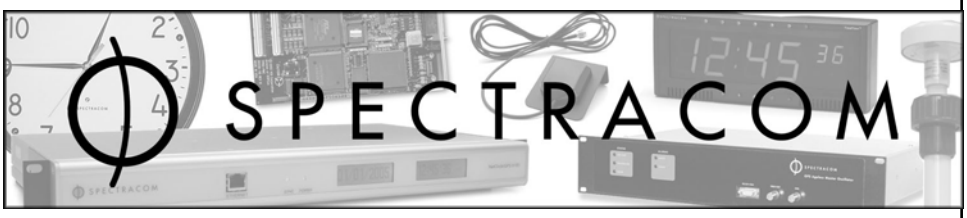
TPRO-PCI-U/TSAT-PCI-U
SYNCHRONIZABLE TIMECODE GENERATOR with UNIVERSAL PCI BUS INTERFACE

Solaris Driver Application Programmer's Guide

95 Methodist Hill Drive
Rochester, NY 14623
Phone: US +1.585.321.5800
Fax: US +1.585.321.5219



www.spectracomcorp.com
Part Number 1186-5004-0050
Manual Revision B
24 April 2009

Specifications subject to change or improvement without notice.

SPECTRACOM LIMITED WARRANTY

LIMITED WARRANTY

Spectracom warrants each new product manufactured and sold by it to be free from defects in software, material, workmanship, and construction, except for batteries, fuses, or other material normally consumed in operation that may be contained therein AND AS NOTED BELOW, for five years after shipment to the original purchaser (which period is referred to as the "warranty period"). This warranty shall not apply if the product is used contrary to the instructions in its manual or is otherwise subjected to misuse, abnormal operations, accident, lightning or transient surge, repairs or modifications not performed by Spectracom.

The GPS receiver is warranted for one year from date of shipment and subject to the exceptions listed above. The power adaptor, if supplied, is warranted for one year from date of shipment and subject to the exceptions listed above.

THE ANALOG CLOCKS ARE WARRANTED FOR ONE YEAR FROM DATE OF SHIPMENT AND SUBJECT TO THE EXCEPTIONS LISTED ABOVE.

THE TIMECODE READER/GENERATORS ARE WARRANTED FOR ONE YEAR FROM DATE OF SHIPMENT AND SUBJECT TO THE EXCEPTIONS LISTED ABOVE.

The Rubidium oscillator, if supplied, is warranted for two years from date of shipment and subject to the exceptions listed above.

All other items and pieces of equipment not specified above, including the antenna unit, antenna surge suppressor and antenna pre-amplifier are warranted for 5 years, subject to the exceptions listed above.

WARRANTY CLAIMS

Spectracom's obligation under this warranty is limited to in-factory service and repair, at Spectracom's option, of the product or the component thereof, which is found to be defective. If in Spectracom's judgment the defective condition in a Spectracom product is for a cause listed above for which Spectracom is not responsible, Spectracom will make the repairs or replacement of components and charge its then current price, which buyer agrees to pay.

Spectracom shall not have any warranty obligations if the procedure for warranty claims is not followed. Users must notify Spectracom of the claim with full information as to the claimed defect. Spectracom products shall not be returned unless a return authorization number is issued by Spectracom.

Spectracom products must be returned with the description of the claimed defect and identification of the individual to be contacted if additional information is needed. Spectracom products must be returned properly packed with transportation charges prepaid.

Shipping expense: Expenses incurred for shipping Spectracom products to and from Spectracom (including international customs fees) shall be paid for by the customer, with the following exception. For customers located within the United States, any product repaired by Spectracom under a "warranty repair" will be shipped back to the customer at Spectracom's expense unless special/faster delivery is requested by customer.

Spectracom highly recommends that prior to returning equipment for service work, our technical support department be contacted to provide trouble shooting assistance while the equipment is still installed. If equipment is returned without first contacting the support department and "no problems are found" during the repair work, an evaluation fee may be charged.

EXCEPT FOR THE LIMITED WARRANTY STATED ABOVE, SPECTRACOM DISCLAIMS ALL WARRANTIES OF ANY KIND WITH REGARD TO SPECTRACOM PRODUCTS OR OTHER MATERIALS PROVIDED BY SPECTRACOM, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OR MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Spectracom shall have no liability or responsibility to the original customer or any other party with respect to any liability, loss, or damage caused directly or indirectly by any Spectracom product, material, or software sold or provided by Spectracom, replacement parts or units, or services provided, including but not limited to any interruption of service, excess charges resulting from malfunctions of hardware or software, loss of business or anticipatory profits resulting from the use or operation of the Spectracom product or software, whatsoever or howsoever caused. In no event shall Spectracom be liable for any direct, indirect, special or consequential damages whether the claims are grounded in contract, tort (including negligence), or strict liability.

EXTENDED WARRANTY COVERAGE

Extended warranties can be purchased for additional periods beyond the standard five-year warranty. Contact Spectracom no later than the last year of the standard five-year warranty for extended coverage.

SPECTRACOM 95 Methodist Hill Drive Rochester, NY 14623

+1.585.321.5800 FAX: +1.585.321.5218 www.spectracomcorp.com sales@spectracomcorp.com

Table of Contents

1 Overview

The Solaris Driver for the Spectracom TPRO/TSAT PCI boards provides the interface for multiple users to access the board using the API documented in Chapter Three.

This driver contains a 32-bit & 64-bit version for SPARC system running Solaris 10. It also contains a 32-bit & 64-bit version for x64 PC architectures running Solaris 10. The API library is delivered in two configurations: one for 32-bit linking and one for 64-bit linking.

Most API methods may be used unrestricted. Several methods, however (used to configure operational parameters of the board), will return as ***busy*** if there are users actively waiting for events based on parameters that would be changed by allowing the method to complete. One diagnostic API method will also return as ***busy*** under the same circumstances. These methods include:

- TPRO_resetFirmware (diagnostic)
- TPRO_setHeartbeat
- TPRO_setMatchTime
- TPRO_setPropDelayCorr
- TPRO_setTime
- TPRO_setYear

2 Installing the Driver

If the system contains a previously installed version of the TPRO Solaris driver, that version must first be uninstalled. As root, execute the following command from the shell:

pkgrm tpro

NOTE: For driver 1.42 or older, you may have to remove the following files manually and reboot before installing a newer driver:

rm /platform/i86pc/kernel/drv/tpro
rm /platform/i86pc/kernel/drv/tpro.conf
rm /platform/i86pc/kernel/drv/amd64/tpr
m /usr/lib/libtpro.a
rm /usr/lib/amd64/libtpro.a
rm /usr/include/tpro.h

The driver and API library (lib files and header) are contained in the package:

tpro.pkg

This package is delivered via the archive file:

tpro.<rev>.<arch>.pkg.tar.gz

- where **<rev>** is the current driver revision
- where <**arch**> is either: ***i86pc64*** or ***i86pc32*** ***sun4u64*** or ***sun4u32***

As root, place the above archive file in a directory and within that directory extract the package using the following command:

gunzip –c tpro.<rev>.<arch>.pkg.tar.gz | tar –xvf –

From the same directory, execute the following command to add the driver and API library package to the system:

pkgadd -d tpro.pkg

2.1 File Locations

The following table illustrates where package contents ultimately reside in relation to the support of either API classification.

SPARC Architecture

File	Destination	64-bit API	32-bit API
tpro (driver)	/usr/kernel/drv/sparcv9	Yes	Yes
tpro.conf	/usr/kernel/drv/sparcv9	Yes	Yes
libtpro.a	/usr/lib/sparcv9	Yes	No
libtpro_32.a	/usr/lib	No	Yes
tpro.h	/usr/include	Yes	Yes

X64 Architecture

File	Destination	64-bit API	32-bit API
tpro (driver)	/usr/kernel/drv/amd64	Yes	Yes
tpro.conf	/usr/kernel/drv/amd64	Yes	Yes
libtpro.a	/usr/lib/amd64	Yes	No
libtpro_32.a	/usr/lib	No	Yes
tpro.h	/usr/include	Yes	Yes

2.2 Application Example

The driver package also includes folders with example programs and scripts to interface to the board. The source code and make files for the example programs are included. All the example programs were compiled using **Sun Studio 12**.

There are 2 sets of example programs:

TProSamples.i86pc.tar.gz – 32-bit apps for x86 based machines
TProSamples.sun4u.tar.gz – 32-bit apps for Sun based machines

To run an examples program:

./GetTime -i <x>
./GetInfo -i <x>

Where <x> is the board you want to communicate with. (ex. 0,1,2)

3 Interface to the Solaris API

3.1 Header File

The following is the “TPRO.H” API Interface Header File.

```
#ifndef _defined_TPRO_H
#define _defined_TPRO_H

/*****************************************************************************
**  Module:     tpro.h
**  Date:       08/08/07
**  Purpose:    This is the TPRO/TSAT PCI Card interface file.
**
** (C) Copyright 2006 Spectracom Corporation  All rights reserved.
**
*****************************************************************************/

#ifdef __cplusplus
extern "C" {
#endif

#define DLL_EXPORT /* */

#include <sys/types.h>
#include <sys/int_types.h>


/*============================================================================
        SUPPORT CONSTANTS
============================================================================*/

/*
**  Heartbeat constants
*/

#define SIG_PULSE                   ( 0xE5 )    /* heartbeat is a pulse */
#define SIG_SQUARE                  ( 0xE7 )    /* heartbeat is a squarewave */

#define SIG_NO_JAM                  ( 0 )       /* start next cycl */
#define SIG_JAM                     ( 1 )       /* start immediately */

/*
**  Match constants
*/

#define MATCH_TIME_START            ( 0 )       /* start time */
#define MATCH_TIME_STOP             ( 1 )       /* stop time */

/*
**  board sync options
*/
#define TPRO_DISABLE_SYNC           ( 0x0 )     /* freewheel */
#define TPRO_ENABLE_SYNC            ( 0x1 )     /* sync to input */
#define TPRO_SYNC_STAT_FREE         ( 0x0 )     /* sync is freewheel */

/*
** Oscillator frequencies - for PMC/cPCI boards only
*/
#define OSC_OUT_OFF                 ( 0 )
#define OSC_OUT_1KHZ                ( 1 )
#define OSC_OUT_1MHZ                ( 2 )
#define OSC_OUT_5MHZ                ( 3 )
#define OSC_OUT_10MHZ               ( 4 )
```

```
/*
** Propagation delay min's and max's for CPCI and PCI boards
*/
#define PMC_CPCI_PROP_DELAY_MIN     ( -999 )
#define PMC_CPCI_PROP_DELAY_MAX     ( 999 )
#define PCI_PROP_DELAY_MIN          ( -1000 )
#define PCI_PROP_DELAY_MAX          ( 8999 )

/*
** Length of firmware rev string
*/
#define TPRO_FIRMWARE_LENGTH        ( 4 )

/*
** Length of driver version string "XX.YY"
** (not including termination)
*/
#define TPRO_DRV_VERSION_LENGTH     ( 5 )


/*=========================================================================
        ERROR CODES
=========================================================================*/
#define TPRO_SUCCESS                    (0)   /* success */
#define TPRO_HANDLE_ERR                 (1)   /* error bad handle */
#define TPRO_OBJECT_ERR                 (2)   /* error creating obj */
#define TPRO_CLOSE_HANDLE_ERR           (3)   /* err closing device */
#define TPRO_DEVICE_NOT_OPEN_ERR        (4)   /* device not opened */
#define TPRO_INVALID_BOARD_TYPE_ERR     (5)   /* invalid device */
#define TPRO_FREQ_ERR                   (6)   /* invalid frequency */
#define TPRO_YEAR_PARM_ERR              (7)   /* invalid year */
#define TPRO_DAY_PARM_ERR               (8)   /* invalid day */
#define TPRO_HOUR_PARM_ERR              (9)   /* invalid hour */
#define TPRO_MIN_PARM_ERR               (10)  /* invalid minutes */
#define TPRO_SEC_PARM_ERR               (11)  /* invalid seconds */
#define TPRO_DELAY_PARM_ERR             (12)  /* invalid delay */
#define TPRO_TIMEOUT_ERR                (13)  /* device timed out */
#define TPRO_COMM_ERR                   (14)  /* communication error */
#define TPRO_DEV_BUSY                   (15)  /* device busy */

/*=========================================================================
        TPRO BOARD OBJECT
=========================================================================*/

typedef struct TPRO_BoardObj
{
    int             file_descriptor;    /* handle to device */
    unsigned short  devid;              /* PCI Device ID */
    unsigned short  options;            /* PCI Subsystem Product ID */
}
TPRO_BoardObj;


/*=========================================================================
        TPRO ALTITUDE OBJECT
=========================================================================*/

typedef struct TPRO_AltObj
{
    float           meters;     /* altitude in meters */
}
TPRO_AltObj;


/*=========================================================================
        TPRO DATE OBJECT
=========================================================================*/

typedef struct TPRO_DateObj
{
```

```
    unsigned short  year;           /* year  */
    unsigned char   month;          /* month */
    unsigned char   day;            /* day   */
}
TPRO_DateObj;

/*=========================================================================
        TPRO LONGITUDE/LATTITUDE OBJECT
=========================================================================*/

typedef struct TPRO_LongLat
{
    unsigned short  degrees;        /* degrees */
    float           minutes;        /* minutes */
}
TPRO_LongObj, TPRO_LatObj;


/*=========================================================================
        TPRO MATCH OBJECT
=========================================================================*/

typedef struct TPRO_MatchObj
{
    unsigned char   matchType;      /* start/stop time */
    double          seconds;        /* seconds */
    unsigned char   minutes;        /* minutes */
    unsigned char   hours;          /* hours */
    unsigned short  days;           /* days */
}
TPRO_MatchObj;


/*=========================================================================
        TPRO SATINFO OBJECT
=========================================================================*/

typedef struct TPRO_SatObj
{
    unsigned char   satsTracked;    /* count of satellites tracked */
    unsigned char   satsView;       /* count satellites in view */
}
TPRO_SatObj;

/*=========================================================================
        TPRO HEARTBEAT OBJECT
=========================================================================*/

typedef struct TPRO_HeartObj
{
    unsigned char   signalType;     /* square or pulse */
    unsigned char   outputType;     /* jamming option */
    double          frequency;      /* heartbeat freq */
}
TPRO_HeartObj;

/*=========================================================================
        TPRO TIME OBJECT
=========================================================================*/

typedef struct TPRO_TimeObj
{
    double          secsDouble;     /* seconds floating pt */
    unsigned char   seconds;        /* seconds whole num */
    unsigned char   minutes;        /* minutes */
    unsigned char   hours;          /* hours */
    unsigned short  days;           /* days */
    unsigned short  sync;           /* bit 15 flagsInvalid(1);  bit 2 SYNC, bit 1 TCODE;
                                all others 0 */
}
TPRO_TimeObj;
```

```
/*=========================================================================
        TPRO WAIT OBJECT
=========================================================================*/

typedef struct TPRO_WaitObj
{
    unsigned int    ticks;          /* num of clock ticks to wait */
    double          seconds;
    unsigned char   minutes;
    unsigned char   hours;
    unsigned short  days;
}
TPRO_WaitObj;

/*=========================================================================
        TPRO MEM OBJECT FOR PEEK/POKE
=========================================================================*/

typedef struct TPRO_MemObj
{
    unsigned short  offset;         /* offset from base register */
    unsigned int   value;          /* value to use at register location */
}
TPRO_MemObj;


/*=========================================================================
        PUBLIC ROUTINE PROTOTYPES
=========================================================================*/

DLL_EXPORT
unsigned char TPRO_open             (TPRO_BoardObj **hnd, char *deviceName);

DLL_EXPORT
unsigned char TPRO_close            (TPRO_BoardObj *hnd);

DLL_EXPORT
unsigned char TPRO_getAltitude      (TPRO_BoardObj *hnd, TPRO_AltObj *Altp);

DLL_EXPORT
unsigned char TPRO_getDate          (TPRO_BoardObj *hnd, TPRO_DateObj *Datep);

DLL_EXPORT
unsigned char TPRO_getDriver        (TPRO_BoardObj *hnd, char *version);

DLL_EXPORT
unsigned char TPRO_getFirmware      (TPRO_BoardObj *hnd, char *firmware);

DLL_EXPORT
unsigned char TPRO_getFPGA          (TPRO_BoardObj *hnd, unsigned char *fpga);

DLL_EXPORT
unsigned char TPRO_getLatitude      (TPRO_BoardObj *hnd, TPRO_LatObj *Latp);

DLL_EXPORT
unsigned char TPRO_getLongitude     (TPRO_BoardObj *hnd, TPRO_LongObj *Longp);

DLL_EXPORT
unsigned char TPRO_getSatInfo       (TPRO_BoardObj *hnd, TPRO_SatObj *Satp);

DLL_EXPORT
unsigned char TPRO_getTime          (TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);

DLL_EXPORT
unsigned char TPRO_resetFirmware    (TPRO_BoardObj *hnd);

DLL_EXPORT
unsigned char TPRO_setHeartbeat     (TPRO_BoardObj *hnd, TPRO_HeartObj *Heartp);

DLL_EXPORT
```

```
unsigned char TPRO_setMatchTime     (TPRO_BoardObj *hnd, TPRO_MatchObj *Matchp);

DLL_EXPORT
unsigned char TPRO_setOscillator    (TPRO_BoardObj *hnd, unsigned char *freq);

DLL_EXPORT
unsigned char TPRO_setPropDelayCorr (TPRO_BoardObj *hnd, int *uSec);

DLL_EXPORT
unsigned char TPRO_setTime          (TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);

DLL_EXPORT
unsigned char TPRO_setYear          (TPRO_BoardObj *hnd, unsigned short *yr);

DLL_EXPORT
unsigned char TPRO_simEvent         (TPRO_BoardObj *hnd);

DLL_EXPORT
unsigned char TPRO_synchControl     (TPRO_BoardObj *hnd, unsigned char *enbp);

DLL_EXPORT
unsigned char TPRO_synchStatus      (TPRO_BoardObj *hnd, unsigned char *status);

DLL_EXPORT
unsigned char TPRO_waitEvent        (TPRO_BoardObj *hnd, TPRO_WaitObj *waitp);

DLL_EXPORT
unsigned char TPRO_waitHeartbeat    (TPRO_BoardObj *hnd, unsigned int *ticks);

DLL_EXPORT
unsigned char TPRO_waitMatch        (TPRO_BoardObj *hnd, unsigned int *ticks);

DLL_EXPORT
unsigned char TPRO_peek             (TPRO_BoardObj *hnd, TPRO_MemObj *pMem);

DLL_EXPORT
unsigned char TPRO_poke             (TPRO_BoardObj *hnd, TPRO_MemObj *pMem);

#ifdef __cplusplus
}
#endif

#endif  /* _defined_TPRO_H */
```

3.2 TPRO API — Restrictions

The device driver and API allow multiple users to exercise most of the TPRO API routines without restriction. The following routines are restricted such that if a secondary thread attempts to exercise them while one or more threads are waiting for a related event(s), the offending thread will return a **busy** status. It is up to the caller to decide on the course of action if a busy status is returned. The table following illustrates the active wait-types that will result in a ***busy*** status if the corresponding method is invoked:

API Method	Related Active Wait Type
TPRO_resetFirmware()	Event, Match, Heartbeat
TPRO_setHeartbeat()	Heartbeat
TPRO_setMatchTime()	Match
TPRO_setPropDelayCorr()	Event, Match, Heartbeat
TPRO_setTime()	Event, Match
TPRO_setYear()	Event, Match

3.3 TPRO API — Routine Descriptions

3.3.1 TPRO_open

```
*
* DESCRIPTION:  This routine allocates a TPRO_BoardObj object, sets
*               a handle to the board, and retrieves both the PCI
*               SubSystem Product ID and PCI device ID.
*
* ARGUMENTS:    Pointer to pointer of TPRO_BoardObj handle
*               Device name (e.g. tpro<x> )
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_OBJECT_ERR - memory allocation error
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
* NOTES:        It no longer retrieves the board firmware and
*               FPGA revision information. The caller is responsible
*               for closing the board (via TPRO_close) only if the
*               return from here is successful (TPRO_SUCCESS).
*
------------------------------------------------------------------------------

unsigned char TPRO_open(TPRO_BoardObj **hnd, char *deviceName);
```

3.3.2 TPRO_close

```
*
* DESCRIPTION:  This routine closes the device bound to the TPRO_BoardObj
*               object and frees the allocated TPRO_BoardObj object.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*
* RETURNS:      TPRO_SUCCESS
*               TPRO_DEVICE_NOT_OPEN
*
------------------------------------------------------------------------------

unsigned char TPRO_close(TPRO_BoardObj *hnd);
```

3.3.3 TPRO_getAltitude

```
*
* DESCRIPTION:  This routine retrieves the altitude information from
*               a TPRO TSAT board.  The altitude is in meters.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_AltObj object
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid arg pointers
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
------------------------------------------------------------------------------


unsigned char TPRO_getAltitude(TPRO_BoardObj *hnd, TPRO_AltObj *Altp);
```

3.3.4 TPRO_getDate

```
*
* DESCRIPTION:  This routine retrieves the current date (Gregorian format)
*               from the TPRO board.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_DateObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
-------------------------------------------------------------------------------

unsigned char TPRO_getDate(TPRO_BoardObj *hnd, TPRO_DateObj *Datep);
```

3.3.5 TPRO_getDriver

```
*
* DESCRIPTION:  This routine will return the drivers compiled version number.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to 'version' string
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_getDriver(TPRO_BoardObj *hnd, char *version);
```

3.3.6 TPRO_getFirmware

```
*
* DESCRIPTION:  This diagnostic routine returns the firmware version for
*               the TPRO board. The buffer returned is packed with the
*               nibbles of each byte corresponding to two characters in the
*               range: F-A, 0-9.   For example:
*               The firmware version 0x12 0x16 0x42 0x23
*               would be interpreted as the version string: "12164223"
*               It is up to the caller to interpret the buffer correctly.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to firmware version byte buffer
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_getFirmware(TPRO_BoardObj *hnd, char *firmware);
```

3.3.7 TPRO_getFPGA

```
*
* DESCRIPTION:  This routine retrieves the FPGA version from the TPRO board
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to FPGA version byte buffer
*
```

```
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
-------------------------------------------------------------------------------

unsigned char TPRO_getFPGA(TPRO_BoardObj *hnd, unsigned char *fpga);
```

3.3.8 TPRO_getLatitude

```
*
* DESCRIPTION:  This routine retrieves the latitude information from
*               the TPRO TSAT board.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_LatObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
-------------------------------------------------------------------------------

unsigned char TPRO_getLatitude(TPRO_BoardObj *hnd, TPRO_LatObj *Latp);
```

3.3.9 TPRO_getLongitude

```
*
* DESCRIPTION:  This routine retrieves the longitude information from
*               the TPRO TSAT board.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_LatObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
*******************************************************************/
unsigned char TPRO_getLongitude(TPRO_BoardObj *hnd, TPRO_LongObj *Longp);
```

3.3.10 TPRO_getSatInfo

```
*
* DESCRIPTION:  This routine retrieves the number of satellites the TPRO
*               TSAT board is tracking.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_SatObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
-------------------------------------------------------------------------------

unsigned char TPRO_getSatInfo(TPRO_BoardObj *hnd, TPRO_SatObj *Satp);
```

3.3.11 TPRO_getTime

```
*
* DESCRIPTION:  This routine retrieves the current time from the
*               TPRO board. ( day is in Julian format )
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_TimeObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_getTime(TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);
```

3.3.12 TPRO_resetFirmware

```
*
* DESCRIPTION:  This routine resets the firmware programmed on the TPRO
*               board. It is for troubleshooting purposes only and
*               should not be used in the main application. Once
*               the reset occurs, this routine will block for 10 seconds
*               allowing the reset to fully complete. This routine
*               will return busy if there is an outstanding
*               TPRO_wait<xxx> request (any type).
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid ptr to board
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_DEV_BUSY   - device busy performing a
*                                 TPRO_wait[Event|Match|Heartbeat]
*
-------------------------------------------------------------------------------

unsigned char TPRO_resetFirmware(TPRO_BoardObj *hnd);
```

3.3.13 TPRO_setHeartbeat

```
*
* DESCRIPTION:  This routine is used to control the heartbeat output. It
*               sets the frequency, signal type and output type expected
*               from the TPRO board. The routine will return busy if there
*               is an outstanding TPRO_waitHeartbeat request.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_HeartObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_FREQ_ERR   - invalid frequency specified
*               TPRO_HANDLE_ERR - invalid ptr to function args
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_DEV_BUSY   - device busy performing a
*                                 TPRO_waitHeartbeat
*
-------------------------------------------------------------------------------

unsigned char TPRO_setHeartbeat(TPRO_BoardObj *hnd, TPRO_HeartObj *Heartp);
```

3.3.14 TPRO_setMatchTime

```
*
* DESCRIPTION:  This routine is used to specify the time (which when reached)
*               the TPRO board will drive the match output line high or low
*               based on the match-type. This routine will return a busy
*               indication if there is an outstanding TPRO_waitMatch request.
*
*               m-type    line direction
*               ------    --------------
*               START        high
*               STOP         low
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_MatchObj object
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid ptr to func args
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_DAY_PARM_ERR   - bad day (valid 0-366)
*               TPRO_HOUR_PARM_ERR  - bad hour (valid 0-23)
*               TPRO_MIN_PARM_ERR   - bad minute (valid 0-59)
*               TPRO_SEC_PARM_ERR   - bad seconds (valid 0-60)
*               TPRO_DEV_BUSY       - device busy performing a
*                                        TPRO_waitMatch

*
-------------------------------------------------------------------------------

unsigned char TPRO_setMatchTime(TPRO_BoardObj *hnd, TPRO_MatchObj *Matchp);
```

3.3.15 TPRO_setOscillator

```
*
* DESCRIPTION:  This routine is used to specify the oscillator frequency of
*               a TPRO (PMC/cPCI type only) board.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to frequency value
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_FREQ_ERR   - invalid frequency requested
*               TPRO_COMM_ERR   - error interacting with device
*               TPRO_INVALID_BOARD_TYPE_ERR
*
-------------------------------------------------------------------------------

unsigned char TPRO_setOscillator(TPRO_BoardObj *hnd, unsigned char *freq);
```

3.3.16 TPRO_setPropDelayCorr

```
*
* DESCRIPTION:  This routine sets the propagation delay correction factor
*               on the TPRO board. The delay specified in micro-seconds.
*               This routine will return busy if there is an outstanding
*               TPRO_wait<xxx> request (of any type).
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to micro-seconds value
*
*               TPRO_COMM_ERR

* RETURNS:      TPRO_SUCCESS        - success
```

```
*               TPRO_HANDLE_ERR     - invalid ptr to func args
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_DELAY_PARM_ERR - invalid delay values specified
*               TPRO_DEV_BUSY       - device busy performing a
*                                     TPRO_wait[Event|Match|Heartbeat]
*
------------------------------------------------------------------------------

unsigned char TPRO_setPropDelayCorr(TPRO_BoardObj *hnd, int *uSec);
```

3.3.17 TPRO_setTime

```
*
* DESCRIPTION:  This routine will specify the time on the on-board clock of
*               the TPRO board (Julian date). This routine will not succeed
*               if there is an outstanding TPRO_wait<xxx> request for match
*               or time events.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_TimeObj object
*
* NOTES:        PMC/cPCI boards unsupported - No driver support.
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid ptr to function args
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_DAY_PARM_ERR   - bad day (valid 0-366)
*               TPRO_HOUR_PARM_ERR  - bad hour (valid 0-23)
*               TPRO_MIN_PARM_ERR   - bad minute (valid 0-59)
*               TPRO_SEC_PARM_ERR   - bad seconds (valid 0-59)
*               TPRO_INVALID_BOARD_TYPE_ERR
*               TPRO_DEV_BUSY       - device busy performing a
*                                     TPRO_wait[Event|Match]
*
------------------------------------------------------------------------------

unsigned char TPRO_setTime(TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);
```

3.3.18 TPRO_setYear

```
*
* DESCRIPTION:  This routine is used to set the year on the TPRO board.
*               This call will return busy if there is an outstanding
*               request for a match or time event.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to the year value
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid ptr to function args
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_YEAR_PARM_ERR  - bad year (valid 1990-2999)
*               TPRO_INVALID_BOARD_TYPE_ERR
*               TPRO_DEV_BUSY       - device busy performing a
*                                     TPRO_wait[Event|Match]
------------------------------------------------------------------------------

unsigned char TPRO_setYear(TPRO_BoardObj *hnd, unsigned short *yr);
```

3.3.19 TPRO_simEvent

```
*
* DESCRIPTION:  This routine is used to simulate an external time-tag
```

```
*               event on the TPRO board.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_simEvent(TPRO_BoardObj *hnd);
```

3.3.20 TPRO_synchControl

```
*
* DESCRIPTION:  Based on flag specified, this routine tells the TPRO board
*               to synchronize either to an input source or to 'freewheel'.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to enable flag
*                   [ TPRO_DISABLE_SYNC | TPRO_ENABLE_SYNC ]
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid ptr to function args
*               TPRO_COMM_ERR       - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_synchControl(TPRO_BoardObj *hnd, unsigned char *enbp);
```

3.3.21 TPRO_synchStatus

```
*
* DESCRIPTION:  This routine is used to query the synchronization status from
*               the TPRO board. The status is returned in the argument
*               'statusFlag'.
*
* ARGUEMENTS:   Pointer to TPRO_BoardObj handle
*               Pointer to status flag
*                   values              -    meaning
*                   ------                   -------
*               TPRO_SYNC_STAT_FREE         freewheel
*               non-TPRO_SYNC_STAT_FREE     sync to input
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
-------------------------------------------------------------------------------

unsigned char TPRO_synchStatus(TPRO_BoardObj *hnd, unsigned char *status);
```

3.3.22 TPRO_waitEvent

```
*
* DESCRIPTION:  This routine is used to report the time of arrival
*               of an external time-tagged event generated  from the TPRO
*               board. If the event does not occur within the supplied time-out
*               period, a TPRO_TIMEOUT_ERR is returned. The time-out
*               is specified in clock-ticks (system configuration dependent –
*               typically 100/sec).
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_WaitObj object
```

```
*                   Supply timeout when calling this method.
*                   Event time returned in waitp
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid arg pointers
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_TIMEOUT_ERR    - no event at timer expiration
*
-------------------------------------------------------------------------------

unsigned char TPRO_waitEvent(TPRO_BoardObj *hnd, TPRO_WaitObj *waitp);
```

3.3.23 TPRO_waitHeartbeat

```
*
* DESCRIPTION:  This routine is used to report the status of the heartbeat
*               interrupt. If the event does not occur within a
*               supplied time-out value, a TPRO_TIMEOUT_ERR is returned.
*               The time-out is specified in clock-ticks
*               (system configuration dependent - typically 100/sec).
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to time-out value in clock-ticks.
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid arg pointers
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_TIMEOUT_ERR    - no event at timer expiration
*
-------------------------------------------------------------------------------

unsigned char TPRO_waitHeartbeat(TPRO_BoardObj *hnd, unsigned int *ticks);
```

3.3.24 TPRO_waitMatch

```
*
* DESCRIPTION:  This routine is used to report the status of matching a
*               time-interrupt that was previously set via the
*               TPRO_setMatchTime() API. If the event does not occur
*               within a supplied time-out value, a TPRO_TIMEOUT_ERR
*               is returned. The time-out is specified in
*               clock-ticks (system configuration dependent -
*               typicaly 100/sec)
*l
* ARGUEMENTS:   Pointer to TPRO_BoardObj handle
*               Pointer to time-out value in clock-ticks.
*
* RETURNS:      TPRO_SUCCESS        - success
*               TPRO_HANDLE_ERR     - invalid arg pointers
*               TPRO_COMM_ERR       - error interacting with device
*               TPRO_TIMEOUT_ERR    - no event at timer expiration
*
-------------------------------------------------------------------------------

unsigned char TPRO_waitMatch(TPRO_BoardObj *hnd, unsigned int *ticks);
```

3.3.25 TPRO_peek

```
*
* DESCRIPTION:  A diagnostic routine used to read the value of a register on
*               the TPRO board. The routine assumes the caller knows what each
*               register offset refers to.  The user is only protected from
*               requesting a register offset that exceeds the largest used.
*
```

```
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_MemObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
--------------------------------------------------------------------------------

unsigned char TPRO_peek(TPRO_BoardObj *hnd, TPRO_MemObj *pMem);
```

3.3.26 TPRO_poke

```
*
* DESCRIPTION:  A diagnostic routine used to write a value to a register on
*               the TPRO board. The routine assumes the caller knows what each
*               register offset refers to.  The user is only protected from
*               requesting a register offset that exceeds the largest used.
*
* ARGUMENTS:    Pointer to TPRO_BoardObj handle
*               Pointer to TPRO_MemObj object
*
* RETURNS:      TPRO_SUCCESS    - success
*               TPRO_HANDLE_ERR - invalid arg pointers
*               TPRO_COMM_ERR   - error interacting with device
*
--------------------------------------------------------------------------------

unsigned char TPRO_poke(TPRO_BoardObj *hnd, TPRO_MemObj *pMem);
```

REVISION HISTORY

Revision Level	*ECN Number*	*Description*
A	**2225**	First draft of Spectracom documentation for this product.
B	**2332**	Minor corrections.

Spectracom Corporation
95 Methodist Hill Drive
Rochester, NY 14623
www.spectracomcorp.com
Phone: US +1.585.321.5800
Fax: US +1.585.321.5219