

**PIKA PREMIERE  
WINDOWS NT™  
AND  
WINDOWS 95  
REFERENCE MANUAL**

**VERSION 1.0**

PIKA TECHNOLOGIES INC.  
155 Terrence Matthews Crescent  
Kanata, Ontario K2M 2A8  
Canada  
Tel: 1-613-591-1555  
Fax: 1-613-591-1488

(c) Copyright 1996 PIKA Technologies Inc.

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or in any other form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior written permission of PIKA Technologies Inc.

Revision: June 1996

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 WINDOWS NT .....	1
1.2 WINDOWS 95.....	2
<b>2. SOFTWARE INSTALLATION .....</b>	<b>3</b>
2.1 FILE LIST .....	4
2.2 CONFIGURATION.....	7
<b>3. LOADING PREMIERE DRIVER .....</b>	<b>8</b>
<b>4. CLOSING PREMIERE DRIVER .....</b>	<b>13</b>
<b>5. APPLICATION PROGRAMMING .....</b>	<b>14</b>
5.1 INTRODUCTION.....	14
5.2 APPLICATION DESIGN.....	14
5.3 BUILDING APPLICATIONS .....	15
5.4 DRIVER INTERFACE .....	15
5.5 EVENT NOTIFICATION.....	15
<b>6. NOTES.....</b>	<b>17</b>



# 1. INTRODUCTION

This manual contains information specific to the PIKA Premiere Windows NT and Windows 95 drivers: contents of the distribution diskette; installation, configuration, loading and closing the driver; and information you might need to develop Windows NT and Windows 95 applications. More information is available in the following documents:

- *Premiere Voice Card User's Guide*
- *PIKA Library Reference Manual*
- *PIKA Premiere Family Hardware Manual*
- *PIKA Technical Bulletins*

This document use the term *Win32* to apply to both Windows NT and Windows 95 operating systems.

## 1.1 Windows NT

The Windows NT driver for the PIKA Premiere consists of:

- `ioxdrv.sys` - Native Windows NT kernel mode driver.
- `dgdrv.exe` - Windows NT application running in the background.
- `ntgdll.dll` - Windows NT DLL which exports all PIKA API services.

The DGDRV.EXE is a Windows NT process that runs at higher priority than applications and does most of the system work. The interface to the driver is through NTDGDLL.DLL.

A C interface for all PIKA API services and related data structures is declared in the header files located in the include directory. There is an NTDGDLL.LIB for all PIKA API services exported from NTDGDLL.DLL (see *PIKA Library Reference Manual* for function descriptions).

You can build a non-C language application if that language supports DLL calling conventions and you can create an interface module and data structures similar to the ones declared for C.

## **1.2 Windows 95**

The Windows 95 driver for the PIKA Premiere consists of:

- dgdrv.exe - Windows 95 application running in the background.
- w95dg.dll - Windows 95 DLL which exports all PIKA API services.

The DGDRV.EXE is the Windows 95 process that runs at higher priority than applications and does all the device driver work. The interface to the driver is through W95DG.DLL.

## 2. SOFTWARE INSTALLATION

The distribution diskette contains the following directories:

Diskette #1:

- \ Contains install.bat
- \Bin Contains Premiere executable and configuration files
- \Api Contains API C header files, libraries and source.
- \Utility Contains applications and facilities useful for application development.

Diskette #2:

- \ Contains install.bat
- \Samples Contains some sample applications.

To install the Premiere software, insert Premiere installation diskette #1 in floppy drive A:, and from the DOS prompt type:

```
a:install source_path dest_path
```

The installation program prompts you when it is time to insert the second diskette. The installation creates the same directory structure as on the distribution diskettes in `dest_path` and copies all files to your hard disk. *For Windows NT only:* Then it installs `ioxdrv.sys`. **NOTE:** you must reboot your system to start this kernel mode driver.

After you reboot, change the current directory to `dest_path\bin` and start `pikainst.exe`. This utility locates your Premiere card(s), and will create `pika.cfg`.

Now you are ready to start `dgdrv.exe`. When you see the message "System ready" on the console window, you can start any of your applications.

The driver process displays appropriate message on its console window, whenever an application starts or exits.

## 2.1 File List

### Root directory

install.bat Installation batch file.  
readme.txt Text file that contains a list of new features added with the current version of the driver.

### \Bin

dgdrv.exe Premiere *Win32* driver.  
msvcrt20.dll Microsoft Visual C++ 2.2 DLL. This DLL must be in the current directory or in the PATH when you start DGDRV.EXE.  
pika.bin Data file required to download Premiere firmware. It must be in the current directory when you run DGDRV.EXE.  
pika.cfg Sample configuration file. To create your own configuration file, run PIKAINST.EXE. The PIKA.CFG file contains data about your Premiere card configuration and it must be in the current directory when you run DGDRV.EXE.  
pikainst.exe Premiere hardware configuration utility. It creates PIKA.CFG file.  
packages.dat Sample file to change the default parameters of the Advanced Tone Detector.  
tones.dat Sample file to change the default parameters for tone generation.

#### *Windows NT only:*

ioxdrv.sys Windows NT kernel mode driver.  
regini.exe Microsoft utility for kernel mode driver installation.  
driver.ini Configuration file for IOXDRV.SYS installation.  
ntdgdll.dll Windows NT DLL exporting all PIKA API services.

#### *Windows 95 only:*

w95dg.dll Windows 95 DLL exporting all PIKA API services.

### \Api\Lib

#### *Windows NT:*

ntdgdll.lib Library file to link with NTDGDLL.DLL.

#### *Windows 95:*

w95dgl.lib Library file to link with W95DG.DLL.

### **\Api\Include**

confapi.h	Declaration of the low-level conference API.
pikacnst.h	Events, error codes and data structure definitions.
pikamvip.h	MVIP related API services and data structure definitions.
pikapi.h	PIKA Application interface header file.
osdefs.h	Internal use definitions.
faxapi.h	Low-level, PIKA facsimile API services and data structures.
pikaconf.h	Low-level, PIKA conference API services and data structures.

### **\Api\Source**

evt2str.cpp	Exports function for converting PIKA events code into more meaningful strings.
evt2str.h	Header file for exports function.
initdrv.cpp	A common module for all applications that export initialization functions.
initdrv.h	Header file for initialization module.
pikamsq.cpp	A common applications module that encapsulate inter-process communication for receiving events from the driver.
pikamsq.h	Header file for inter-process communication module.

### **\Utility\Test**

*Windows NT:*

tnt_dg.exe	A utility to access the driver services for the Premiere.
------------	---

*Windows 95:*

t95_dg.exe	A utility to access the driver services for the Premiere.
------------	---

### **\Samples\Conf**

pikaconf.cpp	Sample conference application source code.
pikaconf.exe	Sample conference application executable.
pikaconf.mak	Sample conference application project file for Visual C++ 2.2.

**\Samples\Mvip**

testmvip.cpp Sample MVIP application for dumping the MVIP Switch Block.  
testmvip.exe Sample MVIP application executable.  
testmvip.mak Sample MVIP application project file for Visual C++ 2.2.

**\Samples\Dtmf**

getdtmf.cpp Sample DTMF digit collection application source code.  
getdtmf.exe Sample DTMF digit collection application executable.  
getdtmf.mak Sample DTMF digit collection application project file for Visual C++ 2.2.

## 2.2 Configuration

The Premiere driver (DGDRV.EXE) uses a text file PIKA.CFG to obtain information about your card configuration. PIKA.CFG contains a list of cards and their port addresses and other global or per-card basis information.

To create PIKA.CFG, use the configuration program PIKAINST.

You only need to run PIKAINST when you first install your Premiere cards or whenever you change card IO addresses.

**Warning:** Because PIKAINST scans the entire possible IO address space from 0x200 to 0x3FF and 0x600 to 0x7FF, it may interfere with other cards installed in your system: for example, network cards or other voice cards.

Each Premiere card in your system **MUST** have a different base IO address. *PIKA Premiere Family Hardware Manual* describes how to change a card base IO address.

### 3. LOADING PREMIERE DRIVER

The following files are required to load and initialize the Premiere Windows NT driver:

DGDRV.EXE, PIKAINIT.BIN, PIKA.CFG, NTDGDLL.DLL and IOXDRV.SYS. IOXDRV.SYS must be running.

Move to the directory where you have installed the Premiere driver.

To start the Premiere driver, run DGDRV.EXE. This program initializes the hardware and downloads the firmware. DGDRV must be able to find and load PIKA.CFG and PIKAINIT.BIN. It also looks for two optional files: PACKAGES.DAT and TONES.DAT. These two files contain data to override the default parameters for tone detection and tone generation.

The full syntax for DGDRV is:

```
DGDRV [-C:x:y] [-Fn] [-Nc] [-O[i]:n] [-P] [-?]
```

Where:

**-C:x:y** Change conference parameters:  $x$  is the gain robustness and  $y$  is the long distance robustness. The default value for each is 0. With release 4.5, the conference provides better quality and higher volume for the voice. Gain values of +5dB for the talk gain and +10dB for listen gain can be used without instability problem. Each channel has a programmable input gain pad and a programmable output gain pad. The input gains can be individually set to any level between +6 dB and -40 dB in 0.1 dB steps. The output gains can be individually set to any level between +24 dB and -40 dB in 0.1 dB steps. The maximum value for <talk gain + listen gain> of a channel is selected by the gain robustness parameter. In general, values which are too low can result in an unstable conference and higher values will decrease the signal quality in double talk conditions.

Gain robustness: this parameter takes a value from 0 to 9. A value of 0 is for minimum gain robustness with a typical maximum gain of -3 dB (talk gain + listen gain); and a value of 9 for

maximum gain robustness with typical maximum gain of 22 dB (talk gain + listen gain).

gain robustness	typical maximum gain
0	- 3 dB
1	0 dB
2	3 dB
3	6 dB
4	9 dB
5	12 dB
6	15 dB
7	18 dB
8	20 dB
9	22 dB

Long distance robustness: this parameter takes a value from 0 to 9. A value of 0 is for minimum long distance robustness and a value of 9 is for maximum long distance robustness. The typical maximum distances apply when the telephone company does not provide echo cancellation in the circuit. There is no limit when a connection has echo cancellation.

long distance robustness	typical maximum distance
0	400 km
1	800 km
2	1200 km
3	1600 km
4	2000 km
5	2400 km
6	2800 km
7	3200 km
8	3600 km
9	>4000 km

-Fn Specify the local phone number (n) for fax transmission identification. This option is equivalent to -O:4.

-Nc Set the maximum number of channels (c) the driver controls.

-O[i]:n

Specifies options for run-time control. The following options exist for this release.

0x0001 If set, A-law companding is used as the voice data format (by default, the driver uses mu-law companding).

0x0002 Make available pulse detection (i.e., load the pulse detection firmware on DSP). See *Premiere Voice Card User's Guide*.

0x0004 Load the fax transmitter firmware on the DSP. See *Premiere Voice Card User's Guide*.

0x0008 Disable the DTMF Double Talker detector which attenuates the playback level to increase the cut-through performance of the DTMF detector.

0x0010, 0x0020, 0x0040, and 0x0080

Specify the size of play and record buffers. The unit is the buffer needed for ADPCM four bits per sample, 8K samples per second. There are rules for choosing the play/record buffer size.

- (1) For 8K samples/sec, the size of the buffer (as a multiple of the unit buffer) is:  $N = (\text{bits per sample}) / 4$ .
- (2) To use fast-forward playback, double the size:  $N*2$ .
- (3) For 4K samples/sec, the size of the buffer is half the size needed for 8K samples/sec.

*Example #1:* You want fast forward with a default ADPCM of 8K samples/sec, 4 bits/samples, with fast forward. Size = 2 (set the bit to 0x0020).

*Example #2:* You want A-law without fast-forward capability at 8K samples/sec, 8 bits/samples. Size = 2 (set the bit to 0x0020).

*Example #3:* You want fast-forward with linear mode at 8K samples/sec, 16 bits/samples. Size = 8 (set the bit to 0x0080).

**Notes:**

(1) When the DSP has 32K of memory, the driver selects a buffer size of one and does not load the code for low-bit rate and pitch-corrected (i.e., fast forward) playback because of memory limitations.

(2) When the DSP has 64K of memory, the driver selects a default buffer size of two and does load the code for low-bit rate and pitch-corrected playback. In this case, you may modify the buffer size using the four options described above.

0x0100 There are three types of firmware that can be loaded. For ADPCM, there is a small one which uses only the default record and play rate of 32kbits/sec and does not have pitch-corrected fast forward. There is a large one which supports all the previous features and is limited only by the size of the play and record buffer. And there is a medium one with a fixed bit-rate of 32 Kbps and pitch-corrected playback. By default, the driver uses the small model for configurations with simple ADPCM buffers and the large model where double-buffered ADPCM is necessary. You select the different models with a combination of command-line switches:

-O:0x2000 (see below) — medium

-O:0x0010 — small

-O:0x0100 or -O:0x0020 or -O:0x0040 or

-O:0x0080 — large

0x0200 The conference firmware is not loaded on DSP and the DSP-based conference algorithm is disabled. By default, the driver loads and enables the conference firmware.

0x0400 The caller id detector in the firmware is not loaded on the DSP. By default, the driver loads it.

0x1000 The complete fax firmware is loaded on the DSP. Set this bit in combination with 0x0004. If only 0x0004 is specified, a smaller version of the fax firmware is loaded; this version supports fax transmission only.

0x2000 Select the medium configuration of firmware (32 Kbps and pitch-corrected playback — see option 0x0100).

0x10000 If set, modify the timing of the transfer on the PC bus. This option can fix problems which have shown up on some Pentium PCs. Setting this option indicates a non-standard ISA bus computer; the default within the driver is to assume a standard ISA bus. This option helps to correct problems observed recording voice.

If *i* is specified, the driver assumes that the options apply only to board *i*; otherwise, the options apply to all boards in the system.

-P Load the firmware for pulse detection and, by default, enable pulse detection in the off-hook state. The pulse digits are reported in the same way as DTMF digits. Pulse can be enabled only on the Premiere GT with a cluster of four channels. This option is equivalent to -O:2.

-? Display this help information.

## **4. CLOSING PREMIERE DRIVER**

To close the *Win32 PIKA Driver*, you press <Ctrl-C> when the driver console is the active window, or just close the console window.

Make sure that all applications using the driver are closed before closing the driver.

## 5. APPLICATION PROGRAMMING

### 5.1 Introduction

The PIKA DLL lets you add voice capability to your *Win32* applications quickly and easily.

### 5.2 Application design

*Win32* applications that use the PIKA driver should be structured in the following way:

1. *pika\_register()* should be called. Events cannot be received by the application, and no other functions can be called, until the application is registered. The driver provides each application with a unique identifier, called an *hApp*, upon registration. This *hApp* is required to validate all other function calls, and is used to prevent applications from interfering with each other's lines.
2. Create a mechanism for inter-process communication. Refer to the manual page for *pika\_register()* in *PIKA Library Reference Manual* for more details about the available options for inter-process communication.
3. *pika\_seize()* or *pika\_seize\_any()* should be called to allocate a line or lines for use by the application.
4. *pika\_set\_ch()*, *pika\_set\_cpl()* (if call analysis required) and *pika\_set\_channel\_gain()* should be called for each channel seized to set channel parameters.
5. Start operations on channels as required. Note that *pika\_set\_cpl()* should also be called if reorder tone detection is required in *pika\_play()*, *pika\_record()* and *pika\_get\_dtmf\_string()* functions.
6. Perform actions based on event codes.
7. Before exiting the application, call *pika\_release()* to release any allocated lines and *pika\_deregister()* to stop event notification. Finally, if you used a mailslot for inter-process communication, you close your handle to the mailslot.

Sample applications are provided on the distribution disk. We suggest that you study the techniques used in these applications, especially to initialize and terminate an application, and the method for event notification. Note that these applications are intended to show how to structure an application, and do not provide all the error checking that one would expect in a commercial program.

### 5.3 Building applications

When building applications, link with the library file provided. This library allows the linker to locate the DLL functions and ensures that the library is loaded at run time.

### 5.4 Driver Interface

Applications control the PIKA voice card by invoking DLL functions. Unlike the PIKA DOS library, the PIKA *Win32* driver allows play and record operations to continue without intervention from the application. Applications are automatically notified of events resulting from a driver function.

### 5.5 Event Notification

The inter-process communication between the PIKA driver process and an application is done through Windows message queues, mailslots, or callback functions.

The following list briefly describes how to use Windows message queues for inter-process communication.

1. The application creates a window with the Windows API `CreateWindow()` function.
2. The application sends the handle returned by `CreateWindow()` to the driver with `pika_register()`.
3. The application uses the Windows API `RegisterWindowMessage(PIKA_EVENT_MSG)` to get the identifier of the Windows message reserved for PIKA events.
4. When there is an event from the driver, the application receives a `PIKA_EVENT_MSG` message in its window message queue.

The following list is a brief description of how the mailslot inter-process communication should work. Note, however, that we do not recommend this mechanism for Windows 95: the operating system does not implement overlapped operations on mailslots.

1. The application creates a mailslot using Windows NT API function *CreateMailslot()*.
2. The application calls *pika\_register()* to pass the mailslot name to the driver.
3. The driver uses that name to open a client end of the mailslot.
4. The driver uses this client end of the mailslot to send messages to the applications.
5. The application periodically reads the mailslot and processes the event messages in its state machine implementation.

The structure of these messages is of type `TPikaEvent`, where `TPikaEvent` is defined as follow:

```
typedef union
{
    dword request;
    struct
    {
        byte event;
        byte channel;
        word data;
    } msg;
} TPikaEvent, *PPikaEvent;
```

For information on how to use the `TPikaEvent` structure see “Event List” in *PIKA Library Reference Manual* and the sample source code provided with the PIKA Development Toolkit.

***IMPORTANT NOTE: All PIKA data structures must be byte aligned for correct operation***

## **6. NOTES**

Windows NT is a trademark of Microsoft Corporation.