INPUT 30

static Float factorial( Int input )
{
    Int i;
    Float result, help;

    for( i=input, result=1; i>0; i-- )
    {
        help = (Float)i;
        result *= help;
    }
    return result;
}

void main( void )
{
    Float finalResult;

    printf("\nIt's getting started !!!!!!!!!!!!!\n");
    printf("...calculating the factorial of %d...\n", INPUT);

    finalResult = factorial(INPUT);

    printf("\nThe result is %le.\n\n", finalResult);
    exit(0);
}
```

The `getIt.c` program calculates the factorial of the value of `input`.

Compiling TriMedia Applications

To compile TriMedia applications, use the **tmcc** compiler driver to compile C programs. **tmcc**, **tmcpp** and **tmCC** invoke the C/C++ Front End (**tmcfe**), the Core Compiler (**tmccom**), the Scheduler (**tmsched**), the Assembler (**tmas**), and the Linker (**tmlld**).

Note

Before proceeding with this section, make sure that your AUTOEXEC.BAT file contains the path to the TriMedia SDE tools (...\\TriMedia\\bin).

Use the following two examples to compile the getIt.c program, listed in the section “Sample Program: getIt.c” beginning on page 46, to run on the TriMedia Simulator (**tmsim**) and TriMedia Processor respectively:

- To compile the getIt.c program to run on the **tmsim**, enter the following command line at the DOS prompt:

```
tmcc -o getIt.out getIt.c
```

The **-o** option allows you to specify the target file getIt.out. The default target is a.out.

- To compile the getIt.c program to run on a TriMedia Processor hosted on a Windows 95 platform, enter the following command line at the DOS prompt:

```
tmcc -host Win95 -o getIt.out getIt.c
```

In this example, unlike the previous one, you must use the **-host** option to specify the system that hosts the TriMedia processor. If you don't use this option, the target will be compiled to run on **tmsim**.

For more detailed information about compiling TriMedia applications, refer to:

- [Chapter 1, *Compiling TriMedia Applications*](#), in Book 2, the *Cookbook*, Part A.
- Book 4, [Software Tools](#), Part A, especially [Chapter 2, *Using the C Compiler*](#).

Also, for more detailed information about the different techniques for generating optimized code (such as using restricted pointers, loop unrolling, profile-driven compilation, custom operators, loop optimization, and decision-tree grafting), refer to Book 2, the *Cookbook*, Part D.

Simulating TriMedia Applications

Simulating TriMedia applications is ideal for working with inner loops. **tmsim** generates accurate cycle data that can be used to optimize each behavior. If speed is a concern, however, or when using peripheral I/O, it's recommended that you run your programs on the TriMedia processor.

To simulate the getIt.out program, enter the following command line at the DOS prompt:

```
tmsim getIt.out
```

tmsim generates the following output:

```
It's getting started !!!!!!!!!!!!!!!  
...calculating the factorial of 30...  
The result is 2.652529e+32.
```

If you want **tmsim** to generate performance reports, you must use certain options when compiling the getIt.c program.

For more information about **tmsim** and how to use it to generate performance reports, refer to:

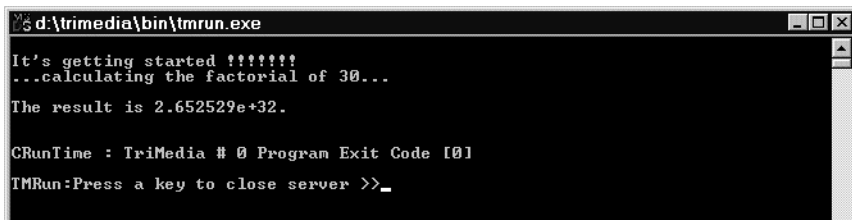
- Book 2, the *Cookbook*, Part D, especially [Chapter 10, *Porting and Optimizing Programs*](#).
- [Chapter 13, *The TriMedia Simulator*](#), in Book 4, *Software Tools*, Part B.

Running TriMedia Applications on the TriMedia Processor

To run the `getIt.out` program that you generated (in *Compiling TriMedia Applications* on page 47) on the TriMedia processor, enter the following command line at the DOS prompt:

```
tmrn getIt.out
```

tmrn downloads the program and runs it on the TriMedia processor. The following output appears on the screen:



```
d:\trimedia\bin\tmrn.exe
It's getting started !!!!!!!
..calculating the factorial of 30...
The result is 2.652529e+32.

CRunTime : TriMedia # 0 Program Exit Code [0]
TMRn:Press a key to close server >>_
```

For more detailed information about running TriMedia applications, refer to:

- [Chapter 1, *Compiling TriMedia Applications*](#), in Book 2, the *Cookbook*, Part A.
- Book 4, *Software Tools*, Part A, especially [Chapter 2, *Using the C Compiler*](#).

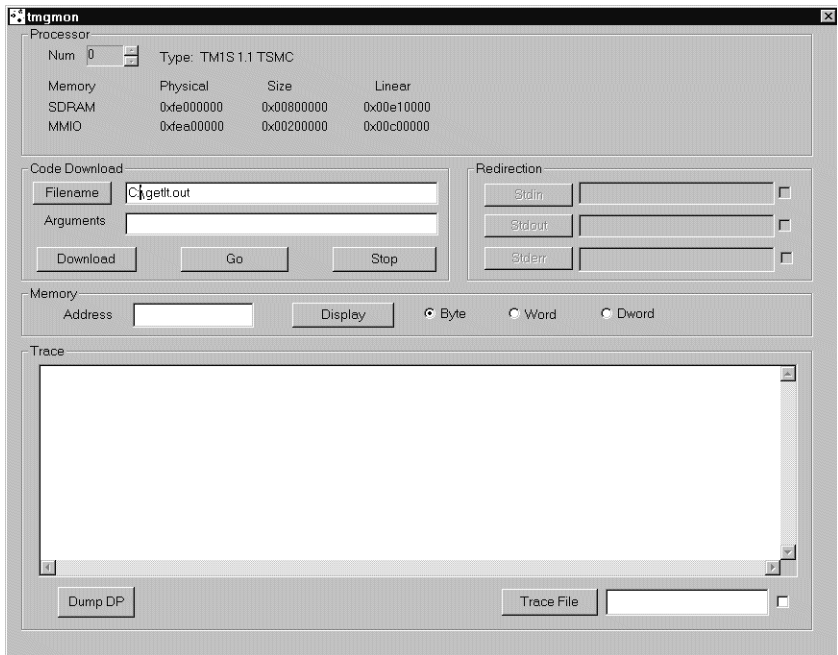
You can also use **tmgmon** to run programs on the TriMedia processor. To use **tmgmon**, do the following:

1. Enter the following command line at the DOS prompt:

```
tmgmon
```

The **tmgmon** window appears.

- Use the Browse button to locate the getIt.out file.



- Click Go.

Debugging TriMedia Applications

To debug getIt.c, do the following:

- Compile getIt.c using the **-g** and **-O0** options.

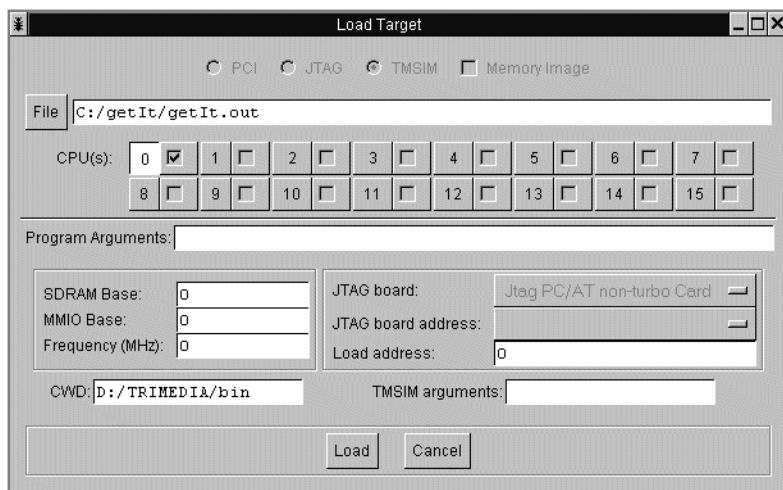
```
tmcc -g -O0 -o getIt.out getIt.c
```

The **-g** option generates the symbolic debugging information and links the program with the debug monitor library and the **-O0** option disables global optimization.

- Start the TriMedia Debugger with:

```
tmdbg
```

3. Choose Load Target from the File menu and load getIt.out.



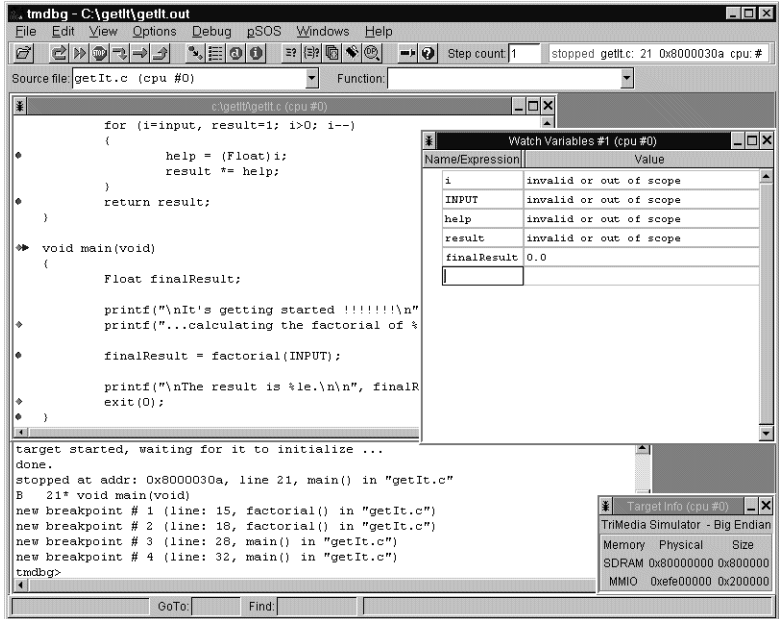
4. Enable CPU 0 by clicking the checkbox to its right and click Load.

tmdbg loads the program into the TriMedia processor and stops execution at the function **main**. In addition, **tmdbg** displays the source code of the program in the `<path>getIt.c (cpu #0)` window.

In the `<path>getIt.c (cpu #0)` window, the green “diamonds” at the beginning of lines are decision tree markers (also referred to as breakpoint markers) that you toggle-click to enable and disable breakpoints. When you enable a decision tree marker, it becomes red.

Decision tree markers appear next to lines that are on the borders of decision trees. In the language of TriMedia intermediate code, a decision tree marker appears next to a **jump** instruction. For more information, refer to Book 4, *Software Tools*, Part A, especially the *Decision Tree Syntax* section in Chapter 4.

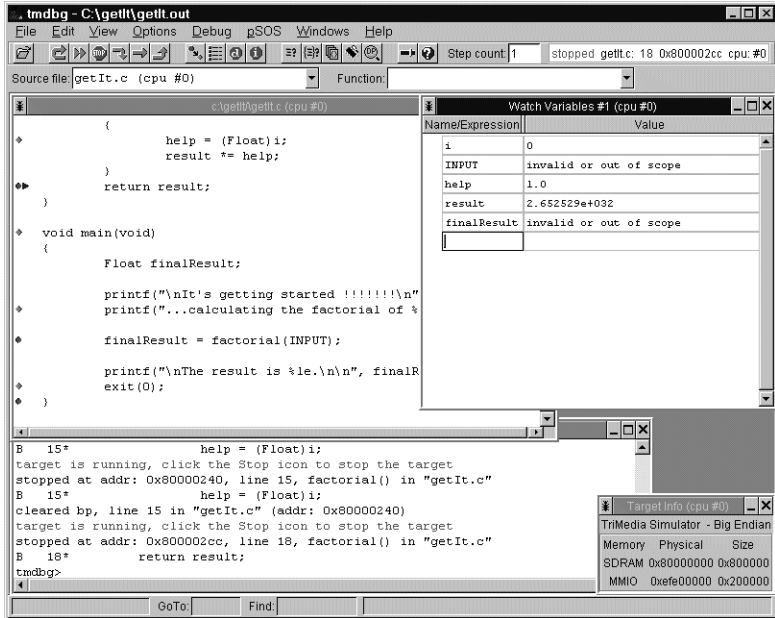
5. Enable the breakpoints in lines 15, 18, 28, 31 by clicking the green “diamonds” at the beginning of these lines.
6. Choose Watch Variables from the View menu.
7. Type the names of the following variables in the Name/Expression column and press Enter after each entry:
 - `i`
 - `input`
 - `help`
 - `result`
 - `finalResult`



8. Click the Continue button to start debugging.
The program stops at line 28 containing the first enabled breakpoint.
9. Click Continue to step the debugger into the **factorial** function.
10. Refresh the Watch Variables window by pressing Return after every variable.
Note that the Watch Variables window is not refreshed automatically.
11. Disable the breakpoint in line 15 by clicking the red "diamond" at the beginning of the line.
12. Click Continue to leave the loop in the **factorial** function.

13. Refresh the Watch Variables window by pressing Return after every variable.

You should see the following:



14. Click Continue.

All the variables of the **factorial** function are now out of scope.

15. Click Continue.

16. Refresh the Watch Variables window by pressing Return after every variable.

You should see **finalResult = 2.652529e+32**.

17. Choose 'Unload from Target' from the File menu.

This step is necessary if you want to start a new debugging session.

There is also a command line version available (**tmddebug**). For more information, refer to Book 4, *Software Tools*, Part C.

Where to Go from Here

- To learn how to improve the performance and efficiency of your code, refer to *Book 2, the Cookbook*, especially Part D.
- To learn how to write real-time audio and video applications, refer to *Book 3, Software Architecture* for an introduction. For more detailed information, refer to *Book 6, Audio Support Libraries* and *Book 7, Video Support Libraries*.
- To learn by example, refer to sample programs that come with your TriMedia SDE. Some of these programs are covered in Chapter 3, *Running Demonstration Programs*, in this book.

Chapter 5

Migration Issues

Topic	Page
Application Libraries	56
Device Libraries and Initialization	59
tmsim—The Simulator	60
tmman—Execution Hosts	60
TSSA	62
tmcc/tmccom—C/C++ Compiler	62
C++ Support	65
Standard C Library	66
tmdbg—Debugger	66
tmprof—Profiler	67
tmsched—Instruction Scheduler	68
Documentation	68
New Components Available as of SDE v2.0	68
Migrating from SDE Version 2.0 to SDE Version 2.1	70

Although this chapter is primarily intended to aid users migrating from the SDE v1.1 to SDE v2.1, it also includes a short section on migrating from SDE v2.0 to SDE v2.1. This chapter can also be used as a short overview of the differences between the version 1 and version 2 products. Refer also to *Important Points to Remember* in Chapter 6 for more important information.

Application Libraries

Audio Renderer (ArendAO)

The audio renderer ArendAO now includes expanded facilities for timestamp-based audio/video synchronization. In short, the operation of the sync mechanism has been made more visible through the use of the callback function.

For more information, please review the following chapter:

[Chapter 6, *Audio Renderer \(ArendAO\) API*](#), of Book 6, *Audio Support Libraries*, Part A.

Audio Digitizer (AdigAI)

There is no change to the AdigAI interface, but the way data from the second output is cached has been corrected. It was possible in previous releases to get garbage data from the second output.

Video In/Out

- The interfaces for the board support packages for the VIconfig and VOconfig initialization functions have changed. They had two parameters, now they each have a structure of parameters. This change allows support of multiple instances of VI on new TriMedia hardware. We have moved these structures from `tmBoard.h` to two separate files, `tmVIboard.h` and `tmVOboard.h`.
- The clock master/slave settings for VI are moved to the board support package, and specifically to the `saa7111.c` file.
- The clock master/slave, frequency, and PLL settings for VO are moved to the individual board description files. Their settings are initialized after decoder initialization.

For more information, please review the following chapter:

[Chapter 1, *Video In/Out \(vi/vo\) API*](#), of Book 7, *Video Support Libraries*, Part A.

Multiple Hardware Units

Support for multiple hardware units has been added for most peripherals (A0, AI, VO, VI, TP, SPDO). Therefore, the APIs have been extended with the two new functions:

```
xxGetCapabilitiesM(
    pxxCapabilities_t  capabilities,
    unitSelect_t      unit
)
```

and

```
xxOpenM(
    Int          *instance,
    unitSelect_t unit
)
```

The old `open` and `getCapabilities` functions use `unit0` by default.

For more information, please review the following chapters:

[Chapter 3, *Audio Device Library*](#), of Book 6, *Audio Support Libraries*.

[Chapter 1, *Video In/Out \(vi/vo\) API*](#), of Book 7, *Video Support Libraries*.

Video Digitizer (VdigVI)

API Changes

- New instance setup flags, `fieldBased` and `Interlaced`, have been added.
- A new 'pointer to video format' structure describes the desired output format of the digitizer.
- The instance configuration function allows changing the acquisition offset on-the-fly.

Other Changes

- The video digitizer supports field bases sending captured data.
- The video digitizer supports two different data organizations in the capture buffer (interlaced, plain).
- The video digitizer uses the progress function to report lost fields and frames.
- The video digitizer uses the error function to report certain errors in the VI ISR.
- The video digitizer has better checking of supplied packets and pointers.
- The video digitizer now supports sending decoder configuration commands to the device library.

For more information, please review the following chapter:

[Chapter 3, *Video Digitizer \(VdigVI\) API*](#), of Book 7, *Video Support Libraries*.

Video Renderer (VrendVO)

- The user can now determine the setting of the HBE, Underrun and yThreshold interrupts using new fields in the **VrendVOInstanceSetup** structure. The value of **yThreshold** can be changed on-the-fly by using **tmaVrendVOInstanceConfig**.
- The definition of the **progressFlags** used by the video renderer has been changed for ease of use. The application is now able to select which progress must be reported by specifying the **progressReportFlags** field in the default instance setup.
- New video formats are defined in **tmVideoFlags_t** in the **tmAvFormats.h** header file. These new formats are: **vdffInterlaced**, **vdffFieldInFrame**, **vdffFieldInField**, and **vdffProgressive**.
- The **tmAVHeaderFlags_t** in **tmAvFormats.h** also contains a new field (**avhField2**) which allows handling of video packets on a field basis (not just a frame basis).
- Additional definitions for progressive mode have been added to **tmaVrendVO.h**.
- The **tmaVrendVOReceiverFormat** function now allows the application to install a new format. If the change in the format is related to the image description, it will be done on-the-fly. If the change is related to the video standard, **tmaVrendVOReceiverFormat** stops the video render, installs the new format, and restarts the video render. Incoming packets with the previous format will be ignored and returned to the application. There is no need to call **tmaVrendVOStop** and **VrendVOStart** to install the format. A new format can be installed by another component, via the **defaultProgress** function, by using some specific flag.
- **VrendMpeg** has been merged with the existing **VrendVO** to allow the video renderer to handle MPEG packets, and perform 3:2 pulldown if required.
- **VrendVO** now supports sub-picture decoding from SVCD.

For more information please review the following chapters:

[Chapter 4, *Video Renderer \(VrendVO\) API*](#), of Book 7, *Video Support Libraries*.

Clock

The clock module can now generate periodic alarms. For TSA compliance, the function **tsaClockGetInstanceSetup** has been added, allowing users to find the clock frequency. A boolean variable **periodic** has been added to **tsaClockSetAlarm**. If true, a periodic alarm is set up; if false, a regular alarm is installed.

For more information, please review the following chapter:

[Chapter 4, *Clock Support API*](#), of Book 5, *System Utilities*.

icptest

The **icptest** example program no longer supports default values for the PCI video card parameters. The user must supply the PCI video display address, the display stride, and

the number of bytes per pixel explicitly using the command arguments `-d`, `-s` and `-b` respectively. The `-h` option displays the program parameter requirements.

Examples

The example `extsaClock` has been renamed `exClock`.

Device Libraries and Initialization

Registry

The device libraries now include a registry mechanism. Like a Windows registry, this registry is a file system that resides in memory and provides facilities similar to a POSIX environment variable. We use this facility to describe the hardware so that there can be a clean separation between the device libraries and the Board Support Packages.

Component Manager

This is a generic manager that allows you to run software before `main` in a structured manner. By default, the SDE uses the component manager to take care of most of the low-level initialization of TriMedia that takes place before `main`. However, this mechanism is generic enough to allow *any* type of initialization before `main` (flash file system initialization, for example).

In combination with the component manager, the new registry manager allows running a single executable on very different boards or TriMedia chips without recompiling (as long as there is binary compatibility).

Specifying Board Support Packages

Board support packages are provided as object files (`.o`) for each board supported. The `tmconfig` file contains entries `BOARD_LIST_EB` or `BOARD_LIST_EL`. These entries specify a set of board support packages to be linked as a default. You can change the default list by editing `tmconfig` or you can invoke `tmcc` this way:

```
tmcc -board=myboard.o ...
```

`tmcc` will override the specification in `tmconfig`, and instead use the file `myboard.o` (for example) as the BSP. Refer to *Linking a Component Into an Application* in [Chapter 3, TriMedia Component Manager API](#), of Book 5, *System Utilities*, Part A.

Migration Notes

Beginning with SDE v2.0, the run-time system supports a more sophisticated board support mechanism. This includes a component manager and a registry, and it supports more devices than the limited set described in the 1.1 BSP. Because of this change, board support packages will have to be updated to work with 2.0.

More information can be found in several places:

Chapter 5, Device Libraries, of Book 3, *Software Architecture*, Part A.

Chapter 19, TMBoard API, of Book 5, *System Utilities*, Part C.

Chapter 2, TriMedia Registry Manager API, of Book 5, *System Utilities*, Part A.

Chapter 3, TriMedia Component Manager API, of Book 5, *System Utilities*, Part A.

In summary, the 1.1 BSP mechanism used a set of static structures to describe the board. The address of these structures was made public as an entry in the `board_config_array`. The SDE v2.1 BSP uses very similar structures, but now the addresses are stored in, and retrieved from, the registry. Refer to the IREF BSP (and that of other boards) whose source is provided in the `examples/boards` directory.

tmsim—The Simulator

Benchmarking

The addition of the component manager and registry adds about 20K of code to standard applications. CPU overhead prior to entering main has increased. The increase is significant and depends on the application.

When comparing performance between the SDE v1.1 and SDE v2.1 or SDE v2.0 releases, users should use **tmsim** with the **-ns** (no startup) option. This option discounts time spent in startup routines. The **-ns** option applies to both the **tmsim -statfile** and **-reportfile** options. Execution overhead using **tmsim** is comparable with the **tmcc -ptm** option for **tmprof**.

As of SDE v2.0, the default target for **tmsim** is TM-1100. If you want the TM-1000 behavior, use the **-target tm1000** option on the **tmsim** command.

tmman—Execution Hosts

- The host communication for Win95 executables was changed after the SDE v1.1 final release, so **-host Win95** or **-host WinNT** executables built with the SDE v1.1 toolset will not work as expected with the SDE v2.1 or v2.0 **tmmon/tmgmon/tmrun** tools. Users should recompile old programs using the new toolset. There is no need to recompile between SDE v2.0 and SDE v2.1.

- The version number of TriMedia host execution tools have been increased to 5.2. Older versions of TriMedia executables, including those compiled with SDE v2.0 alpha or SDE v1.1, will not work with these tools. The executable must be relinked with the tools from SDE v2.1 or SDE v2.0.
- In the past, the TriMedia Manager tools would read some of their settings from the registry and some of their settings from the TMMan.ini file. The TriMedia Manager now reads all of its settings from the registry under the key
 HKEY_LOCAL_MACHINE\SOFTWARE\PhilipsSemiconductors\TriMedia\TMMan.
 If this key does not exist on a machine running the TriMedia host execution tools, the user must create it and put in it only those settings to be overridden. Refer to *TriMedia Manager Registry Entries* in Chapter 14 of Book 5, *System Utilities*, Part A.
- Both SDE v2.0 and SDE v2.1 support execution in both little endian (**-el**) mode and big endian (**-eb**) mode by TM hardware on all windows platforms. This choice is controlled by the registry.
- The TriMedia Manager API that was supported under Windows 95 has changed since the 1.1 final release. The new TriMedia Manager API is the same as the 1.1z Windows NT API and is supported on all Windows platforms. To help in porting existing sources from using the old **TMMan** API to the new **TMMan** API, refer to *Porting Guidelines* in Chapter 14 of Book 5, *System Utilities*, Part A.
- Two new APIs have been added to the TriMedia Manager: **tmmanMAPSDRAM** and **tmmanUnMAPSDRAM**. These APIs enable and disable memory mapping.
- The TriMedia Manager API data structure **DSPInfo** has been modified to accommodate information about non-transparent bridge chips that may be sitting between the PCI bus and the TriMedia processor.

For more information, please review the following chapters:

Chapter 14, *TriMedia Manager API for Windows*, of Book 5, *System Utilities*, Part A.

- To run the new host execution tools the old TriMedia device driver has to be removed from the Windows\System directory and the new driver has to be installed. A utility (tmmanins.exe) has been provided to remove the TriMedia device, the old driver and the related driver installation files from the system.

For more information, please review Chapter 6, *Troubleshooting*.

TSSA

In-Place Components

The components that were called “in-place” components in version 1 of the SDE have been discontinued and are no longer supported by Trimedia. As of version 2.0, there was a new version of “in-place” components with the following requirements:

- They receive a packet from `Datain(GetFull)` and pass it directly to `Dataout(PutFull)`.
- They never call `Datain(PutEmpty)` or `Dataout(GetEmpty)` because they are not attached to any empty queue while they are running. In effect, each corresponding input descriptor and output descriptor share one empty queue.
- The queues are arranged to this configuration during start and reattached to their original configuration during stop by the defaults.
- The use of in-place components is transparent to the application.

A new component, `CopyInPlace`, has been added as an example of in-place components. The corresponding example is `exolCopyInPlace`.

These new in-place components were called asynchronous feedback components in SDE v2.0 alpha. Further information regarding in-place components can be found in [Chapter 11, TSSA Design Details](#), of Book 3, *Software Architecture*, Part B.

tmcc/tmccom—C/C++ Compiler

General Information

- As of SDE v2.0, the compiler portion of the product is a completely new compiler and does not behave exactly like the previous compiler. Users migrating to this compiler are encouraged to try out the new optimization levels.
- The compiler driver `tmcc` has been changed to invoke the new C/C++ compiler. This new compiler consists of three phases `tmcfe` (front end), `tmipa` (inter-procedural analysis at optimization levels `-O4` and `-O5` only), and `tmccom` (back end). The extensions used for the names of the intermediate files have also changed.
- The setting of the environment variable `TCS` is now ignored by `tmcc`. As a migration aid, `tmcc` will issue a warning if `TCS` is set and its value is not the location at which the Trimedia SDE is installed.

- The C/C++ compiler now accepts optimization levels **-O0** through **-O5**, as described below.
 - O0**: No optimizations
 - O1**: Local optimizations (per basic block) only, variables on stack.
 - O2**: Local optimizations (per decision tree) only, variables in registers and loop unrolling.
 - O3**: Global optimizations and loop unrolling.
 - O4**: Global optimizations, loop unrolling and interprocedural analysis with limited inlining.
 - O5**: Global optimizations, loop unrolling, interprocedural analysis and more extensive inlining.
- As of V2.1, the new option **-uselongdouble64** has been added to support the emulation of 64-bit floating point (using a library) on TriMedia. This option must be used both when compiling objects and when linking objects that use 64-bit long double floating point emulation.
- The warning level controls that were available in the 1.1 compiler are no longer available as of the 2.0 compiler. This means that some source code that compiled without warnings using the 1.1 compiler's default settings will have warnings when you are using the 2.1 compiler's default settings.
- **tmcc** passes a default optimization level of **-O1** to the C/C++ compiler when you specify **-g** without any specific optimization level. If you specify neither **-g** nor an optimization level, **tmcc** passes optimization level **-O3**.
- When doing profiling and grafting, the options and the source given on the **-p** (profile) compile should be the same as the options and source given on the **-G** (graft) compile. If the source or options have changed, the compiler may misinterpret the dtprof.out file or issue an error message.
- The compiler now specifies the linker optimizations **-bcompact** and **-bremoveunusedcode**. The **-bcompact** option can rearrange data and cause slower performance because of cache hits. If you wish to turn off these options, specify the **-nocompact** option on the **tmcc** command line. If you wish to turn off **-bcompact** but leave **-bremoveunusedcode** in effect, you must specify the **tmcc** options **-nocompact -tmdl -bremoveunusedcode --**.
- The linker option **-bfoldcode** can also shrink the size of resultant modules but it will give incorrect results if an application wants to distinguish between two functions that have identical implementations. If you wish to specify this linker option, you must use the **tmcc** options **-tmdl -bfoldcode --**.
- The C++ compiler now supports `custom_ops` in C++ programs.

- If you are using `std_arg` functions in your source code you must either provide prototypes for these functions or compile with the `-var_arg` option. The `-var_arg` option is intended only for K&R compatibility and should be used as a last resort.

- **tmcc** now handles six chain symbols for initialization purposes.

`__CC_Init_List` is for C++ initialization.

`__custom_boot` allows execution of user initialization code immediately after stack setup and cache flushing (but before host communication and I/O initialization), which provides a point for triggering debugging code such as exception tracing when debugging low-level libraries.

`__custom_driver` allows execution of user initialization code just after I/O initialization and host communication initialization, but before loading the first immediate code segment, which provides a point for installing drivers (e.g., a flash file system driver) required for dynamic loading.

`__custom_start` allows execution of user initialization code just before the call to `main`, which provides a point for creating system objects (e.g. `stdin`, `stdout` and `stderr`).

`__dataprop_list` and `__component_list` are used by the component manager to chain the list of the components and data properties. Your application should not use the symbols `__custom_boot`, `__dataprop_list`, and `__component_list` directly. The component manager addresses the problem of initialization in a reliable and flexible way.

- Versions 1.x of the TriMedia Compilation System supported the `pragma` statement in addition to the `#pragma` directive. Because the `pragma` statement is a non-portable, non-ANSI language construct, SDE v2.0 and SDE v2.1 support only the `#pragma` directive, not the `pragma` statement.

For more information regarding the new compiler and its options and abilities, please review *Introduction to C Language Users Guide* in Book 4, *Software Tools*.

Long Double Floating Point

At release v2.1, the product include a new **tmcc** option `-uselongdouble64` (or `-usel64` for short). Previously, the compiler represented C types **float**, **double** and **long double** using IEEE single precision format (32 bits), and the C/C++ compiler generated TriMedia hardware floating point operations for all floating point arithmetic operations.

With `-uselongdouble64`, the compiler still represents C types **float** and **double** using IEEE single precision format (32 bits), but it represents C type **long double** using IEEE double precision format (64 bits). The compiler still generates TriMedia hardware floating point operations for single and double floating point arithmetic operations, but it uses a software library to perform long double operations. **Long double** operations are more accurate but considerably slower than than the corresponding **float** or **double** operations.

Because the size of a long double object is different with the `-uselongdouble64` option, the SDE v2.1 distribution includes two separate versions of the standard C library and `tmcc` links with the appropriate version. You should not mix object files compiled with and without the `-uselongdouble64` option.

The standard C library normally defines only double versions of the standard `<math.h>` mathematics library functions, such as

```
extern double cos( double d );
```

The `-uselongdouble64` version of the standard C library also defines corresponding long double routines, such as

```
extern long double _ld_cos( long double d );
```

for users who require the additional precision.

See [Chapter 9, *Library Functions*](#), in Book 4, *Software Tools*, Part A, for more detail.

Applications intended for use with the `-uselongdouble64` option must be written carefully to obtain the benefits of greater floating point precision. Note that unsuffixed floating point constants are of type **double**, not **long double**; floating point constants suffixed with **L** (e.g. `1.23456789E2L`) are of type **long double**. In `printf`, long doubles require formats modified with **L** too, e.g., `%Lf`, `%Le`, or `%10.5Lg`.

C++ Support

C++ Compiler

C++ support is now native to the compiler (not a separate preprocessor) and is much closer to the current ANSI C++ definition. Some parts of the full ANSI definition are not yet implemented. If your program will be using templates, exception handling, or RTTI, it is strongly recommended that you read [Chapter 8, *TriMedia C/C++ Languages*](#), of Book 4, *Software Tools*, Part A.

For a complete discussion of the C++ dialects that are supported, as well as the limitations of the current ANSI support, please review [Chapter 8, *TriMedia C/C++ Languages*](#), of Book 4, *Software Tools*, Part A.

C++ Standard Library

The C++ library now conforms to the library specification of the C++ Standard and is now re-entrant.

As of SDE v2.1, certain template functions in `<iostream>` have been moved to the `LibC++.a` library by default. This move has a definite compile-time benefit and only a slight impact on users who use streams or strings with types other than `char`. For more detail, see *Migrating from SDE Version 2.0 to SDE Version 2.1* at the end of this chapter.

Standard C Library

The C library now includes the functions **opendir**, **readdir**, **rewinddir**, and **closedir** which allow TriMedia programs to process directory information on the host system.

Further information on the C library can be found in [Chapter 9, Library Functions](#), of Book 4, *Software Tools*, Part A.

tmdbg—Debugger

With SDE v2.1 comes extended C++ debugging support for templates and inherited classes. This is above and beyond the following support which was added for SDE v2.0.

Debugging Multiprocessor and Multitasking Applications

- **Multithreaded debugging:** By switching to ‘task-mode’ debugging, users can focus on different pSOS control threads and can stop and start these threads within the debugger. Users can also toggle task event notices including creation, deletion, starting, suspending, and resuming.
- **Multiprocessor debugging:** Users have the ability to launch and debug binaries on multiple TriMedia processors from within a single **tmdbg** debug session. Users can perform context switches to focus on different applications (or possibly the same application) on different processors.
- **New commands related to multiprocessor debugging exist:**
 - **procs:** Display information regarding all installed TriMedia cards.
 - **focus *n*:** Switch the focus to processor *n*.
 - **status all:** Display the program status of all downloaded target programs.

For more information, please review the following chapter:

[Chapter 21, Debugging Multiprocessor and Multitasking Applications](#), of Book 4, *Software Tools*, Part C.

Downloading Support

- **Enhanced stripped binary downloading:** **tmdbg** now can download stand-alone applications compiled without **-g** and stripped executables. **tmdbg** can debug any target compiled with **-g** (as long as it is not stripped) and can download any target except **-g** stripped binaries. Win95 and WinNT applications can now be downloaded whether the target is compiled with **-g** or stripped.
- **Downloading memory image files via JTAG:** **tmdbg** now can download stand-alone memory image files (absolute binary files produced with the **tmdl -mi** option). The

debugger has no way of gaining control over a downloaded memory image file; it simply downloads it and then exits.

Enhanced GUI Features

- C++ class browser.
- Better look-and-feel in the pSOS debug options, the files/functions window, and the downloader interface.
- Per-project attributes are restored across multiple debugging sessions. These attributes include (but are not limited to) software and hardware breakpoints, debugging mode, source file search paths, and target/JTAG configuration.
- Ability to save the DP buffer to a file.
- Ability to disable auto-update of any window.
- New Download File Chooser interface for MP debugging.
- New Source File Locator window (for locating source files from other machines or network mounted file systems).
- pSOS object display now has a more user-friendly graphics representation than the previous version had.
- A new facility allows you to choose whether you want stdout and stderr to be redirected to their own windows or to the message window.

tmprof—Profiler

- Cycle counts are now represented in 32 bits.
- The command line parser has been changed to accept options after the file name arguments.
- The profileStart and profileStop functions provide an independent mechanism to start and stop profiling. These are documented in the tmprof API. The combination of the API and the finite window mechanisms is supported, however the use of the API with an infinite window is not guaranteed to work.

For more information please review the following chapter:

Chapter 5, Performance Analysis Overview, of Book 4, *Software Tools*, Part A.

tmsched—Instruction Scheduler

- The instruction scheduler **tmsched** used in previous releases has been replaced by a new scheduler using different scheduling algorithms.
- The new scheduler recognizes an enhanced decision tree format which includes guarded execution, a join operation to merge values, and after lists for gotrees and cgotos. The compiler uses the enhanced decision tree format when the **-if_convert** option is passed to **tmccom**.
- The tool **dot** displays graphs produced by the **tmsched -d** option. It may be found at <http://www.research.att.com/sw/tools/graphviz/>.

Documentation

The documentation has been reorganized into nine books:

1. *Getting Started*.
2. *Cookbook*.
3. *Software Architecture*.
4. *Software Tools*.
5. *System Utilities*.
6. *Audio Support Libraries*.
7. *Video Support Libraries*.
8. *Graphics Libraries*.
9. *Communication Libraries*.

New Components Available as of SDE v2.0

The following section presents some of the new components available as of SDE v2.0. All these new components are also available in SDE v2.1.

New Code Compression Library

The `libz.a` compression library, for TCS, uses the source and interface of the `zlib` general purpose compression library version 1.1.3, July 9th, 1998, copyright (c) 1995-1998 Jean-loup Gailly and Mark Adler. The `zlib` compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data.

Further information on the code compression library can be found in **Chapter 11, General Purpose Compression API**, of Book 5, *System Utilities*, Part A.

New Flash File System Manager

SDE v2.0 provided a new generic flash file system manager (FFS). The FFS is efficient, provides complete file system functionality, and will remain consistent over power failures and flash write errors. The FFS can store one boot image. A flash-based boot procedure is provided.

For efficient flash utilization, we have provided a number of example tools that are based on the public domain compression library. These tools include:

- A tool for compressing a boot image into a smaller, self-unpacking boot image. Compression ratios of about 50% can be achieved, which might result in considerable flash savings for large images.
- A tool for compressing a directory tree into a self-unpacking archive. This tool allows for easy transfer of files to the flash file system.
- A tool for packing a boot image into a self-unpacking flash boot image writer.

As of SDE v2.1, the Flash File System has been upgraded to allow the Board Support Package to be used as a hardware interface to the on-board flash. Users now have the option of using the board support package or a specific flash driver (as in v2.0) as the interface to the on-board flash.

A separate license is required for integration of the flash file system into your final product. Please contact your TriMedia representative for more information.

Further information on the Flash File System Manager can be found in [Chapter 10, *TriMedia Flash File System API*](#), of Book 5, *System Utilities*, Part A.

Memspace library

The SDE has a new memory management library which can coordinate multiple heaps simultaneously. Multiple heaps make memory management easier, because they allow functional organization of memory. Related pieces of memory can be allocated on their own heap and the heap can eventually be deleted as a whole. Multiple heaps can also help control memory fragmentation.

A separate debugging version of this library is available. This debugging version detects memory errors: various internal consistency checks are enabled that activate assertions in common memory error situations. This debugging version can also track an application's memory usage.

Further information on the TriMedia Memory Manager can be found in [Chapter 6, *TriMedia Memory Manager API*](#), of Book 5, *System Utilities*, Part A.

New Serial I/O Host

The SDE now supports a new host type—serial—which is a variant of host type 'nohost'. The host type can do full ANSI file I/O through a serial channel to an I/O server running

on a different computer. The serial channel is to be defined by the user in the form of a serial driver. The SDE provides serial servers for all compilation hosts.

An example of such a serial channel is a UART. After you provide the appropriate driver, a stand-alone TriMedia board can do full (file) I/O, through one of its UARTs, to a server running on, for example, a PC.

The sample program `$TCS/examples/psos/ethernet_host` illustrates a more sophisticated setup. In this program, the serial channel is implemented by a socket connection. The corresponding serial I/O server now can run on any computer and at any location that has a network connection to the standalone board.

Migrating from SDE Version 2.0 to SDE Version 2.1

There are few considerations when migrating from SDE v2.0 to SDE v2.1.

Every attempt was made to make 2.1 backward compatible. Version 2.0 objects/modules can be linked and executed with version 2.1.

tmcc—the Compiler

By customer request, several compiler warnings in SDE v2.0 have been downgraded to remarks in SDE v2.1. These messages include the “variable is declared but not referenced” message, the “variable is set but not used” message, and the “unreachable code” message. If you wish to receive these messages, add the `-w2` option to your compiler option list.

SDE v2.1 also has some compile-time improvements for both C and C++ code. Specific attention was given to C++ programs that use the `iostream` header file.

The new `-uselongdouble64` option has been added to the compiler to allow some support for 64-bit floating point. If you choose this option (whether you are writing new code, migrating existing TriMedia code, or migrating existing non-TriMedia code), read these brief porting guidelines.

1. If you want a floating point constant to be a 64-bit floating point constant, it must be suffixed with `L`. For example, `3.1415926535` is a floating point number and is represented in floating point as a 32-bit number and lose much precision. But `3.1415926535L` is represented as a 64-bit floating point number when you use the `-uselongdouble64` option and has greater precision.
2. Change all long double `printf` formatting directives from `%f` to `%Lf`. The `L` tells `printf` that the directive is for a long double number. Similar changes apply also to the `%e` and `%g` `printf` directives.

3. 3. Add the following macro text to your source code after `<math.h>` is included.

```
#if defined(__LDBL_LIBC__)
#define acos      _ld_acos
#define asin      _ld_asin
#define atan      _ld_atan
#define atan2     _ld_atan2
#define ceil      _ld_ceil
#define cos       _ld_cos
#define cosh      _ld_cosh
#define exp       _ld_exp
#define fabs      _ld_fabs
#define floor     _ld_floor
#define fmod      _ld_fmod
#define frexp     _ld_frexp
#define ldexp     _ld_ldexp
#define log       _ld_log
#define log10     _ld_log10
#define modf      _ld_modf
#define pow       _ld_pow
#define sin       _ld_sin
#define sinh      _ld_sinh
#define sqrt      _ld_sqrt
#define tan       _ld_tan
#define tanh      _ld_tanh
#define _xpoly    _ld__xpoly
#endif
```

C++ Support

As of SDE v2.1, certain common template functions in `<iostream>` and `<string>` have moved to the `LibC++.a` library. This gives an enormous improvement in compilation time for programs that use `<iostream>`, while only minimally affecting performance. This move has a slight impact for users that use streams or strings with types other than `char`. For these users, there are two compile-time defines that allow full template instantiation in the STL. If you need wider strings, compile with `-DTCS_FULL_STRING`. If you need wider streams, use the `-DTCS_FILL_Iostream` compile-time option.

SDE v2.1 comes with a separate C++ demangler (`tmc++filt`) that allows you to find the real names of functions and variables that appear in messages and output from the compiler.

tmdbg—Debugger

Debugger support is the only area where modules are not completely backward compatible. If you are debugging C++ programs, you will need to recompile with the SDE 2.1 compiler to take advantage of the changes to the stabs interface that enable greater C++ debugging support. The debugger has also been updated to provide better support for pSOS debugging. As part of the update process, a small portion of the debugger/pSOS interface was changed. This interface is not backward compatible. If you are debugging pSOS applications you must either use the pSOS and debug monitor that come with SDE v2.1 or the pSOS and debug monitor that comes with SDE v2.0.

Do not use the SDE v2.1 version of pSOS or debug monitor with the SDE v2.0 version of PSOS or debug monitor.

tmman—Execution Hosts

The **tmman** interface and services are completely compatible with SDE v2.0. A program compiled with SDE v2.0 final can be run with SDE v2.1's **tmman**.

Device Libraries

There is a new device library to support audio (stereo, SAP) demodulation for NTSC TV. Currently, we can only do NTSC TV with modified NIM boards.

There is also a new board support package (BSP) for the DTV Ref5 (Saturn) board which is the reference board for NX2600.

There is a new field in **tsaTvDigDemCapabilities_t** and **boardTvDigDemConfig_t**:

```
unitSelect_t tpUnit;
```

to provide the unit number of the TP/VI unit to which the digital demodulator is connected.

In the types **boardTvTunerConfig_t** and **tsaTvTunerCapabilities_t**, there is a new field:

```
tsaTvTunerConnections_t connections;
```

which replaces **unitConnectedTo**, which was not used (and was also not really usable). Connections contains all the unit numbers of connected modules (audio demod, digital demod, vi unit...):

```
typedef struct {
    unitSelect_t      digDemUnit;
    unitSelect_t      audDemUnit;
    unitSelect_t      viUnit;
    tmVideoAnalogAdapter_t viAdapterType;
    UInt              viAdapterInstance;
}tsaTvTunerConnections_t, *ptsatvTunerConnections_t;
```

Application Libraries

There are no migration concerns with the application libraries, but there are some interesting changes:

- The TSSA examples, especially **GetInstanceSetup**, have been cleaned. This added a function to the default function table.
- VrendVO now supports sub picture decoding from SVCD.
- VdigVI now supports sending decoder configuration commands to the device library. Without exporting the whole tmVI API, it is still accessible through the TSSA component.

Chapter 6

Troubleshooting

Topic	Page
Important Points to Remember	74
Troubleshooting Common Problems Under Windows	80
Frequently Asked Questions	84

This chapter helps you in troubleshooting the problems that you may encounter while using the TriMedia SDE. It also contains answers to commonly asked questions.

Important Points to Remember

This section describes some things you should think about before:

- migrating from the SDE v1.1 to SDE v2.1 or 2.0.
- writing new code for SDE v2.1.
- setting up your hardware environment.
- using the new tools or demos.

The information presented here is arranged by component.

Application libraries

These example programs were not written to run in big-endian mode: exalVtransICP, exolVtransICP, exolAmixSimple, and exolArendAO.

When the video digitizer is working in SIF mode (Halfres), and StartX is enabled, and the value of StartX is less than 136, it works well. If the value of StartX is greater than 136, the captured image is garbage. You can see this effect using exolVrendVO.

Device Libraries and Board Support Packages

The default **tmconfig** file is set to link in a large number of Board Support Packages, so that the generated program can run on a wide variety of boards without relinking. If you are concerned about module size or wish to support only a small number of boards, you should modify the **BOARD_LIST_EB** and **BOARD_LIST_EL** lines of the **tmconfig** file so that they list only the boards you are supporting.

tmcc/tmccom—The C/C++ Compiler

The C/C++ compiler is a new one. It has strengths and weaknesses that differ from the SDE v1.1 compiler. Generally, it is much more aggressive in looking for optimization opportunities. Because of the more aggressive optimizations, small code deficiencies that would have been acceptable with SDE v1.1 will result in incorrect code with SDE v2.1.

The most common problem anticipated is improper use of the **volatile** keyword. There are two main considerations. First, the **volatile** keyword should be used in any case where the value will be updated by other sources (processes, programs, or threads). Second, if the variable in question is a pointer to volatile storage, ensure that you have defined it as such. A common mistake is to mark the pointer as **volatile**, not the storage to which it points.

Because this is a new compiler, it does not generate exactly the same dtrees as the 1.1z compiler did. If you were relying on specific timings or dtrees, you will need to re-evaluate your code.

The Trimedia hardware accesses types only on their natural alignment. That means if an integer load is used, the data should be 4-byte aligned. This can cause trouble in cases where a `char*` array is cast to an `int*` pointer and then dereferenced. In order to make this work you should ensure proper alignment of the `char*` array by dynamically allocating it (word aligned by default) or if it is a static or external array using the `-Xalign` option or the `TCS_align` pragma to force word alignment.

If you are using profiling (`-p`) and grafting (`-G`) in your compilation process, please note that the options and the source given on the profiling compilation must match the options and the source given on the grafting run (with the exception of the `-p` or `-G` options). If the options are different or the source is different, the compiler will not correctly graft the code and a compiler error could be issued.

`tmcc` now turns on the `-bremoveunusedcode` and `-bcompact` linker options by default. These options are generally advantageous, however, in some cases, `-bcompact` can cause a performance degradation because of a cache hit. To turn off both of these options with `tmcc`, you can use the `-nocompact` option. If you use `-nocompact`, we recommend that you manually turn `-bremoveunusedcode` back on by passing `tmcc` (or `tmCC`) the option string `-tmld -bremoveunusedcode --`.

A linker optimization called `-bfoldcode` can create a significant reduction in module size by removing redundant trees. This option should not be used if you have code that depends on functions with different names having different addresses. If you choose to use this function, you can turn it on by passing the following string to `tmcc` (or `tmCC`) when you link: `-tmld -bfoldcode --`.

With the v2.1 release, the compiler has added support for 64-bit emulated long double floating point. If you plan to develop new code using this functionality or if you are porting code from an architecture that currently uses 64-bit doubles, please read the compiler section of *Migrating from SDE Version 2.0 to SDE Version 2.1* in Chapter 5.

The `tmcc` driver passes the `-generate_data_units` flag to the compiler except when the `-t` compiler option is used. If you wish to generate a `.t` file with `-generate_data_units`, you should compile with the options: `-t -tmccom -generate_data_units --`.

The warning level controls that were available in the 1.1 compiler are no longer available in the 2.x compilers. This means that some source code that compiled without warnings using the 1.1 compiler's default settings will have warnings using the 2.0 and 2.1 compilers. Work has been done to reduce the number of warnings that are issued by the 2.1 compiler.

PSOS+ and C++

The C++ front end `tmcfe` performs special processing of the function `main` which results in the execution of static constructors before the invocation of `main`. Because the root task of a pSOS+ program is `root` rather than `main`, a pSOS+ root task written in C++ may not compile as expected and may not execute static constructors as expected, at startup.

To make this work as expected, you must add the declaration:

```
extern "C" void _main(void);
```

Add a call to `_main` at the start of `root`. Note that `_main` and `root` must have C linkage. You must make sure that the link command which builds the executable includes the C++ library (by compiling/linking with `tmCC` or `tmccp` rather than `tmcc`).

Interrupt Service Routines

The PCI device library provides a function (`pciMemoryCopy`) which can be used to copy memory between SDRAM and HOST(PC) even with the PCI Memory window disabled. This function uses synchronous DMA calls that depend on the scheduler being active and hence, cannot be called from interrupt handlers.

tmdbg—The Debugger

Task mode cannot be entered unless stopped at the beginning of the application (i.e. before or at the beginning of the `root` function).

The debugger cannot debug linker-optimized code. Such code includes targets compiled with any of the following linker options: `-bcompact`, `-bfoldcode`, or `-bremoveunused-code`. Although `tmcc` does specify `-bcompact` and `-bremoveunusedcode`, by default, they are not specified if the `-g` option is specified.

The debugger does not use source file/target time stamps. If a source file is modified after the target has been built, the debugger does not warn the user that the source file is out of date.

The debugger cannot debug header files. If any executable code is contained in a header file the debugger will be unable to break on those lines.

The debugger cannot be used with the `-ptm` profiling option. Profile-guided compilation (the `-p` option) is possible with debugging.

tmprof—The Profiler

The `tmprof` in SDE v2.1 is completely upward compatible from the `tmprof` in SDE 2.0. However, there were major changes between the `tmprof` in the v2.0 release and previous releases. This can lead to problems in several areas of which you should be aware. It is important that `tmprof` users carefully read the performance analysis overview in Book 4, *Software Tools*, Part A.

It is no longer necessary to do two profiling runs for profiling cache effects. On the negative side, the space required for a `tmprof` buffer entry has been doubled from 16 to 32 bytes. Use the `-profsz` option to increase buffer size, if necessary.

You will observe differences in measurement results between this and previous versions of `tmprof`. Such differences arise because measurement is necessarily intrusive and the

routines have been changed to add greater flexibility. The main difference is that using the **-ptm** option introduces more cache overhead than before. When using the **tmsim -statfile** option, this is not a problem.

In previous releases, the **-ptm** option measured every dtree introducing a significant performance overhead (160 percent). This overhead is now under the control of the user. Refer to the **tmprof** runtime options (**-ptm**) section. The statistical significance is reduced also, however, so let the user beware.

tmprof can now be used for profiling standalone applications. This is because of both reduced overhead and the existence of a standalone profiling API. The API has been simplified and documented. On the negative side, user-visible changes were made. Migration should be straightforward and the information necessary is present in the end of the performance analysis chapter of Book 4.

The **-ptm** option cannot be used with programs compiled with **-g**. Also, the option cannot be used with the pSOS debug monitor (psosmon.o). Link errors would result.

tmsim—The Simulator

tmsim cycle counts do not take memory refresh into account (see the TM-1000 Data Book). **tmsim** uses the underlying operating system to implement system calls. Differences in the implementation of system calls in the underlying operating systems (HP-UX vs. SunOS vs. Win95 vs. WinNT) and differences in the underlying hardware (e.g., block size) can cause small differences in the reported simulated cycles on different execution host platforms.

tmsim uses the floating point of the host machine to simulate TM-1000 floating point operations. In a few special cases, the behavior of floating point operations involving NaNs and denormals under **tmsim** may differ from the behavior on TM-1000 hardware or on **tmsim** on a different host platform.

tmsim simulation of JTAG, PCI, SSI and VLD peripherals requires the **-mm** option because the peripherals will not work correctly with the **-nomm** option. In general simulation of all peripheral blocks requires the **-mm** option. The **-mm** option is on by default.

tmsim simulation of VLD differs slightly from VLD hardware operation. Several counters (including, but not limited to, the DMA output counters) count differently in TM hardware operation and **tmsim** simulation.

tmman—The Execution Hosts

On systems that have a secondary PCI bus, if the TriMedia Device and a PCI VGA card co-exist on the same (non-primary) bus, then data read and written by the TriMedia device from main host memory may be corrupted. This corruption occurs because of a bug in the PCI bridge chip that causes memory reads/writes to addresses whose last 10 bits are the same as VGA ISA I/O Port addresses (0x3B0–0x3BB) to be blocked from going to main memory. To work around this problem, place the Trimedia device and the VGA

card on different busses or, if they must be on the same bus, put both cards on the primary PCI bus (bus #0).

The C runtime server is the host component that handles POSIX level 2 calls made from the TriMedia processor in a hosted environment. The C runtime server requires the entire SDRAM window to be mapped at all times to provide its services. If the SDRAM is unmapped through the registry keys (as mentioned in [Chapter 14, *TriMedia Manager API for Windows*](#), of Book 5, *System Utilities*, Part A). The C run time server stops working.

If the WDM driver (tmman.sys) is installed on a system running Windows98, and if the file tmman.vxd is present in the Windows\system directory, then running any host tools will cause a PC crash. To work around this problem, delete the tmman.vxd file from the Windows\System directory or rename it.

Currently, the TriMedia device drivers support up to 32 TriMedia devices in a system under Windows platforms. However, because of Windows virtual memory restrictions, SDRAM Mapping might need to be disabled to ensure that all the devices can be handled by the driver.

Configuration settings for the ref3 card have to be specified in the Windows registry for SystemBaseAddress (default 0x10000000), SDRAMBaseAddress (default 0x00000000), and MMIOBaseAddress (default 0xEF000000). They must be defined as **DWORD** values under

```
HKEY_LOCAL_MACHINE\SOFTWARE\PhilipsSemiconductors\TriMedia\TMMan\  
DeviceX
```

where X is 0,1,2, ... depending on the order of card detection. For example, in the case of a system with a single ref3 board, the values must be defined under

```
HKEY_LOCAL_MACHINE\SOFTWARE\PhilipsSemiconductors\TriMedia\TMMan\  
Device0
```

If these registry values are not specified, the default value will be used.

The **TMMan** host driver performs writes to the INT_CTL pins to acknowledge the level-triggered interrupt that TriMedia generates via PCI INT#A. Having the host and the target write to the same register causes contention problems. To circumvent this problem, the host and the target uses a simple handshaking protocol to perform atomic updates of the INT_CTL register.

If any other component running on the TriMedia processor needs to write to the INT_CTL register, it should use a similar handshaking protocol to guarantee that writes are atomic.

The following code runs on the TriMedia processor and shows how to use two semaphores to perform a safe update of the INT_CTL registers in a hosted mode.

The variables `Host2TargetIntPtr` and `Target2HostIntPtr` are initialized by the `TMMAN` initialization code.

```
extern UInt32* Host2TargetIntPtr;
extern UInt32* Target2HostIntPtr;

while ( 1 ) {
    UInt32 Data;

    /* is the lock already claimed */
    pciMemoryReadUInt32( *Host2TargetIntPtr, &Data );

    if( Data==False ) {
        /* claim the lock */
        pciMemoryWriteUInt32( *Target2HostIntPtr, True );

        /* has the host claimed in the mean time */
        pciMemoryReadUInt32( *Host2TargetIntPtr, &Data );

        if( Data==True ) {
            /* yes so we release it */
            pciMemoryWriteUInt32( *Target2HostIntPtr, False );
        } else {
            /* perform update of INT_CTL here */
            MMIO(INT_CTL) = 0x0;
            /* release the lock and get out */
            pciMemoryWriteUInt32( *Target2HostIntPtr, False );
            break;
        }
    }
}
```

Troubleshooting Common Problems Under Windows

This chapter helps you troubleshoot common problems that users encounter while using the TriMedia SDE on Windows.

Table 10 Causes and Solution for Common TriMedia SDE Problems

Problem	Cause	Solution
The error message "Cannot connect to Kernel Mode Driver" occurs when running TMGMon, TMRun, or TMMon.	No device driver has been installed for the TriMedia device or an old and incompatible version of the driver is currently installed.	Install the latest device driver via the appropriate .inf file. If the previous version of the driver is installed, it might be necessary to remove the TriMedia device and the existing device drivers.
The error message "TMRun: ERROR: Cannot Initialize C Run Time Server: CRT I/O calls will not work" is printed in the TMRun Window.	TMRun uses the C Run Time server TMCRT.dll. Only one instance of TMCRT.dll can be running for a given TriMedia processor at any time. If another instance is already running, TMRun generates the error message on the left.	Close all other instance of TMRun.exe that might be running for the given TriMedia processor.
PC freezes upon program execution and requires a power cycle.	The TriMedia compilation system generates executable code that supports speculative loading. Speculative loads can generate accesses in PCI address space. These accesses may result in read operations that are not claimed by any PCI devices. On some Pentium II chipsets, this causes a bus lockup.	The Windows registry should contain the following key: HKLM\SOFTWARE\PhilipsSemiconductors\TriMedia\TMMan If the above key does not exist, create it using regedit.exe and set the SpeculativeLoadFix value to 0x01.

Table 10 Causes and Solution for Common TriMedia SDE Problems (Continued)

Problem	Cause	Solution
<p>Multiple boards are installed in a system but not all of them show up in TMGMon or TMMon.</p>	<p>(1) The system has run out of resources and some TriMedia devices have been assigned resources that are conflicting with other devices in the system.</p> <p>(2) Resources (memory in particular) that have been assigned to a TriMedia device couldn't be mapped into the operating system address space.</p>	<p>The following are possible solutions:</p> <p>The Windows NT registry contains the following key (Windows NT/2000 only): HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management</p> <p>Set the value of System Pages within this key to 0x40000 and reboot the machine. TMMon should be able to map the memory of all the installed TriMedia devices into OS address space. If TMGMon/TMMon still cannot find all the devices, try the next solution.</p> <p>The Windows registry should contain the following key (Windows 95/98/NT/CE): HKLM\SOFTWARE\PhilipsSemiconductors\TriMedia\TMMon</p> <p>If the above key does not exist, create it using regedit.exe and set the MapSDRAM value within this key to 0x0 and reboot the machine. TMMon should be able to detect and enable all the boards as it will not attempt to map the SDRAM memory window of any of the boards. IMPORTANT: C Run Time I/O calls (fopen, fread, fwrite, fclose, printf, scanf) do <i>not</i> work any more with this fix.</p>

Table 10 Causes and Solution for Common TriMedia SDE Problems (Continued)

Problem	Cause	Solution
The error message "Appshell.out : loading of 'xxxxxxx.app' failed with status 1 (DynLoad_FileNotFound)" is printed when running dynamically linked applications with TMGMon, TMMon, or TMRun.	TriMedia software architecture supports running applications that use dynamically linked libraries (DLLs). The search path for DLLs is defined in the registry (DLL-Path). If all the DLLs required by the application are not found in specified search paths, the loading of the dynamic applications fails and the error message on the left is printed.	Make sure that all the DLLs required by the application are in correct search paths.
The TriMedia IREF Board shows up in Device Manager, but it has a yellow exclamation mark in front of it. TMGMon and TMMon do not find any TriMedia device or cannot execute TriMedia programs on the TriMedia device.	Resource conflict.	Microsoft support has knowledge base article (ID Q125174) that gives detailed description and solution for this problem. This can be accessed on-line at support.microsoft.com/support/kb/articles/q125/1/74.asp .
The error message "The TMCRT.DLL file is linked to missing export TMMAN32.DLL:tmmanMessageCreate" occurs when running TMMon, TMRun, or TMMPRun.	Old drivers (vtmman.vxd and tmman32.dll) are installed and you try to run the new TMMon executables under Windows 95/98 platforms.	Remove the existing TMMon device drivers (see 84), including tmman32.dll, and install the newer drivers.

Table 10 Causes and Solution for Common TriMedia SDE Problems (Continued)

Problem	Cause	Solution
The error message "The TMGMon.exe file is linked to missing export TMMAN32.DLL:tmman-DSPGetInfo" pops up when running TMGMon.	Old drivers (vtmman.vxd and tmman32.dll) are installed and you try to run the new version of TMGMon executables under Windows 95/98 platforms.	Remove the existing TMMAN device drivers (see 84), including tmman32.dll, and install the newer drivers.
The error message "TMGMon Error :Loading Executable [*.out] : FAIL[e] [statusExecutableFileWorngEndianness]" pops up when trying to download a TriMedia executable using TMMGMon.	The executable file that you are trying to download is built with a different endianness than the system is configured for currently via registry (DefaultEndianness).	There are two ways to fix this problem: (1) Compile the file again with a different endianness. (2) Modify the registry to set the correct endianness that matches the endianness of the executable file.
Removing the TriMedia Device via Control Panel / Device Manager causes PC to Freeze	This happens due to a bug in the old (1.1 Final) TriMedia Device Driver.	To workaround this problem: Delete or rename the file vtmman.vxd in the Windows\System directory and reboot the machine. Then remove the TriMedia Device via the Control Panel / Device Manager. Since the driver is not loaded after the reboot, the PC will not freeze.

Frequently Asked Questions

Table 11 lists frequently asked questions and their answers.

Table 11 Frequently Asked Questions

Question	Answer
How can I automatically remove the existing TriMedia device and device drivers under Win 95 or Win 98?	The tmmanins.exe utility can be used to achieve this. Tmmanins.exe is a dialog-based Windows application that removes the TriMedia device, TriMedia device drivers and the relevant .INF files from the system. When the user reboots the system, after running tmmanins.exe, Windows prompts the user for .INF file for the TriMedia device. tmmanins.exe runs under Windows 95 and Windows 98 only, it does not run under Windows NT 4.0 or Windows 2000.
How to remove an existing (installed) TriMedia device driver under Win NT 4.0?	Windows NT 4.0 does not support Plug and Play. All device drivers under Windows NT 4.0 have to be installed and removed manually. The driver.exe utility can be used to achieve this. To remove the TriMedia device driver the user needs to take the following steps. (1) Log on to Windows NT machine with administrator privileges. (2) Start a command prompt and run driver -or -ntmman.
How can I manually remove existing TriMedia device and device driver under Win 95, Win 98, or Win 2000?	<p>Removing the driver (tmamn.sys) from the system (Windows 98 and Windows 2000).</p> (1) Click Start, point to Programs, and then click Windows Explorer. (2) In Explorer, go to C:\Windows\System32\Drivers directory. (3) Look for tmman.sys. Click on this file to highlight it, Right Click and then Click Delete. <p>Removing the driver (tmman.vxd) from the system (Windows 95 and Windows 98).</p> (1) Click Start, point to Programs, and then click Windows Explorer. (2) In Explorer, go to C:\Windows\System directory. (3) Look for tmman.vxd. Click on this file to highlight it, Right Click and then Click Delete.

Table 11 Frequently Asked Questions (Continued)

Question	Answer
<p>How can I manually remove existing TriMedia device and device driver under Win 95, Win 98, or Win 2000? <i>Continued</i></p>	<p>Removing all the INF files containing the string "tmman", from the system (Win 95). Click Start, point to Programs, and then click Windows Explorer. In Explorer, go to C:\Windows\INF\Other directory. In Explorer, Click on View, Click on (Folder Options / Options), Click on View Tab, (Click on Hidden Files), Click on Show All Files, Click OK button. In Explorer, Click on Tool, Click on Find, Click on Files or Folders, Find dialog box pops up. In Find dialog box, Click on Advanced Tab. In the Containing Text edit box type "tmman" (without the quotes). Click on Find Now. Files containing the string will be displayed. Click on each file to highlight it, Right Click and then Click Delete.</p>
	<p>Removing all the INF files containing the string "tmman", from the system. (Win98 & Win 2000). Click Start, point to Programs, and then click Windows Explorer. In Explorer, go to C:\Windows\INF\Other directory (for Windows 95 / Windows 98) or c:\WINNT\INF directory (for Windows 2000). In Explorer, Click on View, Click on (Folder Options / Options), Click on View Tab, (Click on Hidden Files), Click on Show All Files, Click OK button. In Explorer, Click on Tool, Click on Find, Click on Files or Folders, Find dialog box pops up. In the Containing Text edit box type "tmman" (without the quotes). Click on Find Now. Files containing the string will be displayed. Click on each file to highlight it, Right Click and then Click Delete.</p>

Table 11 Frequently Asked Questions (Continued)

Question	Answer
<p>How can I manually remove existing TriMedia device and device driver under Win 95 /Win 98 /Win 2000? <i>Continued</i></p>	<p>Removing the hardware device instance from the system.</p> <ol style="list-style-type: none"> (1) Right click My Computer. (2) Click Properties. (3) Click the Device Manager tab. (4) Double-click a device type (Other Devices/Sound, Video and Game Controllers). (5) Click the TriMedia Device (PCI Multimedia Device/TriMedia IREF Board). (6) Click Remove. A Confirm Device Removal dialog box appears. (7) Click OK. <p>Note: after these steps when the system is rebooted Windows Plug and Play will re-detect the device and ask for the INF file and the device driver again.</p>
<p>How can I switch between using the old TriMedia SDE 1.1(Final) and the new TriMedia SDE 2.0(Beta)?</p>	<p>Each one of these releases has its own set of drivers and libraries. Many of the old drivers are incompatible with the new ones and vice versa. In order to switch between releases under Windows 95 / 98 the user has to get rid of the existing (compatible with the installed SDE Release) drivers and install new driver (compatible with the SDE Release the user desires to run). This can be achieved automatically by running tmmanins.exe or manually by executing the following steps:</p> <p>Remove the TriMedia Device Driver.</p> <p>Remove the TriMedia Device.</p> <p>Remove the TriMedia Device Driver .INF file.</p> <p>Ensure that files like tmman32.dll, tmcons.exe, tmrun.exe, tmcrt.dll etc., are not in PATH.</p> <p>Reboot the System.</p> <p>Install the Drivers by providing the correct INF file (for the desired version of the SDE Release) when the system prompts that it has detected a new device.</p> <p>After these steps the system should be ready to execute the required TriMedia Binaries.</p>

Table 11 Frequently Asked Questions (Continued)

Question	Answer
<p>How can I determine if the device drivers are installed for the TriMedia device under Win 95/Win 98/Win 2000?</p>	<p>Follow the steps below to determine if proper device drivers are installed in the system that has the TriMedia device installed.</p> <ol style="list-style-type: none"> (1) Right Click My Computer, click Properties, click Device Manager tab. (2) Double-click a Other Devices, to see the devices in that category. If this category is not present then go to step 4. (3) If the PCI Multimedia Device entry is present then there are NO drivers installed for that device. Goto step 6. (4) Double-click on Sound / Video and Game Controllers, to see the devices in that category. (5) If the TriMedia IREF Board entry is present, then device drivers are installed in the system. (6) Click on OK.
<p>What is the difference between the Win98 drivers : tmman.vxd and tmman.sys ?</p>	<p>Win95\TMMMan.vxd is a Win95 virtual device driver that works both under Windows 95 and Windows 98. Win98\TMMMan.sys is WDM compliant kernel mode driver that works under Windows 98 and Windows 2000 (NT5.0), however it does not work under Win95. Both these drivers work under Windows 98 and neither of them work under Windows NT 4.0. Windows NT 4.0 has its own WinNT\TMMMan.sys driver.</p>
<p>How can I install TriMedia device driver under Win NT 4.0?</p>	<p>Windows NT 4.0 does not support Plug and Play. All device drivers under Windows NT 4.0 have to be installed and removed manually. The driver.exe utility can be used to achieve this. To install the TriMedia device driver the user needs to do the following steps.</p> <p>Log on to Windows NT machine with administrator privileges.</p> <p>Start a command prompt and run driver -oi -sa -ntmman.</p>

Table 11 Frequently Asked Questions (Continued)

Question	Answer
How to find the version information for Tri-Media Device Driver Files?	<ol style="list-style-type: none"> (1) Click Start, point to Programs, and then click Windows Explorer. (2) In Explorer RIGHT click on any TriMedia Device Driver file i.e tmman.sys, tmman32.dll, tmmon.exe, tmrun.exe, tmcrct.dll etc. (3) Click on Properties, Properties dialog box pops up. (4) Click on the Version tab (5) File Version field indicates the version of the executable.
What files are required to setup the hosted execution environment under Windows 95, Windows 98, Windows 2000, or Windows NT 4.0?	<p>The following common files (for all the above platforms) are required to setup an environment to run TriMedia executables on a Tri-Media IREF board.</p> <p>TMMon.exe TMGMon.exe TMRun.exe TMMPRun.exe TMMan32.dll TMCRT.dll AuthHost.dll</p> <p>File specific to Windows 95: Win95\TMMan95.inf Win95\TMMan.vxd</p> <p>Files specific to Windows 98: Win98\TMMan98.inf Win98\TMMan.sys</p> <p>Files specific to Windows NT 4.0: WinNT\driver.exe WinNT\TMMan.sys</p> <p>File specific to Windows 2000 Win98\TMManNT.inf Win98\TMMan.sys</p>
Does TMMan support PCI IRQ sharing?	<p>All versions of TMMan support PCI IRQ sharing. If the same IRQ were assigned to multiple TriMedia devices or to other devices in the system, the TMMan drivers would still work correctly. Note that incorrect implementation of PCI IRQ sharing in other device drivers in the system can cause potential problems.</p>