TERIDIAN SEMICONDUCTOR CORP.

Simplifying System Integration™

73M1866B/73M1966B Infineon TAPI High-Level Driver User Guide

July, 2010
Rev. 2.2
UG_1x66B_021

Teridian Semiconductor Corporation makes no warranty for the use of its products, other than expressly contained in the Company's warranty detailed in the Teridian Semiconductor Corporation standard Terms and Conditions. The company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice and does not make any commitment to update the information contained herein. Accordingly, the reader is cautioned to verify that this document is current by comparing it to the latest version on http://www.teridian.com or by checking with your sales representative.

Teridian Semiconductor Corp., 6440 Oak Canyon, Suite 100, Irvine, CA 92618
TEL (714) 508-8800, FAX (714) 508-8877, http://www.teridian.com

Table of Contents

Figures

Tables

1 Introduction

This document describes the functionalities of the 73M1866B/73M1966B TAPI Driver. This driver software is provided as part of the Infineon BSP package for Danube, Vinax and AR9 platforms. Its design and implementation is tightly integrated with Infineon TAPI architecture, therefore, its sole purpose is to be used in Infineon TAPI environment. In addition to this TAPI driver, the Teredian 73M1866B/73M1966B Reference Driver is also required as for interface to the 73M1x66 chip set.

The 73M1x66B TAPI driver is built as a loadable module. It will be brought into operation by a user application or by an operating system startup script. For Linux, the “insmod” command is used to insert the driver into the kernel. The “insmod” command invokes the *module_init()* macro, which in turn runs the one-time initialization function of the driver.

1.1 Purpose and Scope

IFX TAPI is the API layer used by Infineon. This version of the Teridian TAPI driver is compliant to IFX TAPI Version 3.8.3. This specification is available from Infineon.

The 73M1x66B TAPI Driver provides the necessary system interfaces for the control and management of the 73M1x66B. The driver supports ioctl calls from the application and translates these to and from the device via the Reference Driver layer.

The scope of this document will includes only the TAPI driver’s user interface. Detail of internal driver architecture can be found in the 73M1x66B Reference Driver User Guide document.

Figure 1 illustrates the basic architecture model for the driver. The model is intended to be independent of processor and operating system. Layers above the reference driver address software interfaces which may pre-exist for a given application (e.g. Asterisk®) and the layer below addresses hardware related interfaces between the processor and the 73M1x66B devices.

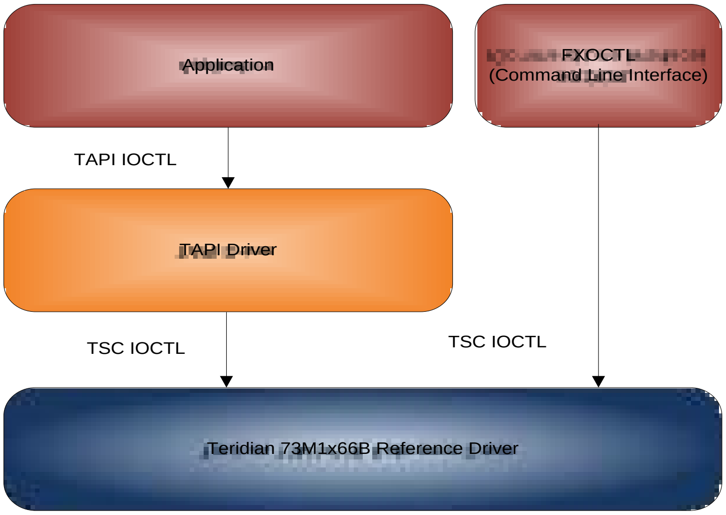

Figure 1: Driver Architecture

1.2 Conventions Used in this Guide

This document uses the following conventions:

- Software code, IOCTL names and data types are presented in Courier font.
- A table with a blue header is a summary table. A table with a gray header is a detail table.

1.3 Acronyms

APOH – Another Phone Off Hook
BSP – Board Support Package
DAA – Data Access Arrangement
FXO – Foreign eXchange Office
IOCTL – I/O Control
NOPOH – No Phone Off Hook
POH – Phone Off Hook

2 Driver Service Interface

The Driver Service provides the link between the FXO device and the user application. First, the driver must be loaded and bonded into the operating system environment before this service can be provided. Access to the driver is done via two file descriptors – the device and channel file descriptors. The device file descriptor provides access to device level management interface while the channel descriptor is used to manage at the channel level interface. The driver supports multiple FXO channels through separated channel descriptors; however, only one device descriptor is used.

The following sections describe how the driver is brought into action based on the operating system environment.

2.1 Linux Operating System

This description is valid for Linux 2.4 and 2.6. The 73M1x66B driver takes the form of a Linux standard character device driver. It is brought into operation by a user application or by Linux startup script using *insmod* command. This command inserts the driver module into the kernel which in turn registers with the kernel using the default major number of 221. Multiple FXO channels are supported via the use of minor number which can varies from 0 to 16. This minor number associated with the device and channel descriptors created using mknod command. The driver expects the minor number 0 to be associated with the device descriptor and the number from 1 to 16 with the channel descriptors. Device major and minor numbers are configurable at build time as described in the Reference Driver document.

The device and channel descriptors must be created in the "/dev" directory at the same time when the driver is insmod into the kernel. The *mknod* command is used to create those descriptors as illustrated below:

```
mknod -m 660 /dev/ter10    c 221 0
mknod -m 660 /dev/ter11    c 221 1
```

In this example above one device descriptor (ter10) is created with major number 221, minor number 0, and one channel descriptor (ter11) is created with major number 221, minor number 1.

Once the driver is installed and the device/channel descriptors are created, the driver service can be accessed via standard C library `open()`, `close()`, and `ioctl()` functions.

The following illustrates how the device and channel are opened, closed, and the ioctl access:

```
devfd  = open("/dev/ter10",O_RDONLY|O_WRONLY);
chanfd = open("/dev/ter11",O_RDONLY|O_WRONLY);
ioctl (devfd, M1966_EVENT_GET, &event_structure);
ioctl (chanfd, M1966_ATH1, NULL);
close (devfd);
close (chanfd);
```

Accessing the driver using ioctl must be done via an opened descriptor. There are two types of ioctl command – the device level commands which can be accessed by an opened device descriptor and channel level commands, which can be accessed using an opened channel descriptor. Section 3 describes the ioctl command.

2.2 Other Operating Systems

To be provided.

3 IFX TAPI IOCTL Commands Description

Once the driver is successfully opened the application can control the operation of the device and the FXO channel. The application in user space communicates with the driver via standard Linux driver interface IOCTL calls. The following sections describe the detail of each IOCTL command.

Some commands pertain to device level configuration such as PCM interface parameters; these must be called using device descriptor, while others are channel level commands and must be called using specific channel descriptor, if more than one channel is active. Table 1 provides a summary of the IOCTLs.

Table 1: Summary of IFX TAPI IOCTLs

IOCTL Name	Description	Descriptor
`IFX_TAPI_CH_INIT`	Initializing FXO channel for operation.	Channel
`IFX_TAPI_PCM_IF_CFG_SET`	Set PCM interface configuration.	Device
`IFX_TAPI_PCM_CFG_SET`	Set PCM configuration.	Channel
`IFX_TAPI_PCM_CFG_GET`	Retrieve PCM configuration.	Channel
`IFX_TAPI_PCM_ACTIVATION_SET`	Activate/Deactivate PCM channel.	Channel
`IFX_TAPI_PCM_ACTIVATION_GET`	Retrieve PCM activation state.	Channel
`IFX_TAPI_EVENT_ENABLE`	Enable FXO event detection.	Channel
`IFX_TAPI_EVENT_DISABLE`	Disable FXO event detection.	Channel
`IFX_TAPI_EVENT_GET`	Retrieve FXO event.	Device
`IFX_TAPI_VERSION_GET`	Retrieve driver version number.	Device
`IFX_TAPI_DEBUG_REPORT_SET`	Set driver debug message trace mask.	Device
`IFX_TAPI_LINE_TYPE_SET`	Set line type – must be FXO only.	Channel
`IFX_TAPI_PHONE_VOLUME_SET`	Set speaker phone/micro phone volume.	Channel
`IFX_TAPI_FXO_HOOK_SET`	Issue on/off hook.	Channel
`IFX_TAPI_FXO_FLASH_SET`	Issue flash hook.	Channel
`IFX_TAPI_FXO_FLASH_CFG_SET`	Configure FXO hook flash parameter.	Channel
`IFX_TAPI_FXO_BAT_ GET`	Retrieve battery status.	Channel
`IFX_TAPI_FXO_APOH_ GET`	Retrieve APOH status.	Channel
`IFX_TAPI_FXO_RING_ GET`	Retrieve ring status.	Channel
`IFX_TAPI_FXO_POLARITY_ GET`	Retrieve line polarity status.	Channel
`IFX_TAPI_LASTERR`	Retrieve driver last error code.	Device
`IFX_TAPI_FXO_LINE_MODE_SET`	Enable/Disable FXO channel.	Channel

3.1 IFX_TAPI_CH_INIT

Description

Perform all 73M1x66 channel initialization. This includes initialize all default registers and country specific threshold parameters.

```
#define  IFX_TAPI_CH_INIT     _IO(IFX_TAPI_IOC_MAGIC, 0x0F)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_CH_INIT,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_CH_INIT`	I/O control identifier for this operation.
`unsigned long`	`param`	The parameter points to a `IFX_TAPI_CH_INIT_t` structure.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed perform channel initialization. IFX_SUCCESS – Successful.

3.2 IFX_TAPI_PCM_IF_CFG_SET

Description

This ioctl configures the PCM interface.

```
#define  IFX_TAPI_PCM_IF_CFG_SET                _IO(IFX_TAPI_IOC_MAGIC, 0x11)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_PCM_IF_CFG_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`dev_fd`	Device descriptor.
`int`	`IFX_TAPI_PCM_IF_CFG_SET`	I/O control identifier for this operation.
`unsigned long`	`param`	The parameter points to a `IFX_TAPI_PCM_IF_CFG_SET_t` structure.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to configure PCM interface. IFX_SUCCESS – Successful.

3.3 IFX_TAPI_PCM_CFG_SET

Description

This ioctl configure the time slot for the PCM channel.

```
#define  IFX_TAPI_PCM_CFG_SET                        _IO(IFX_TAPI_IOC_MAGIC, 0x04)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_PCM_CFG_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_PCM_CFG_SET	I/O control identifier for this operation.
unsigned long	param	The parameter points to a IFX_TAPI_PCM_CFG_t structure.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to configure PCM channel. IFX_SUCCESS – Successful.

3.4 IFX_TAPI_PCM_CFG_GET

Description

This ioctl retrieves the current time slot configuration for the PCM channel.

```
#define  IFX_TAPI_PCM_CFG_GET                        _IO(IFX_TAPI_IOC_MAGIC, 0x05)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_PCM_CFG_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_PCM_CFG_GET	I/O control identifier for this operation.
unsigned long	param	The parameter points to a IFX_TAPI_PCM_CFG_t structure.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to retrieve PCM config. IFX_SUCCESS – Successful.

3.5 IFX_TAPI_PCM_ACTIVATION_SET

Description

This service activates / deactivates the PCM time slots configured for this channel.

```
#define  IFX_TAPI_PCM_ACTIVATION_SET            _IO(IFX_TAPI_IOC_MAGIC, 0x06)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_PCM_ACTIVATION_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_PCM_ACTIVATION_SET	I/O control identifier for this operation.
unsigned long	param	The parameter defines the activation status: 0 deactivate the time slot, 1 activate the time slot.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to perform PCM activation set. IFX_SUCCESS – Successful.

3.6 IFX_TAPI_PCM_ACTIVATION_GET

Description

This service gets the activation status of the PCM time slots configured for this channel.

```
#define  IFX_TAPI_PCM_ACTIVATION_GET            _IO(IFX_TAPI_IOC_MAGIC, 0x07)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_PCM_ACTIVATION_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_PCM_ACTIVATION_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	The parameter points to an integer which returns the status 0: The time slot is deactivate, or 1: The time slot is active.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to retrieve PCM channel activation status. IFX_SUCCESS – Successful.

3.7 IFX_TAPI_EVENT_ENABLE

Description

Enable detection of FXO events.

```
#define IFX_TAPI_EVENT_ENABLE              _IO(IFX_TAPI_IOC_MAGIC, 0xC1)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_EVENT_ENABLE,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_EVENT_ENABLE	I/O control identifier for this operation.
unsigned long	param	N/A.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to enable event detection. IFX_SUCCESS – Successful.

3.8 IFX_TAPI_EVENT_DISABLE

Description

Disable detection of FXO events.

```
#define IFX_TAPI_EVENT_DISABLE              _IO(IFX_TAPI_IOC_MAGIC, 0xC2)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_EVENT_DISABLE,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_EVENT_DISABLE	I/O control identifier for this operation.
unsigned long	param	N/A.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to disable event detection. IFX_SUCCESS – Successful.

3.9 IFX_TAPI_EVENT_GET

Description

Read FXO event from the driver.

```
#define IFX_TAPI_EVENT_GET                    _IO(IFX_TAPI_IOC_MAGIC, 0xC0)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_EVENT_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`dev_fd`	Device descriptor.
`int`	`IFX_TAPI_EVENT_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	Pointer to an `IFX_TAPI_EVENT_t` structure.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to read event. IFX_SUCCESS – Successful.

3.10 IFX_TAPI_VERSION_GET

Description

Retrieves the TAPI Driver version string.

```
#define  IFX_TAPI_VERSION_GET                   _IO(IFX_TAPI_IOC_MAGIC, 0x00)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_VERSION_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`dev_fd`	Device descriptor.
`int`	`IFX_TAPI_EVENT_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	Pointer to version character string.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to get TAPI version. IFX_SUCCESS – Successful.

3.11 IFX_TAPI_DEBUG_REPORT_SET

Description

Set the driver trace mask to enable or disable run-time trace messages. Multiple trace masks can

```
#define  IFX_TAPI_DEBUG_REPORT_SET              _IO(IFX_TAPI_IOC_MAGIC, 0x12)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_DEBUG_REPORT_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	dev_fd	Device descriptor.
int	IFX_TAPI_DEBUG_REPORT_SET	I/O control identifier for this operation.
unsigned long	param	Debug trace mask: M1966_DEBUG_TRACE_MASK.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to set debug report. IFX_SUCCESS – Successful.

3.12 IFX_TAPI_LINE_TYPE_SET

Description

This service configures the line type. Please note that this command only accept FXO line type (IFX_TAPI_LINE_TYPE_FXO).

```
#define  IFX_TAPI_LINE_TYPE_SET           _IOW(IFX_TAPI_IOC_MAGIC, 0x47, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_LINE_TYPE_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_LINE_TYPE_SET	I/O control identifier for this operation.
unsigned long	param	The parameter is a pointer to an IFX_TAPI_LINE_TYPE_CFG_t struct.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to set line type. IFX_SUCCESS – Successful.

3.13 IFX_TAPI_PHONE_VOLUME_SET

Description

Sets the speaker phone and microphone volume settings.

```
#define  IFX_TAPI_PHONE_VOLUME_SET        _IOW(IFX_TAPI_IOC_MAGIC, 0x42, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_PHONE_VOLUME_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_PHONE_VOLUME_SET`	I/O control identifier for this operation.
`unsigned long`	`param`	The parameter points to an `IFX_TAPI_LINE_VOLUME_t` structure.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to set volume. IFX_SUCCESS – Successful.

3.14 IFX_TAPI_FXO_HOOK_SET

Description

Issues on-/off-hook in the fxo interface.

```
#define IFX_TAPI_FXO_HOOK_SET             _IOW(IFX_TAPI_IOC_MAGIC, 0xDB, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_HOOK_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_FXO_HOOK_SET`	I/O control identifier for this operation.
`unsigned long`	`param`	Hook requested.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to perform hook switch. IFX_SUCEESS – Successful.

3.15 IFX_TAPI_FXO_FLASH_SET

Description

Issues flash-hook in the FXO interface.

```
#define IFX_TAPI_FXO_FLASH_SET            _IOW(IFX_TAPI_IOC_MAGIC, 0xDC, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_FLASH_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_FXO_FLASH_SET	I/O control identifier for this operation.
unsigned long	param	Parameter is not required.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to perform hook flash. IFX_SUCCESS – Successful.

3.16 IFX_TAPI_FXO_FLASH_CFG_SET

Description

Configuration of the fxo hook.

```
#define IFX_TAPI_FXO_FLASH_CFG_SET        _IOW(IFX_TAPI_IOC_MAGIC, 0xD7, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_FLASH_CFG_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_FXO_FLASH_CFG_SET	I/O control identifier for this operation.
unsigned long	param	Points to an IFX_TAPI_FXO_FLASH_CFG_t structure.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to set hook flash config. IFX_SUCCESS – Successful.

3.17 IFX_TAPI_FXO_BAT_ GET

Description

Receives battery status from the FXO interface.

```
#define IFX_TAPI_FXO_BAT_GET              _IOW(IFX_TAPI_IOC_MAGIC, 0xDD, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_BAT_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_FXO_BAT_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	Points to `IFX_boolean_t` type, indicating the battery status • IFX_TRUE if the FXO port is disconnected from the PSTN (battery absent). • IFX_FALSE if the FXO port is connected to the PSTN (battery present).

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to read battery status. IFX_SUCCESS – Successful.

3.18 IFX_TAPI_FXO_APOH_ GET

Description

Retrieves APOH (another phone off-hook) status of the fxo interface.

```
#define IFX_TAPI_FXO_APOH_GET            _IOW(IFX_TAPI_IOC_MAGIC, 0xDF, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_APOH_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_FXO_APOH_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	Points to `IFX_boolean_t` type, indicating APOH status. • IFX_TRUE if APOH condition is verified. • IFX_FALSE otherwise.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to read APOH status. IFX_SUCCESS – Successful.

3.19 IFX_TAPI_FXO_RING_ GET

Description

Receives ring status from the FXO interface.

```
#define IFX_TAPI_FXO_RING_GET             _IOW(IFX_TAPI_IOC_MAGIC, 0xE0, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_RING_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`chan_fd`	Channel descriptor.
`int`	`IFX_TAPI_FXO_RING_GET`	I/O control identifier for this operation.
`unsigned long`	`param`	Points to `IFX_boolean_t` type,indicating the ringing status of the FXO line. • IFX_TRUE the line is ringing. • IFX_FALSE the line is not ringing.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to read ring status. IFX_SUCCESS – Successful.

3.20 IFX_TAPI_FXO_POLARITY_ GET

Description

Receives line polarity status from the FXO interface.

```
#define IFX_TAPI_FXO_POLARITY_GET         _IOW(IFX_TAPI_IOC_MAGIC, 0xE1, int)
```

Prototype

```
int ioctl (
      int chan_fd,
      int IFX_TAPI_FXO_POLARITY_GET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_FXO_POLARITY_GET	I/O control identifier for this operation.
unsigned long	param	Points to IFX_boolean_t type, • IFX_TRUE reflects normal polarity, • IFX_FALSE reflects reversed polarity

Return Values

Data Type	Description
int	IFX_ERROR – Failed to read polarity status. IFX_SUCCESS – Successful.

3.21 IFX_TAPI_LASTERR

Description

This service returns the last error code occurred in the TAPI driver or the low level driver.

```
#define  IFX_TAPI_LASTERR              _IOW(IFX_TAPI_IOC_MAGIC, 0x48, int)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_LASTERR,
      unsigned long param );
```

Parameters

Data Type	Name	Description
`int`	`dev_fd`	Device descriptor.
`int`	`IFX_TAPI_LASTERR`	I/O control identifier for this operation.
`unsigned long`	`param`	The parameter points to a `IFX_TAPI_ErrorLine_t` structure.

Return Values

Data Type	Description
`int`	IFX_ERROR – Failed to read last error code. IFX_SUCCESS – Successful.

3.22 IFX_TAPI_FXO_LINE_MODE_SET

Description

This service is used to manage (enable/disable) the FXO channel. When disabled, the FXO channel is inoperative and it does not monitor the physical line for channel events, nor will it detect any incoming ring signal. However, it can be put back in operation using this ioctl with the “enable” parameter.

```
#define  IFX_TAPI_FXO_LINE_MODE_SET       _IOW(IFX_TAPI_IOC_MAGIC, 0xE4, int)
```

Prototype

```
int ioctl (
      int dev_fd,
      int IFX_TAPI_FXO_LINE_MODE_SET,
      unsigned long param );
```

Parameters

Data Type	Name	Description
int	chan_fd	Channel descriptor.
int	IFX_TAPI_FXO_LINE_MODE_SET	I/O control identifier for this operation.
unsigned long	param	The parameter points to a IFX_TAPI_FXO_LINE_MODES_t structure.

Return Values

Data Type	Description
int	IFX_ERROR – Failed to set line mode. IFX_SUCCESS – Successful.

4 Type and Structure Definitions

This section describes the type definitions, data types and structures used in the 73M1x66B TAPI driver.

Table 2: Summary of Types and Structure Definitions

Structure/Type Name	Description
`IFX_TAPI_CH_INIT_t`	TAPI initialization structure used by `IFX_TAPI_CH_INIT`.
`IFX_TAPI_PCM_IF_CFG_t`	PCM interface configuration structure used by `IFX_TAPI_PCM_IF_CFG_SET`.
`IFX_TAPI_PCM_ CFG_t`	Structure for PCM channel configuration.
`IFX_TAPI_EVENT_t`	Structure reported by an `IFX_TAPI_EVENT_GET` ioctl.
`IFX_TAPI_LINE_TYPE_CFG_t`	Line type configuration used by `IFX_TAPI_LINE_TYPE_SET`.
`IFX_TAPI_FXO_LINE_MODES_t`	Enumerates possible FXO channel management commands used by the `IFX_TAPI_LINE_TYPE_SET` ioctl.
`IFX_TAPI_LINE_VOLUME_t`	Configures phone volume settings.
`IFX_TAPI_FXO_HOOK_t`	Defines the possible hook status for FXO, used in `IFX_TAPI_FXO_HOOK_SET`.
`IFX_TAPI_FXO_FLASH_CFG_t`	FXO hook configuration, used in `IFX_TAPI_FXO_FLASH_CFG_SET`.
`IFX_TAPI_EVENT_ID_t`	List of event IDs.
`M1966_DEBUG_TRACE_MASK`	Trace macros used by ioctl `IFX_TAPI_DEBUG_REPORT_SET`.
`M1966_CNTRY_CODE_XX`	Country code macros used by ioctl `IFX_TAPI_CH_INIT`.

4.1 IFX_TAPI_CH_INIT_t

Description

TAPI initialization structure used by `IFX_TAPI_CH_INIT`.

Prototype

```
typedef struct
{
      unsigned char nMode;
      unsigned char nCountry;
      void * pProc;
} IFX_TAPI_CH_INIT_t;
```

Parameters

Data Type	Name	Description
`Unsigned char`	`nMode`	N/A.
`Unsigned char`	`nCountry`	Country code as defined in M1966_CNTRY_CODE_XX.
`Void *`	`pProc`	For details, see the *73M1866/73M1966 Reference Driver User Guide*.

4.2 IFX_TAPI_PCM_IF_CFG_t

Description

Structure for PCM interface configuration used by `IFX_TAPI_PCM_IF_CFG_SET`.

Prototype

```
typedef struct
{
      IFX_TAPI_PCM_IF_MODE_t nOpMode;
      IFX_TAPI_PCM_IF_DCLFREQ_t nDCLFreq;
      IFX_operation_t nDoubleClk;
      IFX_TAPI_PCM_IF_SLOPE_t nSlopeTX;
      IFX_TAPI_PCM_IF_SLOPE_t nSlopeRX;
      IFX_TAPI_PCM_IF_OFFSET_t nOffsetTX;
      IFX_TAPI_PCM_IF_OFFSET_t nOffsetRX;
      IFX_TAPI_PCM_IF_DRIVE_t nDrive;
      IFX_operation_t nShift;
      IFX_uint8_t nMCTS;
} IFX_TAPI_PCM_IF_CFG_t;
```

Parameters

Data Type	Name	Description
`IFX_TAPI_PCM_IF_MODE_t`	`nOpMode`	PCM interface mode (master or slave mode).
`IFX_TAPI_PCM_IF_DCLFREQ_t`	`nDCLFreq`	DCL frequency to be used in master and/or slave mode.
`IFX_operation_t`	`nDoubleClk`	Activation/deactivation of the double clock mode. •IFX_DISABLE: single clocking is used. •IFX_ENABLE: double clocking is used.
`IFX_TAPI_PCM_IF_SLOPE_t`	`nSlopeTX`	Slope to be considered for the PCM transmit direction.
`IFX_TAPI_PCM_IF_SLOPE_t`	`nSlopeRX`	Slope to be considered for the PCM receive direction.
`IFX_TAPI_PCM_IF_OFFSET_t`	`nOffsetTX`	Transmit bit offset.
`IFX_TAPI_PCM_IF_OFFSET_t`	`nOffsetRX`	Receive bit offset.
`IFX_TAPI_PCM_IF_DRIVE_t`	`nDrive`	Drive mode for bit 0.
`IFX_operation_t`	`nShift`	Enable/disable shift access edge. Shift the access edges by one clock cycle. • IFX_DISABLE: no shift takes place. • IFX_ENABLE: shift takes place. Note: This setting is defined only in double clock mode.
`IFX_uint8_t`	`nMCTS`	Reserved.PCM chip specific settings. Set this field to 0x00 if not advised otherwise by the IFX support team.

4.3 IFX_TAPI_PCM_ CFG_t

Description

Structure for PCM channel configuration.

Prototype

```
typedef struct
{
      unsigned long nTimeslotRX;
      unsigned long nTimeslotTX;
      unsigned long nHighway;
      unsigned long nResolution;
} IFX_TAPI_PCM_CFG_t;
```

Parameters

Data Type	Name	Description
unsigned long	nTimeslotRX	PCM timeslot for the receive direction.
unsigned long	nTimeslotTX	PCM timeslot for the transmit direction.
unsigned long	nHighway	Defines the PCM highway number which is connected to the channel.
unsigned long	nResolution	Defines the PCM interface coding, values defined in IFX_TAPI_PCM_RES_t.

4.4 IFX_TAPI_EVENT_t

Description

This structure is reported by an IFX_TAPI_EVENT_GET ioctl.

Prototype

```
typedef struct
{
      IFX_TAPI_EVENT_ID_t id;
      IFX_uint16_t ch;
      IFX_uint16_t more;
      IFX_TAPI_EVENT_DATA_t data;
} IFX_TAPI_EVENT_t;
```

Parameters

Data Type	Name	Description
IFX_TAPI_EVENT_ID_t	id	Event type and sub-type.
IFX_uint16_t	ch	FXO channel number.
IFX_uint16_t	more	This field is used to report whether new events are ready (IFX_TRUE) or not (IFX_FALSE).
IFX_TAPI_EVENT_DATA_t	data	N/A

4.5 IFX_TAPI_LINE_TYPE_CFG_t

Description

Line type configuration used by ioctl IFX_TAPI_LINE_TYPE_SET.

Prototype

```
typedef struct
{
      IFX_TAPI_LINE_TYPE_t lineType;
      IFX_uint8_t nDaaCh;
} IFX_TAPI_LINE_TYPE_CFG_t;
```

Data Type	Name	Description
IFX_TAPI_LINE_TYPE_t	lineType	Must be IFX_TAPI_LINE_TYPE_FXO.
IFX_uint8_t	nDaaCh	N/A

4.6 IFX_TAPI_FXO_LINE_MODES_t

Description

This data type enumerates possible FXO channel management commands used by the IFX_TAPI_LINE_TYPE_SET ioctl.

Prototype

```
/** Defines the possible line modes for fxo, used in
IFX_TAPI_FXO_LINE_MODES_t */

typedef enum
{
   /** Disabled. */
   IFX_TAPI_FXO_LINE_MODE_DISABLED = 0,
   /** Active. */
   IFX_TAPI_FXO_LINE_MODE_ACTIVE = 1
} IFX_TAPI_FXO_LINE_MODES_t;
```

Data Type	Name	Description
IFX_TAPI_FXO_LINE_MODE_DISABLED	0	Disable FXO channel.
IFX_TAPI_FXO_LINE_MODE_ACTIVE	1	Enable FXO channel.

4.7 IFX_TAPI_LINE_VOLUME_t

Description

Structure used to configure phone volume settings.

Prototype

```
typedef struct
{
      int nGainRx;
      int nGainTx;
} IFX_TAPI_LINE_VOLUME_t;
```

Data Type	Name	Description
int	nGainRx	Volume setting for the receiving path.
int	nGainTx	Volume setting for the transmitting path.

4.8 IFX_TAPI_FXO_HOOK_t

Description

Defines the possible hook status for fxo, used in IFX_TAPI_FXO_HOOK_SET**.**

Prototype

```
typedef enum
{
      IFX_TAPI_FXO_HOOK_ONHOOK = 0,
      IFX_TAPI_FXO_HOOK_OFFHOOK = 1
} IFX_TAPI_FXO_HOOK_t;
```

Data Type	Name	Description
IFX_TAPI_FXO_HOOK_ONHOOK	0	On-hook.
IFX_TAPI_FXO_HOOK_OFFHOOK	1	Off-hook.

4.9 IFX_TAPI_FXO_FLASH_CFG_t

Description

Hook configuration for FXO, used in IFX_TAPI_FXO_FLASH_CFG_SET.

Prototype

```
typedef struct
{
      IFX_uint32_t nFlashTime;
} IFX_TAPI_FXO_FLASH_CFG_t;
```

Data Type	Name	Description
IFX_uint32_t	nFlashTime	Duration of a flash-hook. Default 100 ms.

4.10 IFX_TAPI_EVENT_ID_t

Description

List of event IDs. These are the `#define` macros of the FXO event identification.

Prototype

```
typedef enum
{
I      IFX_TAPI_EVENT_NONE           = IFX_TAPI_EVENT_TYPE_NONE | 0x0000,
       IFX_TAPI_EVENT_FXO_BAT_FEEDED = IFX_TAPI_EVENT_TYPE_FXO | 0x0001,
       IFX_TAPI_EVENT_FXO_BAT_DROPPED= IFX_TAPI_EVENT_TYPE_FXO | 0x0002,
       IFX_TAPI_EVENT_FXO_POLARITY   = IFX_TAPI_EVENT_TYPE_FXO | 0x0003,
       IFX_TAPI_EVENT_FXO_RING_START = IFX_TAPI_EVENT_TYPE_FXO | 0x0004,
       IFX_TAPI_EVENT_FXO_RING_STOP  = IFX_TAPI_EVENT_TYPE_FXO | 0x0005,
       IFX_TAPI_EVENT_FXO_OSI        = IFX_TAPI_EVENT_TYPE_FXO | 0x0006,
       IFX_TAPI_EVENT_FXO_APOH       = IFX_TAPI_EVENT_TYPE_FXO | 0x0007,
       IFX_TAPI_EVENT_FXO_NOPOH      = IFX_TAPI_EVENT_TYPE_FXO | 0x0008,
} IFX_TAPI_EVENT_ID_t;
```

4.11 M1966_DEBUG_TRACE_MASK

Description

Trace macros used by ioctl `IFX_TAPI_DEBUG_REPORT_SET`.

Prototype

```
#define M1966_DEBUG_EVENT         0x00000001
#define M1966_DEBUG_INIT          0x00000002
#define M1966_DEBUG_RING_PATH     0x00000004
#define M1966_DEBUG_TRACE         0x00000008
#define M1966_DEBUG_COUNTRY_CODE  0x00000010
#define M1966_DEBUG_CLIP          0x00000020
#define M1966_DEBUG_LINE_STATE    0x00000040
#define M1966_DEBUG_IOCTL         0x00000080
#define M1966_DEBUG_PCM           0x00000100
#define M1966_DEBUG_BARRIER       0x00000200
#define M1966_DEBUG_INT           0x00000400
#define M1966_DEBUG_PHU           0x00000800
#define M1966_DEBUG_TAPI          0x00001000
#define M1966_DEBUG_KPROC         0x00002000
#define M1966_DEBUG_SPI           0x00004000
#define M1966_DEBUG_ERROR         0x80000000
```

4.12 M1966_CNTRY_CODE_XX

Description

Country code macros used by ioctl `IFX_TAPI_CH_INIT`.

Prototype

```
/*******************************************************************
**  73M1966 Country code List - Internet Country Codes
*******************************************************************/
#define M1966_CNTRY_CODE_AR             0       /* "Argentina"    */
#define M1966_CNTRY_CODE_AU             1       /* "Australia"    */
#define M1966_CNTRY_CODE_AT             2       /* "Austria"      */
#define M1966_CNTRY_CODE_BH             3       /* "Bahrain"      */
#define M1966_CNTRY_CODE_BE             4       /* "Belgium"      */
#define M1966_CNTRY_CODE_BR             5       /* "Brazil"       */
#define M1966_CNTRY_CODE_BG             6       /* "Bulgaria"     */
#define M1966_CNTRY_CODE_CA             7       /* "Canada"       */
#define M1966_CNTRY_CODE_CL             8       /* "Chile"        */
#define M1966_CNTRY_CODE_C1             9       /* "ChineData"    */
#define M1966_CNTRY_CODE_C2             10      /* "ChinaVoice"   */
#define M1966_CNTRY_CODE_CO             11      /* "Columbia"     */
#define M1966_CNTRY_CODE_HR             12      /* "Croatia"      */
#define M1966_CNTRY_CODE_TB             13      /* "TBR 21"       */
#define M1966_CNTRY_CODE_CY             14      /* "Cyprus"       */
#define M1966_CNTRY_CODE_CZ             15      /* "Czech Rep"    */
#define M1966_CNTRY_CODE_DK             16      /* "Denmark"      */
#define M1966_CNTRY_CODE_EC             17      /* "Equador"      */
#define M1966_CNTRY_CODE_EG             18      /* "Egypt"        */
#define M1966_CNTRY_CODE_SV             19      /* "El Salvador"  */
#define M1966_CNTRY_CODE_FI             20      /* "Finland"      */
#define M1966_CNTRY_CODE_FR             21      /* "France"       */
#define M1966_CNTRY_CODE_DE             22      /* "Germany"      */
#define M1966_CNTRY_CODE_GR             23      /* "Greece"       */
#define M1966_CNTRY_CODE_GU             24      /* "Guam"         */
#define M1966_CNTRY_CODE_HK             25      /* "Hong Kong"    */
#define M1966_CNTRY_CODE_HU             26      /* "Hungary"      */
#define M1966_CNTRY_CODE_IS             27      /* "Iceland"      */
#define M1966_CNTRY_CODE_IN             28      /* "India"        */
#define M1966_CNTRY_CODE_ID             29      /* "Indonesia"    */
#define M1966_CNTRY_CODE_IE             30      /* "Ireland"      */
#define M1966_CNTRY_CODE_IL             31      /* "Israel"       */
#define M1966_CNTRY_CODE_IT             32      /* "Italy"        */
#define M1966_CNTRY_CODE_JP             33      /* "Japan"        */
#define M1966_CNTRY_CODE_JO             34      /* "Jordan"       */
#define M1966_CNTRY_CODE_KZ             35      /* "Kazakhstan"   */
#define M1966_CNTRY_CODE_KW             36      /* "Kuwait"       */
#define M1966_CNTRY_CODE_LV             37      /* "Latvia"       */
#define M1966_CNTRY_CODE_LB             38      /* "Lebanon"      */
#define M1966_CNTRY_CODE_LU             39      /* "Luxembourg"   */
#define M1966_CNTRY_CODE_MO             40      /* "Macao"        */
#define M1966_CNTRY_CODE_MY             41      /* "Malaysia"     */
#define M1966_CNTRY_CODE_MT             42      /* "Malta"        */
#define M1966_CNTRY_CODE_MX             43      /* "Mexico"       */
#define M1966_CNTRY_CODE_MA             44      /* "Morocco"      */
#define M1966_CNTRY_CODE_NL             45      /* "Netherlands"  */
#define M1966_CNTRY_CODE_NZ             46      /* "New Zealand"  */
#define M1966_CNTRY_CODE_NG             47      /* "Nigeria"      */
#define M1966_CNTRY_CODE_NO             48      /* "Norway"       */
```

```
#define M1966_CNTRY_CODE_OM             49      /* "Oman"         */
#define M1966_CNTRY_CODE_PK             50      /* "Pakistan"     */
#define M1966_CNTRY_CODE_PR             51      /* "Peru"         */
#define M1966_CNTRY_CODE_PH             52      /* "Philippines"  */
#define M1966_CNTRY_CODE_PL             53      /* "Poland"       */
#define M1966_CNTRY_CODE_PT             54      /* "Portugal"     */
#define M1966_CNTRY_CODE_RO             55      /* "Romainia"     */
#define M1966_CNTRY_CODE_RU             56      /* "Russia"       */
#define M1966_CNTRY_CODE_SA             57      /* "Saudi Arabia" */
#define M1966_CNTRY_CODE_SG             58      /* "Singapore"    */
#define M1966_CNTRY_CODE_SK             59      /* "Slovakia"     */
#define M1966_CNTRY_CODE_SI             60      /* "Slovenia"     */
#define M1966_CNTRY_CODE_ZA             61      /* "S. Africa"    */
#define M1966_CNTRY_CODE_KR             62      /* "S. Korea"     */
#define M1966_CNTRY_CODE_ES             63      /* "Spain"        */
#define M1966_CNTRY_CODE_SE             64      /* "Sweden"       */
#define M1966_CNTRY_CODE_CH             65      /* "Switzerland"  */
#define M1966_CNTRY_CODE_SY             66      /* "Syria"        */
#define M1966_CNTRY_CODE_TW             67      /* "Taiwan"       */
#define M1966_CNTRY_CODE_TH             68      /* "Thailand"     */
#define M1966_CNTRY_CODE_AE             69      /* "UAE"          */
#define M1966_CNTRY_CODE_UK             70      /* "UK"           */
#define M1966_CNTRY_CODE_US             71      /* "USA"          */
#define M1966_CNTRY_CODE_YE             72      /* "Yemen"        */
```

5 Related Documentation

The following 73M1x66B documents are available from Teridian Semiconductor Corporation:

73M1866B/73M1966B Reference Driver User Manual
73M1866B/73M1966B Data Sheet
73M1866B/73M1966B Demo Board User Manual
73M1866B/73M1966B GUI User Guide
73M1866B/73M1966B Layout Guidelines
73M1x66 Worldwide Design Guide
TAPI V3 User's Manual (available from Infineon)

6 Contact Information

For more information about Teridian Semiconductor products or to check the availability of the 73M1866B and 73M1966B, contact us at:

6440 Oak Canyon Road
Suite 100
Irvine, CA 92618-5201

Telephone: (714) 508-8800
FAX: (714) 508-8878
Email: modem.support@teridian.com

For a complete list of worldwide sales offices, go to http://www.teridian.com.

Revision History

Revision	Date	Description
1.0	10/28/2008	First publication.
2.0	3/20/2009	Provided detail IOCTL interface. Removed redundant implementation description (those should be referred to the *73M1866B/73M1966B Reference Driver User Guide*). Restructured document to conform to Teridian standard format.
2.2	7/16/2010	Added FXO line enable/disable IOCTLs.