



**TIP250-SW-42**  
**8 Mbytes Flash Memory**  
**VxWorks Device Driver**

Version 1.0

**Reference Manual**  
Issue 1.0

December 20, 2001

---

TEWS TECHNOLOGIES GmbH  
Am Bahnhof 7  
D-25469 Halstenbek  
Germany  
Tel.: +49 (0)4101 4058-0  
Fax.: +49 (0)4101 4058-19  
<http://www.tews.com>  
E-mail: [info@tews.com](mailto:info@tews.com)

# **TIP250-SW-42**

## **8 Mbytes Flash Memory**

### **VxWorks Device Driver**

This document contains information, which is proprietary to TEWS TECHNOLOGIES. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES reserves the right to change the product described in this document at any time without notice.

This product has been designed to operate with IndustryPack® compatible carriers. Connection to incompatible hardware is likely to cause serious damage.

TEWS TECHNOLOGIES is not liable for any damage arising out of the application or use of the device described herein.

IndustryPack is a registered trademark of GreenSpring Computers, Inc

©2001 by TEWS TECHNOLOGIES GmbH

<b>Issue</b>	<b>Description</b>	<b>Date</b>
1.0	First Issue	December 20, 2001

# Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>4</b>
<b>2</b>	<b>INSTALLATION.....</b>	<b>5</b>
2.1	Install the driver to VxWorks .....	5
2.2	TIP250 mounted on MVME162.....	5
2.3	TIP250 mounted on extern carrier board .....	5
<b>3</b>	<b>I/O SYSTEM FUNCTIONS .....</b>	<b>6</b>
3.1	t250Drv().....	6
3.2	t250DevCreate().....	7
<b>4</b>	<b>I/O INTERFACE FUNCTIONS.....</b>	<b>9</b>
4.1	open() .....	9
4.2	ioctl() .....	10
4.2.1	FIO_T250_CHIP_ERASE .....	11
4.2.2	FIO_T250_SECTOR_ERASE .....	12
4.2.3	FIO_T250_BLOCK_WRITE .....	13
4.2.4	FIO_T250_GET_PTR .....	15
4.2.5	FIO_T250_PROTECT .....	16
4.2.6	FIO_T250_UNPROTECT.....	17
4.2.7	FIO_T250_RESET .....	18

# **1 Introduction**

The TIP250-SW-42 VxWorks device driver software allows the operation of the TIP250 8Mbytes Flash Memory IP conforming to the VxWorks I/O system specification. This includes a device-independent basic I/O interface with *open()* and *ioctl()* functions.

The TIP250 driver includes the following functions:

- Erasing the whole 8 MB Flash Memory (Chip Erase)
- Erasing a single 128 KB Sector (Sector Erase)
- Programming the Flash Memory
- Enable Flash Memory write protection (default after reset)
- Disable Flash Memory write protection to allow erasing and programming.
- Resetting the Flash Memory logic.

## 2 Installation

The device driver software is delivered on a 3½" HD diskette.

Following files are located on the diskette:

t250drv.c	TIP250 Driver Source
tip250.h	TIP250 Driver Include File
t250defs.h	TIP250 Internal Driver Include File
ipchip.h	TIP250 Driver IPIC programming model definitions
t250exam.c	TIP250 Device Driver Example Application
TIP250-SW-42.pdf	TIP250 VxWorks Device Driver Manual

For installation the files have to be copied to the desired target directory.

### 2.1 Install the driver to VxWorks

You have to perform the following steps to install the TIP250 device driver to your VxWorks-System.

- Build the object code of the TIP250 device driver.
- Link or load the driver object file to your VxWorks-System.
- Call the '*t250Drv()*' function to install the device driver.

### 2.2 TIP250 mounted on MVME162

The TIP250-SW-42 Device Driver Software supports the onboard IP-slots of the MVME162 by default. That means, the IP-Chip will be set up by the driver, if one of the specific IO-addresses is selected when creating the device.

### 2.3 TIP250 mounted on extern carrier board

The TIP250-SW-42 Device Driver Software supports also other IP carrier boards, but you have to take care, that your system allows full access to the TIP250. The following points must be guaranteed:

- Access to TIP250 module I/O and MEM addresses must be possible.
- The CPU must detect interrupts from the TIP250 module over the VME-Bus. The corresponding VME-Bus Interrupts must be enabled on the CPU (Refer to the BSP-documentation).

## **3 I/O system functions**

This chapter describes the driver-level interface to the I/O system. The purpose of these functions is to install the driver in the I/O system, add and initialize devices.

### **3.1 t250Drv()**

#### **NAME**

t250Drv() - installs the TIP250 device driver in the I/O system and initialize the driver.

#### **SYNOPSIS**

STATUS t250Drv(void)

#### **DESCRIPTION**

This function installs the TIP250 driver in the I/O system, allocates driver resources and initializes them. The call of this function is the first thing we have to do, before adding any device to the system or performing any I/O request.

#### **RETURNS**

OK or ERROR if the driver cannot be installed

#### **SEE ALSO**

VxWorks Programmer's Guide: I/O System

## 3.2 t250DevCreate()

### NAME

t250DevCreate() - adds a TIP250 device to the system and initialize device hardware

### SYNOPSIS

```
STATUS t250DevCreate
(
    char          *name,
    unsigned char *IpIoAddr,
    unsigned char *IpMemAddr,
    int           vector,
    int           level
)
```

### DESCRIPTION

This routine creates a device for the module specified by the IP I/O and IP MEM address that will be serviced by the TIP250 device driver. This function must be called before performing any I/O request to this device.

Function Arguments:

<b>name</b>	Pointer to null terminated unique string to identify this device (e.g. "/t250A").
<b>IpIoAddr</b>	Pointer to the IP I/O space (TIP250 device register). For TIP250 modules plugged onboard on a MVME162-xxx this is for example 0xFFF58000 for the first IP slot. For TIP250 modules plugged on external carrier boards this address depends on the mapping of the VMEbus windows and the address configuration of the carrier board.
<b>IpMemAddr</b>	Pointer to the IP Memory space (8 MB Flash Memory). For TIP250 modules plugged onboard on a MVME162-xxx this address can be any unused address space (end of DRAM until 0xEFFFFFFF). Be sure that this address space is mapped by the MMU and <u>not</u> mapped to the VMEbus by the VMEchip2 (see also syslib.c for necessary adaptations). For TIP250 modules plugged on external carrier boards this address depends on the memory space address configuration of the carrier board. Be also sure that the memory space size is set to 8MB.
<b>vector</b>	Contains the interrupt vector used for this device. For 68K systems or TIP250 modules plugged on external VME carrier boards this can be any free user vector (64..254).
<b>level</b>	Contains the interrupt level on which the TIP250 generates interrupts (1..7). For TIP250 modules plugged onboard on a MVME162-xxx the driver setup the IPCHIP with the desired interrupt level. For modules plugged on an external carrier board this level must match to the interrupt level configuration of the carrier board. The driver call the sysIntEnable() function by default if the TIP250 is plugged external, to enable the desired interrupt level to the CPU.

## EXAMPLE

```
#include    "tip250.h"

...

/*-----
   Create the device "/t250D" for the module plugged at
   local IP-slot D of a MVME162.
   -----*/
status = t250DevCreate(
                "/t250D",
                (unsigned char*)0xFFF58300,
                (unsigned char*)0xE0000000,
                0xA0,
                3 );

...
```

## RETURNS

OK or ERROR if the driver is not installed or the device already exists or any other error occurred during the creation.

## ERRORS

S_t250Drv_NXIO	No TIP250 device found at the specified address. Possible reasons could be, the IP I/O space is not accessible.
S_t250Drv_IPARAM	The desired interrupt level is out of range (valid 1..7).
S_t250Drv_NOMEM	Unable to allocate memory for internal device control structures.
VxWorks error codes	Error codes returned by the VxWorks function iosDevAdd().

# 4 I/O interface functions

This chapter describes the interface to the basic I/O system.

## 4.1 open()

### NAME

open() - open a device or file

### SYNOPSIS

```
int open
(
    const char    *name,          /* name of the device to open      */
    int           flags,         /* not used for TIP250 driver, must be 0 */
    int           mode           /* not used for TIP250 driver, must be 0 */
)
```

### DESCRIPTION

Before I/O can be performed to the TIP250 device, a file descriptor must be opened by invoking the basic I/O function *open()*.

### EXAMPLE

```
...
/*-----
   open the device named "/t250D" for I/O
   -----*/
fd = open ("/t250D", 0, 0);
...

```

### RETURNS

A device descriptor number, or ERROR if the device does not exist or no device descriptors are available.

### SEE ALSO

ioLib, basic I/O routine - open()

## 4.2 ioctl()

### NAME

ioctl() - performs an I/O control function

### SYNOPSIS

```
int ioctl
(
    int          fd,          /* device descriptor          */
    int          function,   /* function code              */
    int          arg          /* optional function-dependent argument */
)
```

### DESCRIPTION

Special I/O operation that does not fit to the standard basic I/O calls will be performed by calling the *ioctl()* function with a specific function code and an optional function-dependent argument.

The **function** parameter selects the action, which will be executed by the driver. The structure of the **arg** parameter depends on the function. The different functions are described behind the general description of this function.

FIO_T250_CHIP_ERASE	Erasing the whole 8 MB Flash Memory (Chip Erase)
FIO_T250_SECTOR_ERASE	Erasing a single 128 KB Sector (Sector Erase)
FIO_T250_BLOCK_WRITE	Programming the Flash Memory
FIO_T250_GET_PTR	Get a pointer to the start of the Flash Memory
FIO_T250_PROTECT	Enable Flash Memory write protection (default after reset)
FIO_T250_UNPROTECT	Disable Flash Memory write protection
FIO_T250_RESET	Reset the Flash Memory logic

### RETURNS

OK or ERROR if the device descriptor does not exist, the function code is unknown or an error occurred.

### INCLUDE FILES

ioLib.h, tip250.h

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.1 FIO\_T250\_CHIP\_ERASE

### NAME

FIO\_T250\_CHIP\_ERASE - Erasing the whole 8 MB Flash Memory

### DESCRIPTION

This ioctl function erases the whole 8MB Flash Memory. All previous programmed data will be lost. After successful completion each bit of the flash should be set to '1'. The chip erase operation can take up to 90 seconds (typical 45 seconds). The optional argument can be omitted.

### Note

Erasing is only possible if the write protection (default after Reset) of the Flash is disabled (see also *FIO\_T250\_UNPROTECT*).

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;

...

/*-----
   Erase the whole 8 MB Flash Memory
   -----*/
status = ioctl(fd, FIO_T250_CHIP_ERASE, 0);

if (status == ERROR) {
    /* handle function specific error conditions */
}

...
```

### ERRORS

S_t250Drv_PROTECT	The Flash Memory is write (erase) protected. Execute ioctl function <i>FIO_T250_UNPROTECT</i> to unprotect the Flash.
S_t250Drv_NOSEM	Unable to allocate a semaphore for synchronization.
S_t250Drv_TIMEOUT	The erase operation did not finish within the allowed interval. This error could occur if interrupts does not work properly.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.2 FIO\_T250\_SECTOR\_ERASE

### NAME

FIO\_T250\_SECTOR\_ERASE - Erasing a single 128 KB Sector

### DESCRIPTION

This ioctl function erases a specified 128 KB sector of the Flash Memory. All previous programmed data within this sector will be lost. All other sectors will be not affected. After successful completion each bit within the sector should be set to '1'. The sector erase operation can take up to 15 seconds.

The optional argument *arg* must be set to the sector index (0..63) to erase. The address offset and range for sector index 0 is 0x000000...0x01FFFF, for sector index 1 is 0x020000...0x03FFFF and so on.

### Note

Erasing is only possible if the write protection (default after Reset) of the Flash is disabled (see also *FIO\_T250\_UNPROTECT*).

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;

...

/*-----
   Erase the last sector of the Flash Memory (offset 0x007E0000)
   -----*/
status = ioctl(fd, FIO_T250_SECTOR_ERASE, 63);

if (status == ERROR) {
    /* handle function specific error conditions */
}

...
```

### ERRORS

S_t250Drv_PROTECT	The Flash Memory is write (erase) protected. Execute ioctl function <i>FIO_T250_UNPROTECT</i> to unprotect the Flash.
S_t250Drv_IPARAM	Sector index is out of range (valid 0..63).
S_t250Drv_NOSEM	Unable to allocate a semaphore for synchronization.
S_t250Drv_TIMEOUT	The erase operation did not finish within the allowed interval. This error could occur if interrupts does not work properly.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.3 FIO\_T250\_BLOCK\_WRITE

### NAME

FIO\_T250\_BLOCK\_WRITE - Programming the Flash Memory

### DESCRIPTION

This ioctl function writes the contents of a user-supplied buffer into the specified location of the Flash Memory.

A pointer to the caller's parameter buffer (T250\_PROG\_PARAM) must be passed by the argument *arg* to the driver.

The T250\_PROG\_PARAM structure has the following layout:

```
typedef struct {  
    unsigned long    offset;  
    unsigned char    *buf;  
    unsigned long    len;  
} T250_PROG_PARAM;
```

#### ***offset***

Start offset in the Flash Memory. Every "word-aligned" (modulo 2) address offset within the Flash Memory is valid.

#### ***buf***

Pointer to a valid data buffer which provides the data to be programmed. Up to 8 MB at once can be programmed.

#### ***len***

Number of data bytes to program. Because write operations are performed 16-bit wise, the value of len must be modulo 2. Therefore the minimum amount of data written to the Flash memory is 2 byte.

### **Note**

Programming is only possible if the write protection (default after Reset) of the Flash is disabled (see also *FIO\_T250\_UNPROTECT*).

## EXAMPLE

```
#include "tip250.h"

...

int          fd;
int          status;
unsigned char buffer[256];
T250_PROG_PARAM ProgParam;
...

/*-----
   Write 256 byte into the Flash memory
   -----*/
ProgParam.offset = 0x7E0000;
ProgParam.buf = buffer;
ProgParam.len = 256;

status = ioctl(fd, FIO_T250_BLOCK_WRITE, (int)&ProgParam);

if (status == ERROR) {
    /* handle function specific error conditions */
}
...
```

## ERRORS

S_t250Drv_PROTECT	The Flash Memory is write protected. Execute ioctl function <i>FIO_T250_UNPROTECT</i> to unprotect the Flash.
S_t250Drv_TIMEOUT	Programming of a Flash memory cell did not finish within the allowed interval (about 1 millisecond).
S_t250Drv_IRANGE	The data buffer to be programmed overlaps the Flash memory boundaries. No write will be performed.
S_t250Drv_IPARAM	Either the structure member offset or len are not modulo 2. No write will be performed.
S_t250Drv_VERIFY	The contents of the programmed data location does not match the contents of the supplied data buffer.

## SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.4 FIO\_T250\_GET\_PTR

### NAME

FIO\_T250\_GET\_PTR - Get a pointer to the start of the Flash Memory

### DESCRIPTION

This ioctl function returns the start address of the Flash memory to the caller. This address can be used for direct read access to the Flash memory. A pointer to a variable, which receives the Flash start address must be passed by the argument *arg* to the driver.

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;
unsigned char *BufPtr

...

/*-----
   Get the base address of the Flash Memory
   -----*/
status = ioctl(fd, FIO_T250_GET_PTR, (int)&BufPtr);

if (status == ERROR) {
    /* handle function specific error conditions */
}

printf("Contents of first Flash location = %d\n", *BufPtr);

...
```

### ERRORS

No function specific error codes.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.5 FIO\_T250\_PROTECT

### NAME

FIO\_T250\_PROTECT - Enable write protection.

### DESCRIPTION

This ioctl function enables the write protection of the Flash memory. All further write or erase requests will be fail. The Flash memory remain in this state until the FIO\_T250\_UNPROTECT function is executed. The optional argument can be omitted.

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;

...

/*-----
   Protect Flash Memory against program and erase operations
   -----*/
status = ioctl(fd, FIO_T250_PROTECT, 0);

if (status == ERROR) {
    /* handle function specific error conditions */
}

...
```

### ERRORS

No function specific error codes.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.6 FIO\_T250\_UNPROTECT

### NAME

FIO\_T250\_UNPROTECT - Disable write protection.

### DESCRIPTION

This ioctl function disables the write protection of the Flash memory. This function must be called before any erase or write commands are possible. The optional argument can be omitted.

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;

...

/*-----
   Disable write protection
   -----*/
status = ioctl(fd, FIO_T250_UNPROTECT, 0);

if (status == ERROR) {
    /* handle function specific error conditions */
}

...
```

### ERRORS

No function specific error codes.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

## 4.2.7 FIO\_T250\_RESET

### NAME

FIO\_T250\_RESET – Reset Flash Memory

### DESCRIPTION

This ioctl function resets the Flash memory logic. This could be necessary after an erase function returns with an error.  
The optional argument can be omitted.

### EXAMPLE

```
#include      "tip250.h"

...

int          fd;
int          status;

...

/*-----
   Reset the Flash memory
   -----*/
status = ioctl(fd, FIO_T250_RESET, 0);

if (status == ERROR) {
    /* handle function specific error conditions */
}

...
```

### ERRORS

No function specific error codes.

### SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.