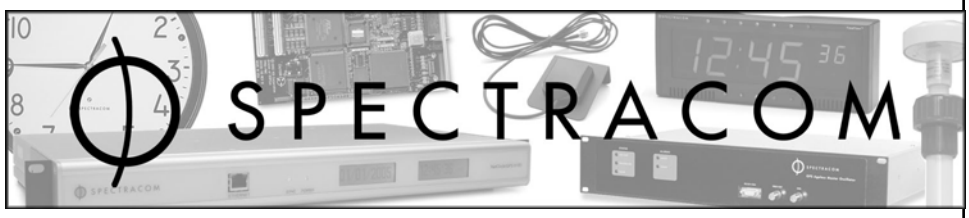
TPRO-PCI-U/TSAT-PCI-U
SYNCHRONIZABLE TIMECODE GENERATOR with UNIVERAL PCI BUS INTERFACE

Linux 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 1186-5003-0050
Manual Revision A
8 July 2008

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

The TPRO/TSAT PCI-U performs timing and synchronization functions referenced to an input timecode signal. The board synchronizes its on-board clock to the incoming timecode. The on-board clock's time is also provided as an IRIG-B output. The board includes a time-tag TTL input, a programmable "heartbeat" pulse or squarewave output (with interrupt capability), and a programmable "match" start/stop time output (with interrupt capability)

The TPRO/TSAT PCI-U continues to increment time ("freewheel") in the absence of an input timecode. Thus, the board can be used as an IRIG-B timecode generator by setting the initial time via the PCI bus.

The input timecode format (IRIG-B, IRIG-A, or NASA36) is detected automatically. Synchronization to the input timecode is also automatic and can be enabled/disabled via the PCI bus. A propagation delay offset may be specified to compensate for cable delays.

The timecode input is an amplitude-modulated sine wave. An automatic gain control (AGC) circuit permits a wide range of input amplitudes. The timecode input is differential; the board does not reference this signal to ground. A single-ended input (referenced to ground) is also acceptable.

The board can be ordered with option "-M" to synchronize to a one-pulse-per-second (1PPS) input instead of an incoming timecode. In this case, the initial time is programmed via the PCI bus, and the board begins counting on the next 1 PPS pulse.

2 Installing the Driver

2.1 CPU and Kernel Support

The driver is designed to operate with 32bit or 64bit Linux kernel versions 2.4 and 2.6 running on a PC system with Intel x86 processor(s). Testing was performed under Redhat 2.6.18 (64 bit Multi-processor/Multi-core), 2.4.22 (32-bit, single processor), 2.6.5 (32-bit, single processor), 2.6.13.1 (32-bit, single processor) and 2.6.18 (64-bit, multiple processors). The driver was also tested under Ubuntu with 2.6.15 kernel (32-bit, single processor).

NOTE: Due to kernel version differences, the driver will need to be built before it is used. You will need *GCC* and *Make* utilities. You will also need the *GNU C Library*.

2.2 To Build and Install the Driver:

1) Open a terminal window.
2) Make sure you are logged in as a root user.
3) Copy the drive file to a convenient directory location and extract the driver.
4) Change to the directory where the driver and its sources were extracted.
5) Build the driver by issuing the commands below:

```
> make clean
> make
> cd driver
> make install
```

NOTE: Due to the differences between the many Linux distributions, some build errors may occur. The most likely cause is an incorrectly installed kernel source. Refer to the documentation for your release of Linux for instructions concerning installing the kernel source.

6) Install the driver by issuing the command:

```
> modprobe tpropci
```

To verify that the driver has been installed, type at the prompt:

```
> lsmod
```

Verify that the driver “tpropci” is present.

2.3 Uninstalling the Driver

To uninstall the driver, issue the following command:

```
> rmmod tpropci
```

2.4 Application Example

With the drivers, there is a folder called “tools” that contains source code with example programs and scripts. These examples demonstrate how to use the API calls to interface to the card. The examples will need to be built before they are used, again due to kernel version differences.

To run a program, type at the prompt:

> ./GetTime <x> *x = PCI card with which you want to interface*

This program will return the current time from the TPROPCI card 0.

3 Interface to the Linux API

3.1 Header File

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

```
/*****************************************************************************
**
**  Module  : tpro.h
**  Date    : 04/02/08
**  Purpose : This is the TPRO-PCI interface include file.
**
**  Copyright(C) 2008 Spectracom Corporation. All Rights Reserved.
**
*****************************************************************************/

#ifndef _defined_TPRO_
#define _defined_TPRO_

/*****************************************************************************
            DEFINES
*****************************************************************************/

/*
**  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 cycle */
#define SIG_JAM          (1)    /* start immediately */

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

/*
**  Oscillator frequencies - for Compact PCI Card 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)

/*
** TPRO BOARD OBJECT
*/
typedef struct TPRO_BoardObj {

  int            file_descriptor;
  unsigned short devid;
  unsigned short options;

} TPRO_BoardObj;



/*
** TPRO ALTITUDE OBJECT
```

```
*/
typedef struct TPRO_AltObj {

  float meters;

} TPRO_AltObj;

/*
** TPRO DATE OBJECT
*/
typedef struct TPRO_DateObj {

  unsigned short year;
  unsigned char  month;
  unsigned char  day;

} TPRO_DateObj;

/*
** TPRO LONGITUDE/LATTITUDE OBJECT
*/
typedef struct TPRO_LongLat {

  unsigned short degrees;
  float          minutes;

} TPRO_LongObj, TPRO_LatObj;

/*
** TPRO MATCH OBJECT
*/
typedef struct TPRO_MatchObj {

  unsigned char   matchType; /* start/stop time */
  double          seconds;
  unsigned char   minutes;
  unsigned char   hours;
  unsigned short  days;

} TPRO_MatchObj;

/*
** TPRO SATINFO OBJECT
*/
typedef struct TPRO_SatObj {

  unsigned char satsTracked; /* num sats tracked */
  unsigned char satsView;    /* num sats 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;
  unsigned char  hours;
  unsigned short days;
  unsigned short year;
  unsigned short flags;      /* bit 15 flagsInvalid(1); bit 2 SYNC, bit 1 TCODE; all others 0 */

} TPRO_TimeObj;

/*
** TPRO WAIT OBJECT
*/
typedef struct TPRO_WaitObj {

  int jiffies;            /*-- # jiffies 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;
  unsigned short value;
  unsigned long  l_value;

} TPRO_MemObj;

/*****************************************************************************
            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 */

/*****************************************************************************
            PUBLIC ROUTINE PROTOTYPES
*****************************************************************************/
unsigned char TPRO_open             (TPRO_BoardObj **hnd, char *deviceName);
unsigned char TPRO_close            (TPRO_BoardObj *hnd);
unsigned char TPRO_getAltitude      (TPRO_BoardObj *hnd, TPRO_AltObj *Altp);
unsigned char TPRO_getDate          (TPRO_BoardObj *hnd, TPRO_DateObj *Datep);
unsigned char TPRO_getDriver        (TPRO_BoardObj *hnd, char *driver);
unsigned char TPRO_getFirmware      (TPRO_BoardObj *hnd, char *firmware);
unsigned char TPRO_getFPGA          (TPRO_BoardObj *hnd, char *fpga);
unsigned char TPRO_getLatitude      (TPRO_BoardObj *hnd, TPRO_LatObj *Latp);
unsigned char TPRO_getLongitude     (TPRO_BoardObj *hnd, TPRO_LongObj *Longp);
unsigned char TPRO_getSatInfo       (TPRO_BoardObj *hnd, TPRO_SatObj *Satp);
unsigned char TPRO_getTime          (TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);
unsigned char TPRO_resetFirmware    (TPRO_BoardObj *hnd);
```

```
unsigned char TPRO_setHeartbeat     (TPRO_BoardObj *hnd, TPRO_HeartObj *Heartp);
unsigned char TPRO_setMatchTime     (TPRO_BoardObj *hnd, TPRO_MatchObj *Matchp);
unsigned char TPRO_setOscillator    (TPRO_BoardObj *hnd, unsigned char *freq);
unsigned char TPRO_setPropDelayCorr (TPRO_BoardObj *hnd, int *us);
unsigned char TPRO_setTime          (TPRO_BoardObj *hnd, TPRO_TimeObj *Timep);
unsigned char TPRO_setYear          (TPRO_BoardObj *hnd, unsigned short *yr);
unsigned char TPRO_simEvent         (TPRO_BoardObj *hnd);
unsigned char TPRO_synchControl     (TPRO_BoardObj *hnd, unsigned char *enbp);
unsigned char TPRO_synchStatus      (TPRO_BoardObj *hnd, unsigned char *status);
unsigned char TPRO_waitEvent        (TPRO_BoardObj *hnd, TPRO_WaitObj *waitp);
unsigned char TPRO_waitHeartbeat    (TPRO_BoardObj *hnd, int *jiffies);
unsigned char TPRO_waitMatch        (TPRO_BoardObj *hnd, int *jiffies);
unsigned char TPRO_peek             (TPRO_BoardObj *hnd, TPRO_MemObj *pMem);
unsigned char TPRO_poke             (TPRO_BoardObj *hnd, TPRO_MemObj *pMem);

/*****************************************************************************
           PUBLIC ROUTINE AVAILABILITY
*****************************************************************************/

/*                                                          Available only to
                       Available when  Available to all users  first user when
Routine                #users = 1      when #users > 1         #users > 1
---------------------  --------------  ----------------------  -----------------
TPRO_open                     Y                  Y
TPRO_close                    Y                  Y
TPRO_getAltitude              Y                                        Y
TPRO_getDate                  Y                                        Y
TPRO_getDriver                Y                  Y
TPRO_getFirmware              Y                                        Y
TPRO_getFPGA                  Y                                        Y
TPRO_getLatitude              Y                                        Y
TPRO_getLongitude             Y                                        Y
TPRO_getSatInfo               Y                                        Y
TPRO_getTime                  Y                  Y
TPRO_resetFirmware            Y                                        Y
TPRO_setHeartbeat             Y                                        Y
TPRO_setMatchTime             Y                                        Y
TPRO_setOscillator            Y                                        Y
TPRO_setPropDelayCorr         Y                                        Y
TPRO_setTime                  Y                                        Y
TPRO_setYear                  Y                                        Y
TPRO_simEvent                 Y                                        Y
TPRO_synchControl             Y                                        Y
TPRO_synchStatus              Y                  Y
TPRO_waitEvent                Y                                        Y
TPRO_waitHeartbeat            Y                                        Y
TPRO_waitMatch                Y                                        Y
TPRO_peek                     Y                  Y
TPRO_poke                     Y                  Y
*/

#endif  // _defined_TPRO_
```

3.2 TPRO API — Routine Descriptions

3.2.1 TPRO_open

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

This routine allocates a TPRO_BoardObj obect, sets a handle to the tpro/tsat, and sets the driver firmware, fpga revision (if applicable), and the driver revision strings.

```
Arguments:  Pointer to TPRO_BoardObj handle
            Device name - "TPROpciX"

Returns:    TPRO_OBJECT_ERR – error allocating board object
            TPRO_HANDLE_ERR – error retrieving handle to device
            TPRO_COMM_ERR   - error communicating with driver
            TPRO_SUCCESS    - success
```

3.2.2 TPRO_close

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

This routine frees the allocated board object and closes the handle to the tpro/tsat device.

```
Arguments:  Pointer to TPRO_BoardObj

Returns:    TPRO_CLOSE_HANDLE_ERR – error closing handle to device
            TPRO_DEVICE_NOT_OPEN  - device is not open
            TPRO_SUCCESS          - success
```

3.2.3 TPRO_getAltitude

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

This routine retrieves the altitude information from the tSAT board. Altitude distance is in meters.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to TPRO_AltObj

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.4 TPRO_getDate

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

This routine retrieves the current date from the TPRO/tSAT board. The date is in Gregorian Format.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to TPRO_DateObj

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.5 TPRO_getLatitude

unsigned char TPRO_getLattitude(TPRO_BoardObj *hnd, TPRO_LatObj *Latp);

This routine retrieves the latitude information from the tSAT device.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to TPRO_LatObj

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.6 TPRO_getLongitude

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

This routine retrieves the longitude information from the /tSAT device.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_LongObj

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.7 TPRO_getSatInfo

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

This routine retrieves the number of satellites tracked from the tSAT device.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_SatObj

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.8 TPRO_getTime

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

This routine retrieves the current time from the TPRO/tSAT device. The seconds value is received as type double.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_TimeObj

Returns:    TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.9 TPRO_resetFirmware

unsigned char TPRO_resetFirmware(TPRO_BoardObj *hnd);

This routine resets the firmware programmed on the TPRO/tSAT device. This function is for troubleshooting purposes only and should not be used in the main application.

```
Arguments:  Pointer to the TPRO_BoardObj

Returns:    TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

3.2.10 TPRO_setHeartbeat

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

This routine controls the hearbeat output. The heartbeat output may be a square wave or pulse at various frequencies.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_HeartObj

Returns:    TPRO_FREQ_ERR – invalid frequency value
            TPRO_COMM_ERR - error communicating with driver
            TPRO_SUCCESS  - success
```

3.2.11 TPRO_setMatchTime

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

This routine drives the match output line high (start time) or low (stop time) when the desired time is met.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_MatchObj

Returns:    TPRO_DAY_PARM_ERR  - invalid days parameter (must be 0-366)
            TPRO_HOUR_PARM_ERR – invalid hours parameter (must be 0 – 23)
            TPRO_MIN_PARM_ERR  - invalid minutes parameter (must be 0 – 59)
            TPRO_SEC_PARM_ERR  - invalid seconds paramter (must be 0 – 69)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

3.2.12 TPRO_setPropDelayCorr

unsigned char TPRO_setPropDelayCorr (TPRO_BoardObj *hnd, int *us);

This routine sets the propagation delay correction factor.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to correction factor in microseconds

Returns:    TPRO_DELAY_PARM_ERR – invalid propagation delay factor
            TPRO_COMM_ERR       - error communicating with driver
            TPRO_SUCCESS        - success
```

3.2.13 TPRO_setTime

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

This routine sets the time on the on-board clock of the TPRO/tSAT device. If the board is synchronized to a GPS antenna this value will not be accepted.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the TPRO_TimeObj

Returns:    TPRO_DAY_PARM_ERR  - invalid days parameter (must be 0-366)
            TPRO_HOUR_PARM_ERR – invalid hours parameter (must be 0 – 23)
            TPRO_MIN_PARM_ERR  - invalid minutes parameter (must be 0 – 59)
            TPRO_SEC_PARM_ERR  - invalid seconds paramter (must be 0 – 69)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

3.2.14 TPRO_setYear

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

This routine programs the TPRO/tSAT device with the desired year. If the board is synchronized to a GPS antenna this value will not be accepted.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the desired year

Returns:    TPRO_INVALID_BOARD_TYPE_ERR – invalid board type for function
            TPRO_COMM_ERR               - error communicating with driver
            TPRO_SUCCESS                - success
```

REVISION HISTORY

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

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