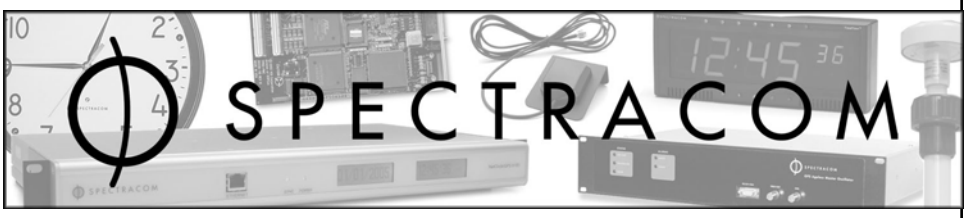
TPRO-cPCI, PMC/TSAT-cPCI, PMC
SYNCHRONIZABLE TIMECODE GENERATOR with COMPACT PCI BUS INTERFACE

Windows 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 1152-5002-0050
Manual Revision A
28 April 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 Windows Driver for the Spectracom TPRO/TSAT cPCI and PMC boards provides the interface for multiple users to access the board using the API documented in Chapter Three. It is compatible with the following versions of Windows:

- Microsoft Windows Vista (x86, x64, AMD64)
- Microsoft Windows Server 2003 SP1 (x86, x64, AMD64)
- Windows XP x64 Edition
- Windows XP Family (including Service Pack 2)
- Windows 2000 Family

The TPRO-cPCI and PMC provide high-accuracy timing functions on a plug-in board for the CompactPCI computer bus. The board has an on-board clock, which is kept in sync to an external time code input. Several timing functions are derived from the on-board clock, including a programmable periodic pulse rate output (“heartbeat”), a programmable start-stop output (“Match”), a selectable frequency output (“oscillator out”, 1 kHz, 1, 5, or 10 MHz), and a time-stamping input (“Time-Tag”).

The TPRO-cPCI, PMC obtains time from an input time code, which can be in either IRIG-B or IRIG-A format–-the board automatically detects which format is being used. Timing accuracy is the same regardless of which format is being used. The time code conveys the day, hour, minute, and second. The on-board 10 MHz oscillator is disciplined to maintain a 1-microsecond accuracy. An IRIG-B time code output is provided, which is in sync with the incoming time code.

The TPRO-cPCI, PMC may be used as a stand-alone time code generator. The computer programs the day, hour, minute, and second. The board then continues to count from that time, using the on-board oscillator as the timebase reference. This is called “freewheeling”.

The host computer communicates to the board through a set of memory-mapped registers. When the computer boots up, the board identifies itself to the Compact PCI bus by specifying the unique Subsystem Vendor ID and Subsystem Device ID. The host computer can then read the instantaneous time, and command the board to set time, and/or to provide an interrupt at a periodic rate, at a specified time, and/or when a time-tag event occurs.

2 Installing the Driver

To install the driver, perform the following steps:

1. Install TPRO/TSAT compact PCI card in a vacant slot of desired compact PCI chassis.

2. Apply power to the compact PCI computer. When the Plug and Play Manager finds new hardware, choose “Have Disk” to install the driver from the CD. Select the “TPRO.INF” file from the CD and follow prompts to finish installation.

3. Double-click on the setup.exe file from the CD distribution to launch the setup utility.

4. Follow the on-screen prompts, making sure to accept defaults. The utility commences copying application files. When this process is finished, the control utility installation is complete.

5. Click the “Finish” button on the screen.

2.1 Executing the TPRO/TSAT Control Utility

1. From the start Menu, select the “Programs” folder.
2. Select the “KSI” folder.
3. Select the “TPRO-TSAT Control Utility” program.

2.2 Application Example

```
/*****************************************************************************
  MAIN.C
*****************************************************************************/

#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <winioctl.h>

#include <tpro.h>


/*****************************************************************************
  MAIN
*****************************************************************************/

int main (void)
{
  TPRO_BoardObj *hnd;


  /**
  ***  Get firmware revision
  **/
```

```
  printf ("\nGetting firmware revision...\n");
  {
    unsigned char tproFirmware[10];
    unsigned char k;

    TPRO_getFirmware (hnd, tproFirmware);

    for (k=0; k < 4; k++) {
      printf ("0x%02X\n", tproFirmware[k]);
    }
  }



 /**
  ***  set year
  **/
  printf ("\nSetting the year...\n");
  {
    unsigned short year = 2001;

    TPRO_setYear (hnd, &year);
  }


  /**
  ***  set time
  **/

  printf ("\nSetting time...\n");
  {
    TPRO_TimeObj tproTime;

    tproTime.seconds = 00;
    tproTime.minutes = 59;
    tproTime.hours   = 12;
    tproTime.days    = 97;

    TPRO_setTime (hnd, &tproTime);
  }


  /**
  ***  get time
  **/
  printf ("\nGetting time...\n");
  {
    TPRO_TimeObj tproTime;

    TPRO_getTime (hnd, &tproTime);

    printf ("Days: %d\n", tproTime.days);
    printf ("Hours: %d\n", tproTime.hours);
    printf ("Minutes: %d\n", tproTime.minutes);
    printf ("Seconds: %2.6lf\n", tproTime.secsDouble);
  }


  /**
  ***  get gregorian date
  **/
  printf ("\nGetting Gregorian Date...\n");
  {
```

```
    TPRO_DateObj tproDate;

    TPRO_getDate (hnd, &tproDate);
    printf ("%02d-%02d-%04d\n", tproDate.month, tproDate.day, tproDate.year);
  }


  TPRO_close (hnd);
  return (0);
```

3 Interface to the Windows API

3.1 Header File

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

```
/*************************************************************************
  DSPCon TPRO/tSAT - Interface Header

              DSPCon, Inc.
              380 Foothill Road
              Bridgewater, NJ 08807

              e-mail: info@dspcon.com

  (C) Copyright 2001 DSPCon, Inc.  All rights reserved.
  Use of copyright notice is precautionary and does not imply publication
*************************************************************************/

/*************************************************************************
  TPRO.H
*************************************************************************/

#ifndef _defined_TPRO_
#define _defined_TPRO_

#pragma pack(8)

#ifdef __cplusplus
extern "C" {
#endif


#define DLL_EXPORT  __declspec(dllexport)



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


/**
***  Heartbeat constants
**/

#define SIG_PULSE     (0xE5)         // signal type for PCI card
#define SIG_SQUARE    (0xE7)         // signal type for PCI card

#define SIG_NO_JAM    (0)            // output type for PCI card
#define SIG_JAM       (1)            // output type for PCI card

#define HEART_NORMAL  (0)            // signal type for CPCI card
#define HEART_INVERT  (8)            // signal type for CPCI card

#define HEART_DISABLE (0)            // output type for CPCI card
#define HEART_ENABLE  (4)            // output type for CPCI card

#define CLK_10MHZ     (0)            // clock frequency for CPCI board
#define CLK_3MHZ      (1)            // clock frequency for CPCI board
#define CLK_1MHZ      (2)            // clock frequency for CPCI board
#define CLK_1KHZ      (3)            // clock frequency for CPCI board
```

```
/**
***  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)


/*****************************************************************************
                              OBJECTS
*****************************************************************************/


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

typedef struct TPRO_BoardObj
{ /*-------------------------------------------------------------------------*/
  void *hDevice;

  unsigned short options;
  unsigned char slotPosition;
} /*-------------------------------------------------------------------------*/
TPRO_BoardObj;


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

typedef struct TPRO_AltObj
{ /*-------------------------------------------------------------------------*/
  float         meters;                     /*-- 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;                /*-- num sats tracked ----*/
  unsigned char satsView;                   /*-- num sats in view ----*/
} /*-----------------------------------------------------------------------*/
TPRO_SatObj;


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

typedef struct TPRO_HeartObj
{ /*-----------------------------------------------------------------------*/
  unsigned char signalType;                 /*-- heart signal type ---*/
  unsigned char outputType;                 /*-- jamming option ------*/

  unsigned char clockFreq;                  /*-- clock freq for CPCI -*/

  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 year;                      /*-- year for CPCI board -*/
} /*-----------------------------------------------------------------------*/
TPRO_TimeObj;


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

typedef struct TPRO_WaitObj
{ /*-----------------------------------------------------------------------*/
  unsigned int ticks;                       /*-- # ticks to wait -----*/
```

```
  double  seconds;                          /*-- seconds -------------*/
  unsigned char  minutes;                   /*-- minutes -------------*/
  unsigned char  hours;                     /*-- hours ---------------*/
  unsigned short days;                      /*-- days ----------------*/

  unsigned short year;                      /*-- year for cPCI card --*/
  unsigned char  month;                     /*-- month for cPCI card -*/
  unsigned char  day;                       /*-- day for cPCI card ---*/
} /*-------------------------------------------------------------------*/
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 creating handle to device
#define TPRO_OBJECT_ERR             (2)    // error creating device object
#define TPRO_CLOSE_HANDLE_ERR       (3)    // error closing device handle
#define TPRO_DEVICE_NOT_OPEN_ERR    (4)    // tpro device was not opened
#define TPRO_INVALID_BOARD_TYPE_ERR (5)    // function is not available for board type
#define TPRO_FREQ_ERR               (6)    // invalid frequency
#define TPRO_YEAR_PARM_ERR          (7)     // invalid year parameter
#define TPRO_DAY_PARM_ERR           (8)     // invalid day parameter
#define TPRO_HOUR_PARM_ERR          (9)     // invalid hour parameter
#define TPRO_MIN_PARM_ERR           (10)     // invalid minutes parameter
#define TPRO_SEC_PARM_ERR           (11)     // invalid seconds parameter
#define TPRO_DELAY_PARM_ERR         (12)    // invalid delay factor
#define TPRO_TIMEOUT_ERR            (13)     // device timed out
#define TPRO_COMM_ERR               (14)    // error communicating with driver


/*********************************************************************
                    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_flushFIFO      (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 *driver);

DLL_EXPORT
```

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

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

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 *Mem);

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


#ifdef __cplusplus
}
#endif

#pragma pack()

#endif  // _defined_TPRO_
```

3.2 TPRO API — Routine Descriptions

The TPRO-cPCI and PMC driver permits overlapping use of the TPRO-waitXXX routines and other device access routines. However, it should be noted that simultaneous access to the device requires multiple open device handles. That is, if an application requires access to the device driver from two different threads, then each thread must have its own device handle.

3.2.1.1.1 TPRO_open

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

This routine allocates a TPRO_BoardObj object, sets a handle to the tpro/tsat, and initializes the card.

```
Arguments:  Pointer to TPRO_BoardObj handle
            Device name - "TPROcpci0"

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
```

3.2.1.1.2 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
```

3.2.1.1.3 TPRO_flushFIFO

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

This routine is not applicable to the compact PCI card as it does NOT have a FIFO.

3.2.1.1.4 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
```

3.2.1.1.5 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
```

3.2.1.1.6 TPRO_getDriver

unsigned char TPRO_getDriver (TPRO_BoardObj *hnd, char *driver);

Retrieves the driver version information into the driver buffer.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to zero padded allocated buffer at least 10 bytes

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

3.2.1.1.7 TPRO_getFirmware

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

Retrieves the firmware version information into the firmware buffer.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to zero padded allocated buffer at least 10 bytes

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

3.2.1.1.8 TPRO_getFPGA

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

Retrieves the FPGA version information into the fpga buffer.

```
Arguments:  Pointer to TPRO_BoardObj
            Pointer to zero padded allocated buffer at least 10 bytes

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

3.2.1.1.9 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
```

3.2.1.1.10 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
```

3.2.1.1.11 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
```

3.2.1.1.12 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
```

3.2.1.1.13 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
```

3.2.1.1.14 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
```

3.2.1.1.15 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 parameter (must be 0 – 69)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

3.2.1.1.16 TPRO_setOscillator

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

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 frequency value

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

3.2.1.1.17 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
```

3.2.1.1.18 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 parameter (must be 0 – 69)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

3.2.1.1.19 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
```

3.2.1.1.20 TPRO_simEvent

`unsigned char TPRO_simEvent(TPRO_BoardObj *hnd);`

This routine simulates an external time tag event.

```
Arguments:  Pointer to the TPRO_BoardObj

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

3.2.1.1.21 TPRO_synchControl

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

This routine commands the TPRO/tSAT device to synchronize to input or freewheel. This distinction is made using the enable argument. If the enable argument is (0) the clock will freewheel, otherwise it will synchronize to input. When disabling synchronization (freewheeling), the device will continue to synchronize until the time is set.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the synch enable

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

3.2.1.1.22 TPRO_synchStatus

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

This routine reports the synchronization status of the TPRO/tSAT device. When status is equal to zero, the device is freewheeling. Otherwise the device is synchronized to its input.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to the synch status variable

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

3.2.1.1.23 TPRO_waitEvent

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

This routine will report the time an external event was detected on the Time Tag Input pin. The routine will block for a given number of ticks (in milliseconds) until an event occurs or the timeout period has been reached.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to wait time

Returns:    TPRO_TIMEOUT_ERR – routine has timed-out
            TPRO_COMM_ERR    - error communicating with driver
            TPRO_SUCCESS     - success
```

3.2.1.1.24 TPRO_waitHeartbeat

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

This routine will reports the condition of the heartbeat output. The routine will block for a given number of ticks (in milliseconds) until a heartbeat occurs or the timeout period has been reached.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to timeout variable in milliseconds

Returns:    TPRO_TIMEOUT_ERR – routine has timed-out
            TPRO_COMM_ERR    - error communicating with driver
            TPRO_SUCCESS     - success
```

3.2.1.1.25 TPRO_waitMatch

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

This routine will report the condition of the match start time. The routine will block for a given number of ticks (in milliseconds) until a match start time occurs or the timeout period has been reached.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to wait time

Returns:    TPRO_TIMEOUT_ERR – routine has timed-out
            TPRO_COMM_ERR    - error communicating with driver
            TPRO_SUCCESS     - success
```

3.2.1.1.26 TPRO_peek

unsigned char TPRO_peek(TPRO_BoardObj *hnd, TPRO_MemObj *Mem);

This is a diagnostic routine for the user to read the registers on the TPRO/TSAT card.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to Memory Object

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

3.2.1.1.27 TPRO_poke

unsigned char TPRO_poke(TPRO_BoardObj *hnd, TPRO_MemObj *Mem);

This is a diagnostic routine for the user to write to registers on the TPRO/TSAT card.

```
Arguments:  Pointer to the TPRO_BoardObj
            Pointer to Memory Object

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

REVISION HISTORY

Revision Level	*ECN Number*	*Description*
A	**2187**	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