TERIDIAN SEMICONDUCTOR CORP.

Simplifying System Integration™

73M1822/73M1922
Hardware Module for SMDK412 User Guide

January 20, 2010
Rev. 1.1
UG_1x22_054

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 capabilities of the 73M1822/73M1922 Hardware Module for the SMDK2412. This driver software is provided for use and integration by Teridian customers on their individual platforms.

Throughout this document the 73M1822/73M1922 Hardware Module for the SMDK2412 is simply referred to as “driver” or “device driver”. The 73M1822 and 73M1922 is collectively referred to as the 73M1x22 or the device.

1.1 Purpose and Scope

The 73M1822/73M1922 Hardware Module for the SMDK2412 provides the necessary system interfaces for the control and management of the 73M1x22. The driver supports calls from a user space application or kernel module and translates these to and from the device. The driver can be used as is, in whole or in part, or customized to accommodate a customer’s unique environment.

The scope of this document includes discussion of driver’s architecture and design and interface to the user. The 73M1822/73M1922 Hardware Module for the SMDK2412 is intended specifically for the Samsung S3C2412 processor on the SMDK2412 development board and for the Linux® operating system version 2.6.19 as customized by Samsung for the aforementioned system. Layers above the reference driver address software interfaces which may pre-exist for a given application and the layer below addresses hardware related interfaces between the processor and the 73M1x22 device.

1.2 Conventions Used in this Guide

This document uses the following conventions:

- Software code, IOCTL names, modem events, data types, and Linux commands are presented in Courier font.

1.3 Acronyms

MAFE – Modem Analog Front End
DAA – Data Access Arrangement
IOCTL – Linux I/O Control
ISR – Interrupt Service Routine
BSP – Board Support Package

2 Overview

2.1 Driver Architecture

The driver provides a framework by which applications can leverage the features of the 73M1x22 device. The main interface of the driver provides an abstraction layer for monitoring and control of the device status. Figure 1 depicts the driver functional block diagram.

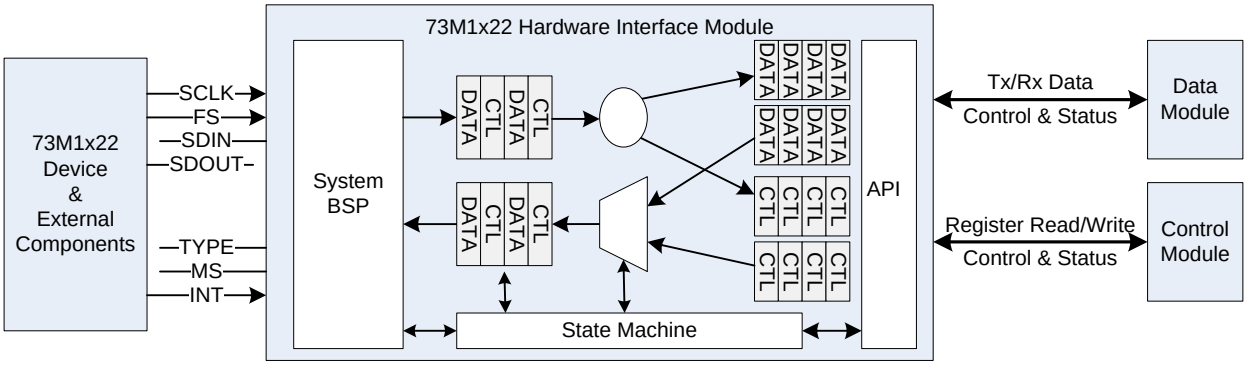

Figure 1: Driver Functional Block Diagram

2.2 Functional Overview

The 73M1x22 driver is built as a Linux loadable module (for Linux) or integrated with the operating system kernel. It is brought into operation by a user application or by an operating system start up 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. Before exiting the initialization the driver enters its main operational state via the scheduling of one of many timers that make up the driver's main processing.

This description is valid for Linux 2.6. The 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 the `insmod` command. This command inserts the driver module into the kernel which in turn registers with the kernel using the default major number of 250

Once installed, the driver is a self-contained module running independently along with the kernel processes. Its main purpose is to monitor and control the MAFE interface, transfer MAFE frames to and from the 73M1x22 device and to provide access to the 73M1x22 device for management purposes, via standard driver access methods such as `open`, `close`, `select`, `ioctl`, etc. The following sections provide an overview of that functionality.

3 Accessing the Driver from the Linux User Space

The driver provides the link between the modem device and the user application via a standard linux character device. Access to the driver is done via Linux supplied file descriptors The following sections describe how the driver is brought into action based on the operating system environment.

3.1 mknod

The device descriptors can be created using the `mknod` command as illustrated below:

```
mknod -m 660 /dev/tsc_1x22_mafe_0   c 251 0
mknod -m 660 /dev/tsc_1x22_mafe_1   c 251 1
```

In this example, one descriptor (`tsc_1x22_ctl_mafe_0`) is created with major number 251, minor number 0, and one descriptor (`tsc_1x22_ctl_mafe_l`) is created with major number 251, minor number 1. The minor number associated with the file descriptor is unused by the driver in this implementation.

3.2 open

Description

Once the driver is installed and the device descriptors are created, the driver service can be accessed by user applications via standard C library `open()`, and subsequently reversed with `close()`

Prototype

```
int open (
      const char * path,
      int flags);
```

Parameters

Data Type	Name	Description
`Const char *`	**`path`**	Path to the descriptor created by mknod.
`int`	`flags`	O_RDWR.

Return Values

Data Type	Description
`int`	File descriptor.

3.3 close

Description

A file descriptor obtained with an open() command must be subsequently released with a corresponding close..

Prototype

```
int close (int fd);
```

Parameters

Data Type	Name	Description
int	fd	File descriptor to close.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.4 read

Description

A file descriptor obtained with an open() command can be called with the standard Linux read command to obtain MAFE audio samples. The read command reads as many 16 bit samples into the user supplied buffer as possible and return the amount of data that was copied.

Prototype

```
size_t read (int fd, void * buf, size_t buf_size);
```

Parameters

Data Type	Name	Description
int	fd	File descriptor.
void *	buf	Buffer to read samples into.
size_t	buf_size	Maximum size of data that can be copied.

Return Values

Data Type	Description
size_t	Amount of data that was actually copied.

3.5 write

Description

A file descriptor obtained with an open() command can be called with the standard Linux write command to transmit MAFE audio samples. The write command copy as many 16 bit samples from the user supplied buffer as possible and return the amount of data that was copied.

Prototype

```
size_t write (int fd, const void * buf, size_t buf_size);
```

Parameters

Data Type	Name	Description
int	fd	File descriptor.
Const void *	buf	Buffer to copy samples from.
size_t	buf_size	Maximum size of data that can be copied.

Return Values

Data Type	Description
size_t	Amount of data that was actually copied.

3.6 ioctl

Description

A file descriptor obtained with an `open()` command can be called with the standard Linux `ioctl` command to perform various tasks.

Prototype

```
int ioctl (int fd, int request, … args);
```

Parameters

Data Type	Name	Description
int	fd	File descriptor.
int	request	Requested service.
…	args	Request dependent arguments.

Return Values

Data Type	Description
int	Request dependent return value.

The following sections describe the detail of each IOCTL command.

3.6.1 TSC_1X22_MAFE_GET_RS232

Description

Retrieves the current status of the RS232 RTS, DTS, CTR and DTR pins.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_GET_RS232,
      void );
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_GET_RS232	I/O control identifier for this operation.
void	NA	

Return Values

Data Type	Description
unsigned char	See RS232 #defines.

3.6.2 TSC_1X22_MAFE_SET_RS232

Description

Sets the current status of the RS232 RTS, DTS, CTR and DTR pins and then returns the resulting status via call to TSC_1X22_MAFE_GET_RS232.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_SET_RS232,
      unsigned char RS232);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_SET_RS232	I/O control identifier for this operation.
unsigned char	RS232	Bits to set. See RS232 #defines.

Return Values

Data Type	Description
unsigned char	See TSC_1X22_MAFE_GET_RS232.

3.6.3 TSC_1X22_MAFE_GET_RX_STAT

Description

Retrieves the current status of the Rx Data Sample FIFO.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_GET_RX_STAT,
      void );
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_GET_RX_STAT	I/O control identifier for this operation.
void	NA	

Return Values

Data Type	Description
Unsigned char	See Rx and Tx FIFO status #defines.

3.6.4 TSC_1X22_MAFE_GET_TX_STAT

Description

Retrieves the current status of the Tx Data Sample FIFO.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_GET_TX_STAT,
      void );
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_GET_TX_STAT	I/O control identifier for this operation.
void	NA	

Return Values

Data Type	Description
Unsigned char	See Rx and Tx FIFO status #defines.

3.6.5 TSC_1X22_MAFE_RD_REG_NB

Description

It copies the last read value of the supplied register number into the members of the supplied structure pointer and then requests another read access from the device.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_RD_REG_NB,
      TSC_1X22_REG_IOCTL_t *);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_RD_REG_NB	I/O control identifier.
TSC_1X22_REG_IOCTL_t *		See TSC_1X22_REG_IOCTL_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.6 TSC_1X22_MAFE_RD_REG_IM

Description

It copies the last read value of the supplied register number into the members of the supplied structure pointer without requesting another read from the device.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_RD_REG_IM,
      TSC_1X22_REG_IOCTL_t *);
```

Parameters

Data Type	Name	Description
Int	Fd	Channel descriptor.
Int	TSC_1X22_MAFE_RD_REG_IM	I/O control identifier.
TSC_1X22_REG_IOCTL_t *		See TSC_1X22_REG_IOCTL_t.

Return Values

Data Type	Description
Int	0 = Success. -1 = Failure.

3.6.7 TSC_1X22_MAFE_RD_REG_BL

Description

Issues a MAFE command to the 73M1x22 to the supplied register and copies the read value into the members of the supplied structure pointer.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_RD_REG_BL,
      TSC_1X22_REG_IOCTL_t *);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_WR_REG_BL	I/O control identifier.
TSC_1X22_REG_IOCTL_t *		See TSC_1X22_REG_IOCTL_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.8 TSC_1X22_MAFE_WR_REG_NB

Description

Issues a write command to the 73M1x22 device of the supplied register number with the supplied data.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_WR_REG_NB,
      TSC_1X22_REG_IOCTL_t *);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_WR_REG_NB	I/O control identifier.
TSC_1X22_REG_IOCTL_t *		See TSC_1X22_REG_IOCTL_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.9 TSC_1X22_MAFE_WR_REG_BL

Description

Issues a write command to the 73M1x22 device of the supplied register number with the supplied data. It only returns once confirmation is received that the command has been sent.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_WR_REG_BL,
      TSC_1X22_REG_IOCTL_t *);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_WR_REG_BL	I/O control identifier.
TSC_1X22_REG_IOCTL_t *		See TSC_1X22_REG_IOCTL_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.10 TSC_1X22_MAFE_SET_REG_AUTO_POLL

Description

This IOCLT starts/stops the automatic polling of the 73M1x22 device registers dependent upon the user supplied arguments. An argument (mask) of 0 disables this feature.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_SET_REG_AUTO_POLL,
      unsigned int mask);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_SET_REG_AUTO_POLL	I/O control identifier.
unsigned int	mask	If mask[bit i] is set then poll register i.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.11 TSC_1X22_MAFE_IRQ_HW

Description

This IOCTL enables the use of the HW interrupt pin on the 73M1x22 device to be reported through the HW Module. The current SMDK interrupt assigned is through EINT11 / IRQ 39. The interrupt is enabled by default after this call.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_HW);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_HW	I/O control identifier.

Return Values

Data Type	Description
int	Returns 0 upon success, -1 upon failure

3.6.12 TSC_1X22_MAFE_IRQ_SW

Description

This IOCTL enables the use of a SW based interrupt scheme for the 73M1x22 device to be reported through the HW Module. Auto polling (see TSC_1X22_MAFE_SET_REG_AUTO_POLL) must be enabled for register 0x03 for this feature to work properly. The interrupt mutex/enables remains in their current state after this call.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_SW);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_SW	I/O control identifier.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.13 TSC_1X22_MAFE_IRQ_ENABLE

Description

This IOCTL enables the interrupt scheme for the 73M1x22 device to be reported through the HW Module.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_ENABLE);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_ENABLE	I/O control identifier.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.14 TSC_1X22_MAFE_IRQ_DISABLE

Description

This IOCTL disables the reporting of interrupts for the 73M1x22 device through the HW Module.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_DISABLE);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_DISABLE	I/O control identifier.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.15 TSC_1X22_MAFE_IRQ_DOWN

Description

This IOCTL attempts a mutex down call on an internal mutex. This IOCTL blocks until successful. This mutex is up'd interrupt service routine (SW or HW triggered) and by TSC_1X22_MAFE_IRQ_UP.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_DOWN);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_DOWN	I/O control identifier.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

3.6.16 TSC_1X22_MAFE_IRQ_DOWN_TRY

Description

This IOCTL attempts a mutex down_try call on an internal mutex. This IOCTL does not block and always completes.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_DOWN_TRY);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_DOWN_TRY	I/O control identifier.

Return Values

Data Type	Description
int	Returns 0 upon successfully acquiring the lock, non-zero upon failure.

3.6.17 TSC_1X22_MAFE_IRQ_UP

Description

This IOCTL attempts a mutex up call on an internal mutex. This IOCTL does not block and always completes.

Prototype

```
int ioctl (
      int fd,
      int TSC_1X22_MAFE_IRQ_UP);
```

Parameters

Data Type	Name	Description
int	fd	Channel descriptor.
int	TSC_1X22_MAFE_IRQ_UP	I/O control identifier.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

4 Accessing the Driver from the Linux Kernel Space

When accessing the driver from the Linux Kernel space the user should use the exported API function calls instead of the standard device file based user space functions.

4.1 mknod

The device does not require a character device file to be created via a mknod command to access the driver functionality from the kernel space.

4.2 tsc_1x22_mafe_open

Description

Once the driver is installed (via insmod) the driver service can be accessed and initialized with a `tsc_1x22_mafe_open()` call.

Prototype

```
int tsc_1x22_mafe_open (void);
```

Parameters

Data Type	Name	Description
`void`	**NA**	NA

Return Values

Data Type	Description
`int`	0 = Success. Not 0 = Failure.

4.3 tsc_1x22_mafe_close

Description

A file descriptor obtained with a `tsc_1x22_mafe_open()` command must be subsequently released with a corresponding `close`.

Prototype

```
int tsc_1x22_mafe_close (void);
```

Parameters

Data Type	Name	Description
`Void`	NA	NA

Return Values

Data Type	Description
`int`	0 = Success. Not 0 = Failure.

4.4 tsc_1x22_mafe_read

Description

The user can call this `tsc_1x22_mafe_read` command to obtain MAFE audio samples. The read command reads as many 16 bit samples into the user supplied kernel space buffer as possible and returns the amount of data that was copied.

Prototype

```
size_t tsc_1x22_mafe_read (void * buf, size_t buf_size);
```

Parameters

Data Type	Name	Description
`int`	`fd`	File descriptor.
`void *`	`vuf`	Kernel buffer to read samples into.
`size_t`	`vuf_size`	Maximum size of data that can be copied.

Return Values

Data Type	Description
`size_t`	Amount of data that was actually copied.

4.5 tsc_1x22_mafe_write

Description

The user may call `tsc_1x22_mafe_write` to transmit MAFE audio samples. The write command copies as many 16 bit samples from the user supplied kernel space buffer as possible and returns the amount of data that was copied.

Prototype

```
size_t write (const void * buf, size_t buf_size);
```

Parameters

Data Type	Name	Description
`int`	`fd`	File descriptor.
`const void *`	`buf`	Kernel buffer to copy samples from.
`size_t`	`buf_size`	Maximum size of data that can be copied.

Return Values

Data Type	Description
`size_t`	Amount of data that was actually copied.

4.6 ioctl

Description

The driver, while accessed from the kernel space, does not require the calling of a specific IOCTL command. The following sections detail the equivalent exported direct function calls that are available in the Kernel space.

4.6.1 tsc_1x22_rs232_get

Description

Retrieves the current status of the RS232 RTS, DTS, CTR and DTR pins.

IOCTL Equivalent

`TSC_1X22_MAFE_GET_RS232`

Prototype

```
unsigned char tsc_1x22_rs232_get(void);
```

Parameters

Data Type	Name	Description
`Void`	NA	NA

Return Values

Data Type	Description
`unsigned char`	See RS232 #defines.

4.6.2 tsc_1x22_rs232_set

Description

Sets the current status of the RS232 RTS,DTS,CTR and DTR pins and then returns the resulting status via call to tsc_1x22_rs232_get.

IOCTL Equivalent

`TSC_1X22_MAFE_SET_RS232`

Prototype

```
unsigned char tsc_1x22_rs232_set(unsigned char RS232);
```

Parameters

Data Type	Name	Description
`unsigned char`	RS232	Bits to set. See RS232 #defines.

Return Values

Data Type	Description
`unsigned char`	See `tsc_1x22_rs232_get`.

4.6.3 tsc_1x22_mafe_get_rx_stat

Description

Retrieves the current status of the Rx Data Sample FIFO.

IOCTL Equivalent

`TSC_1X22_MAFE_GET_RX_STAT`

Prototype

```
int tsc_1x22_mafe_get_rx_stat (void);
```

Parameters

Data Type	Name	Description
`void`	NA	NA

Return Values

Data Type	Description
`unsigned char`	See Rx and Tx FIFO status #defines.

4.6.4 tsc_1x22_mafe_get_tx_stat

Description

Retrieves the current status of the Tx Data Sample FIFO.

IOCTL Equivalent

`TSC_1X22_MAFE_GET_TX_STAT`

Prototype

```
int tsc_1x22_mafe_get_tx_stat (void);
```

Parameters

Data Type	Name	Description
`void`	NA	

Return Values

Data Type	Description
`unsigned char`	See Rx and Tx FIFO status #defines.

4.6.5 tsc_1x22_mafe_rd_rg_nb

Description

It copies the last read value of the supplied register number into the members of the supplied structure pointer and then requests another read update from the device.

IOCTL Equivalent

`TSC_1X22_MAFE_RD_REG_NB`

Prototype

```
int tsc_1x22_mafe_rd_rg_nb(TSC_1X22_REG_t *);
```

Parameters

Data Type	Name	Description
`TSC_1X22_REG_t *`		See `TSC_1X22_REG_t`.

Return Values

Data Type	Description
`int`	0 = Success. -1 = Failure.

4.6.6 tsc_1x22_mafe_rd_rg_im

Description

It copies the last read value of the supplied register number into the members of the supplied structure pointer without requesting another read from the device.

IOCTL Equivalent

TSC_1X22_MAFE_RD_REG_IM

Prototype

```
int tsc_1x22_mafe_rd_rg_im(TSC_1X22_REG_t *);
```

Parameters

Data Type	Name	Description
TSC_1X22_REG_t *		See TSC_1X22_REG_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

4.6.7 tsc_1x22_mafe_rd_rg_bl

Description

Issues a MAFE command to the 73M1x22 to the supplied register and copies the read value into the members of the supplied structure pointer.

IOCTL Equivalent

TSC_1X22_MAFE_RD_REG_BL

Prototype

```
int tsc_1x22_mafe_rd_rg_bl(TSC_1X22_REG_t *);
```

Parameters

Data Type	Name	Description
TSC_1X22_REG_t *		See TSC_1X22_REG_t.

Return Values

Data Type	Description
int	0 = Success. -1 = Failure.

4.6.8 tsc_1x22_mafe_wr_rg_nb

Description

Issues a write command to the 73M1x22 device of the supplied register number with the supplied data.

IOCTL Equivalent

`TSC_1X22_MAFE_WR_REG_NB`

Prototype

```
int tsc_1x22_mafe_wr_rg_nb(TSC_1X22_REG_t *);
```

Parameters

Data Type	Name	Description
`TSC_1X22_REG_t *`		See `TSC_1X22_REG_t`.

Return Values

Data Type	Description
`int`	0 = Success. -1 = Failure.

4.6.9 tsc_1x22_mafe_wr_rg_bl

Description

Issues a write command to the 73M1x22 device of the supplied register number with the supplied data. It only returns once confirmation is received that the command has been sent.

IOCTL Equivalent

`TSC_1X22_MAFE_WR_REG_BL`

Prototype

```
int tsc_1x22_mafe_wr_rg_bl(TSC_1X22_REG_t *);
```

Parameters

Data Type	Name	Description
`TSC_1X22_REG_t *`		See `TSC_1X22_REG_t`.

Return Values

Data Type	Description
`int`	0 = Success. -1 = Failure.

4.6.10 tsc_1x22_mafe_set_reg_auto_poll

Description

This starts/stops the automatic polling of the 73M1x22 device registers dependent upon the user supplied arguments. An argument (mask) of 0 disables this feature.

IOCTL Equivalent

`TSC_1X22_MAFE_SET_REG_AUTO_POLL`

Prototype

```
int tsc_1x22_mafe_set_reg_auto_poll(unsigned int mask);
```

Parameters

Data Type	Name	Description
`unsigned int`	`mask`	If mask[bit i] is set then poll register i.

Return Values

Data Type	Description
`int`	0 = Success. -1 = Failure.

4.6.11 tsc_1x22_mafe_irq_hw

Description

This function enables the use of the HW interrupt pin on the 73M1x22 device to be reported through the HW Module. The current SMDK interrupt assigned is through EINT11 / IRQ 39. The interrupt is enabled by default after this call.

IOCTL Equivalent

`TSC_1X22_MAFE_IRQ_HW`

Prototype

```
void tsc_1x22_mafe_irq_hw(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
`void`	NA

4.6.12 tsc_1x22_mafe_irq_sw

Description

This IOCTL enables the use of a SW based interrupt scheme for the 73M1x22 device to be reported through the HW Module. Auto polling (see TSC_1X22_MAFE_SET_REG_AUTO_POLL) must be enabled for register 0x03 for this feature to work properly. The interrupt mutex/enables remains in their current state after this call.

IOCTL Equivalent

TSC_1X22_MAFE_IRQ_SW

Prototype

```
void tsc_1x22_mafe_irq_sw(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
void	NA

4.6.13 tsc_1x22_mafe_irq_enable

Description

This IOCTL enables the interrupt scheme for the 73M1x22 device to be reported through the HW Module.

IOCTL Equivalent

TSC_1X22_MAFE_IRQ_ENABLE

Prototype

```
void tsc_1x22_mafe_irq_enable(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
void	NA

4.6.14 tsc_1x22_mafe_irq_disable

Description

This IOCTL disables the reporting of interrupts for the 73M1x22 device through the HW Module.

IOCTL Equivalent

`TSC_1X22_MAFE_IRQ_DISABLE`

Prototype

```
void tsc_1x22_mafe_irq_disable(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
`void`	NA

4.6.15 tsc_1x22_mafe_irq_down

Description

This IOCTL attempts a mutex down call on an internal mutex. This IOCTL blocks until successful. This mutex is up'd interrupt service routine (SW or HW triggered) and by `TSC_1X22_MAFE_IRQ_UP`.

IOCTL Equivalent

`TSC_1X22_MAFE_IRQ_DOWN`

Prototype

```
void tsc_1x22_mafe_irq_down(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
`void`	NA

4.6.16 tsc_1x22_mafe_irq_down_try

Description

This IOCTL attempts a mutex down_try call on an internal mutex. This IOCTL does not block and always complete.

IOCTL Equivalent

`TSC_1X22_MAFE_IRQ_DOWN_TRY`

Prototype

```
int tsc_1x22_mafe_irq_down_try(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
int	Returns 0 upon successfully acquiring the lock, non-zero upon failure.

4.6.17 tsc_1x22_mafe_irq_up

Description

This IOCTL attempts a mutex up call on an internal mutex. This IOCTL does not block and always completes.

IOCTL Equivalent

`TSC_1X22_MAFE_IRQ_UP`

Prototype

```
void tsc_1x22_mafe_irq_up(void);
```

Parameters

Data Type	Name	Description
NA	NA	NA

Return Values

Data Type	Description
void	NA

4.7 Type and Structure Definition Reference

This section contains the type definitions, reference of data type and structure used in the driver.

4.7.1 RS232 #defines

Description

#defines used in the TSC_1X22_MAFE_GET_RS232 and TSC_1X22_MAFE_SET_RS232 IOCTLs.

Prototype

```
#define RS232_DTR_O         0x40
#define RS232_RTS_O         0x02
#define RS232_DSR_I         0x80
#define RS232_CTS_I         0x01
```

4.8 Rx and Tx FIFO status #defines

Description

#defines used in the TSC_1X22_MAFE_GET_RX_STAT and TSC_1X22_MAFE_GET_TX_STAT IOCTLs.

Prototype

```
#define FIFO_FF               0x0008
#define FIFO_AF               0x0004
#define FIFO_AE               0x0002
#define FIFO_EE               0x0001
```

4.9 TSC_1X22_REG_IOCTL_t

Description

Structure used in the TSC_1X22_MAFE_RD_REG_NB, TSC_1X22_MAFE_RD_REG_BL, TSC_1X22_MAFE_WR_REG_NB and TSC_1X22_MAFE_WR_REG_BL IOCTLs.

Prototype

```
typedef struct
{
unsigned char reg;
unsigned char dat;
} TSC_1X22_REG_IOCTL_t;
```

4.10 TSC_1X22_REG_t

Description

Structure used by the functions.

Prototype

```
typedef struct
{
unsigned char reg;
unsigned char dat;
} TSC_1X22_REG_t;
```

4.11 Driver Source and Include Files

The driver software is written exclusively in the C programming language and consists of the following:

Table 1: Driver Source Code Files

File Name	Description
`tsc_1x22_mafe_module.h`	Contains headers for the Linux module.
`tsc_1x22_mafe_module.c`	Contains definitions for the Linux module.
`tsc_1x22_mafe_api.h`	Contains headers for the exported API functions.
`tsc_1x22_mafe_api.c`	Contains definitions and support for the API functions.
`tsc_1x22_mafe_ioctls.h`	Contains definitions of the Linux IOCTLs.
`tsc_1x22_mafe_global.h`	Contains global variables and definitions.

4.12 Compile Time Configurable Parameter

The following parameter can be changed at compile and build time to reflect the real customer environment. These parameters are found in the header files:

```
#define LINUX_DEVICE_MAJOR       250
```

5 Related Documentation

The following 73M1x22B documents are available from Teridian Semiconductor Corporation:

73M1822B73M1X22 Data Sheet
73M1822B73M1X22 Layout Guidelines
73M1x22 Worldwide Design Guide
73M1822/73M1922 Control Module User Guide
73M1822/73M1922 Hardware Module for SMDK412 User Guide
73M1822/73M1922 Modem API User Guide
73M1822/73M1922 Modem CTL Application User Guide
73M1822/73M1922 MicroDAA Software Architecture

6 Contact Information

For more information about Teridian Semiconductor products or to check the availability of the 73M1822 and 73M1922, 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	12/23/2009	First publication.
1.1	01/20/2010	In Section 3.6.5, added “and then requests another read access from the device” to the end of the description. Added Section 3.6.6, TSC_1x22_MAFE_RD_REG_IM. In Section 4.6.5, added “and then requests another read update from the device” to the end of the description. Added Section 4.6.6, tsc_1x22_mafe_rd_rg_im.