Application Programmer's Guide

Windows NT

for TPRO/TSATPRO-PC

Edition 1.0

KSI

a division of DSP*Con*, Inc.

For technical support, contact John Bodino at 888-377-2668

TPRO/TSATPRO-PC APPLICATION PROGRAMMERS GUIDE
Windows NT
Edition 1.0 —October 2002

A Publication of
KSI

REVISION HISTORY

Revision Date	Revision No.	Revised Section(s)	Comments/Notes
October 2002	——	——	First Edition

Table of Contents

Chapter One

Overview

This guide provides comprehensive information on the
Windows NT Driver for the KSI TPRO/TSATPRO-PC board.

Introduction

The Driver for the KSI TPRO/TSATPRO-PC Board provides functionality to the board's interfaces and devices.

Product Description

The TPRO–PC is a precision clock that automatically synchronizes to standardized time code signals or (for TSATPRO–PC configuration) to the GPS satellite system and can be read from a host PC bus processor. It is used in 16 and 32-bit ISA bus systems for time tagging. Time tagging can be caused by reading four 16-bit time registers or by a logic pulse from the outside world (an "external event"). The 16-bit reads are usually used for software initiated time tagging, (for example time-tagging the time a block of data transfer starts or completes). Reading the first 16-bit time register (for units of microseconds through units of milliseconds) also freezes the tens of milliseconds through hundreds of days.

External events are usually used for time-tagging hardware related events (for example the exact time of a radar transmit pulse) because the added error of variation in software delays degrades the accuracy of a software initiated time tag. The time tag data for external events is transferred to the host through a byte-wide hardware FIFO as a sequence of 10 bytes.

Inputs to the TPRO-PC are modulated time code (or GPS receiver signals for TSATPRO–PC), host commands (not usually needed) and external event pulses as required for the application.

Outputs are modulated IRIG-B time code, and a "heartbeat" pulse rate that can be specified by the user program. The TPRO-PC also can generate interrupts to the host system as enabled and selected by the host system. Interrupt sources include the heartbeat, external event data FIFO not empty, and start or stop time match.

The clock will automatically synchronize to specified time code signals. Status bits advise the host of synchronization status. If there is no synchronization source the TPRO-PC will start counting at 000 days/00 seconds at power-on. The clock time can be set by user commands.

Caveats

Caution

*If your system is equipped with the TPRO-PC 450-10 software driver, it **must be uninstalled** before installing the new, updated driver.*

To uninstall the driver:

1. *Go to "Control Panel", "Add/Remove Programs".*
2. *Remove the TPRO-TSAT PC DEVELOPERS KIT.*
3. *Perform a "find file" for **450-7** and remove all.*
4. *Perform a second "find file" for **TPRO** and remove all.*
5. *Install the new drivers as indicated in this manual.*

This page intentionally left blank.

Chapter Two

Installation and Application

Installing the TPRO/TSAT-PC Windows NT

Step One

Configure the jumpers on the TPRO/TSAT-PC device according to the hardware manual for the desired base address and IRQ. Install TPRO/TSAT-PC card in a vacant ISA slot of desired Personal Computer.

Step Four

Browse to the distribution CD, and execute the **setup.exe** program to install the device driver, development files, and control utility.

Step Four

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 along with development files are installed.

Step Five

Click the “Finish” button on the screen.

Executing the TPRO/TSAT Control Utility

Step One
From the start Menu select the “Programs” folder.

Step Two
Select the “KSI” folder.

Step Three
Select the “TPRO-TSAT Control Utility” program.

Application Example

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

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

#include <tpro.h>


/*************************************************************************
  main entry point
*************************************************************************/

int main (int argc, char *argv[])
{
  unsigned char rv;
  TPRO_BoardObj *hBoard;


  /**
  ***  open tpro device
  **/

  if (rv = TPRO_open (&hBoard, "TPROisa0"), rv != TPRO_SUCCESS) {
    printf ("Could not open Device![%d]\n", rv);
    exit (1);
  }
  hBoard->options = 0x9070;   // set our board options


  /**
  ***  get driver version
  **/
  {
    char driver [10] = {0};

    if (rv = TPRO_getDriver (hBoard, driver), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve driver revision![%d]\n", rv);
    }
    else {
      printf (" Driver version %s\n\n", driver);
    }
  }


  /**
  ***  get satellite information
  **/
  {
    TPRO_SatObj TproSat;
    TPRO_AltObj TproAlt;
    TPRO_LatObj TproLat;
    TPRO_LongObj TproLong;

    // altitude
    if (rv = TPRO_getAltitude (hBoard, &TproAlt), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve altitude information![%d]\n", rv);
    }
    else {
      printf (" Altitude: %f\n", TproAlt.meters);
    }

    // latitude
    if (rv = TPRO_getLatitude (hBoard, &TproLat), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve latitude information![%d]\n", rv);
```

```
    }
    else {
      printf (" Latitude degrees: %d\t\tLatitude minutes: %f\n",
              TproLat.degrees, TproLat.minutes);
    }

    // longitude
    if (rv = TPRO_getLongitude (hBoard, &TproLong), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve longitude information![%d]\n", rv);
    }
    else {
      printf (" Longitude degrees: %d\t\tLongitude minutes: %f\n",
              TproLong.degrees, TproLong.minutes);
    }

    // satellites viewed
    if (rv = TPRO_getSatInfo (hBoard, &TproSat), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve satellite information![%d]\n", rv);
    }
    else {
      printf (" Satellites tracked: %d", TproSat.satsTracked);
      printf ("\t\tNumber of satellites in view: %d\n\n", TproSat.satsView);
    }
  }

  /**
  ***  get time
  **/
  {
    TPRO_TimeObj TproTime;

    if (rv = TPRO_getTime (hBoard, &TproTime), rv != TPRO_SUCCESS) {
      printf ("Could not retrieve time![%d]\n", rv);
    }
    else {
      printf (" Time: %03d:%02d:%02d:%f\n\n", TproTime.days, TproTime.hours,
              TproTime.minutes, TproTime.secsDouble);
    }
  }


  /**
  ***  get status register
  **/
  {
    TPRO_MemObj TproMem;

    TproMem.offset = 0x1;

    if (rv = TPRO_peek (hBoard, &TproMem), rv != TPRO_SUCCESS) {
      printf ("Could not read status register![%d]\n", rv);
    }
    else {
      printf (" Status Register: 0x%02X\n", TproMem.value);
    }
  }


  /**
  ***  close device
  **/

  if (rv = TPRO_close (hBoard), rv != TPRO_SUCCESS) {
    printf ("Could not close board [%d]!\n", rv);
  }

  return (0);
}
```

This page intentionally left blank.

Chapter Three

The TPRO/TSAT-PC Windows NT Driver

Header File

The following is the “TPRO.H” Driver 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_
```

Support Routine Descriptions

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-PC device

Note: After the TPRO_open routine has been called, it is mandatory to set the options element in the TPRO_BoardObj informing the driver as to which type of device it is acting on. Use the table below to set board options.

TPRO/TSAT Board Option	TPRO_BoardObj option element value
TPRO-PC Standard	0x9050
TPRO-PC HB1PPS	0x9052
TSAT-PC Standard	0x9070
TSAT-PC HB1PPS	0x9072

```
Arguments:  Pointer to TPRO_BoardObj handle
            Device name - "TPROisa0"

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
```

TPRO_close

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

This routine frees the allocated board object and closes the handle to the TPRO/TSAT-PC 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
```

TPRO_flushFIFO

unsigned char TPRO_flushFIFO (TPRO_BoardObj *hnd);

This routine flushes the onboard FIFO.

```
Arguments:  Pointer to TPRO_BoardObj

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

TPRO_getAltitude

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

This routine retrieves the altitude information from the TSAT-PC device. 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
```

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
```

TPRO_getLatitude

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

This routine retrieves the latitude information from the TSAT-PC 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
```

TPRO_getLongitude

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

This routine retrieves the longitude information from the TSAT-PC 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
```

TPRO_getSatInfo

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

This routine retrieves the number of satellites tracked from the TSAT-PC device and the number of satellites in view.

```
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
```

TPRO_getTime

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

This routine retrieves the current time from the TPRO/TSAT-PC 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
```

TPRO_resetFirmware

unsigned char TPRO_resetFirmware(TPRO_BoardObj *hnd);

This routine resets the firmware programmed on the TPRO/TSAT-PC 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
```

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
```

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 - 59)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

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
```

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-PC 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 - 59)
            TPRO_COMM_ERR      - error communicating with driver
            TPRO_SUCCESS       - success
```

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
```

TPRO_synchControl

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

This routine commands the TPRO/TSAT-PC 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
```

TPRO_synchStatus

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

This routine reports the synchronization status of the TPRO/TSAT-PC 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
```

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
```

TPRO_waitHeartbeat

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

This routine 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
```

TPRO_waitMatch

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

This routine reports 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
```

TPRO_peek

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

This is a diagnostic routine for the user to read a byte register on the TPRO/TSAT-PC card.

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

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

TPRO_poke

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

This is a diagnostic routine for the user to write a byte to a register on the TPRO/TSAT-PC card.

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

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

This page intentionally left blank.

Warranty

KSI warrants that all products manufactured by KSI conform to published DSP*Con* specifications and are free from defects in materials and workmanship for a period of one (1) year from the date of delivery when used under normal conditions and within the service conditions for which they were furnished.

The obligation upon KSI arising from a warranty claim shall be limited to repairing, or, at its option, replacing without charge, any product which, in KSI's sole opinion, proves to be defective within the scope of this warranty.

KSI must be notified in writing of the defect or nonconformity within the warranty period, and the affected product must be returned to KSI within thirty (30) days after discovery of such defect or nonconformity.

The buyer shall prepay shipping charges, taxes, duties and insurance for products returned to KSI for warranty service. KSI shall pay for the return of products to buyer except for products returned to another country or from outside the forty-eight contiguous United States.

KSI shall have no responsibility for any defect or damage caused by improper installation, unauthorized modification, misuse, neglect, inadequate maintenance, accident or for any product that has been repaired or altered by anyone other than KSI or its authorized representatives.

The warranty described above is the buyer's sole and exclusive remedy and no other warranty, whether oral or written, is expressed or implied KSI specifically disclaims fitness for a particular purpose. Under no circumstances shall KSI be liable for any direct, indirect, special, incidental or consequential damages, expenses, losses or delays (including loss of profits) based on contract, tort, or any legal theory.

This warranty is valid for a period of one (1) year from the date of shipment. Upon warranty expiration, all services on KSI hardware and software are subject to a current hourly rate, plus travel expenses where applicable. As an option, KSI offers extended warranties when purchased within ten (10) days of receipt of product.

Extended Warranties

HARDWARE

KSI offers an extended warranty on its hardware at a cost of 15% of the current list price per year of coverage. This coverage is available for KSI hardware only.

SOFTWARE

KSI offers an extended warranty on its software at a cost of 15% of the current list price per year of coverage. This coverage is available for KSI software only.

Service and Repair

Before returning a product for service and repair, please contact KSI at (866) KSI-KSI3 to obtain a Return Material Authorization (RMA) number. Have available the Model and Serial numbers of the item, a description of the problem, your company name, and the name and address of the person returning the product.

11209 Armour Drive
El Paso, TX 79935

Toll Free Phone: **866-KSI-KSI3**
Fax: **866-593-2080**
Website: **www.ksi-corporation.com**
E-mail: **info@ksi-corporation.com**

For Software Technical Support Contact
Toll Free Phone: 888-DSP-CON8
Fax: 908-722-3259
E-mail: jdb@dspcon.com

A Publication of
KSI