

---

# Application Programmer's Guide

# TPRO/TSAT-PCI

for Windows NT 4.0

Edition 1.1



**KSI**  
a division of DSPCon, Inc.

**For technical support, contact Paulina Domínguez at: (866) 574-5743**

TPRO-PCI APPLICATION PROGRAMMERS GUIDE  
Edition 1.1 —April 2001

A Publication of

**KSI**

© MMI

Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced, stored in or transmitted by a retrieval system, or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, without the written permission of KSI.

KSI has worked to verify the accuracy of the information contained in this document as of its publication date; however, such information is subject to change without notice and KSI is not responsible for any errors that may occur in this document.

Trademarks are acknowledged and are the property of their owners.

**REVISION HISTORY**

<b>Revision Date</b>	<b>Revision No.</b>	<b>Revised Section(s)</b>	<b>Comments/Notes</b>
April 2001	—	—	First Edition
June 27, 2001	1.1		Added error codes and updated installation section

# Table of Contents

<b>CHAPTER ONE—OVERVIEW .....</b>	<b>5</b>
Introduction .....	6
Product Description .....	6
Caveats .....	7
<b>CHAPTER TWO—INSTALLATION &amp; APPLICATION .....</b>	<b>9</b>
Installing the TPRO/TSAT-PCI Win NT Driver .....	10
Executing the TPRO/TSAT Control Utility .....	10
Application Example .....	11
<b>CHAPTER THREE—THE TPRO/TSAT-PCI WIN NT DRIVER .....</b>	<b>13</b>
Header File .....	14
Support Routine Descriptions .....	19
TPRO_open .....	19
TPRO_close .....	19
TPRO_getAltitude .....	19
TPRO_getDate .....	20
TPRO_getLatitude .....	20
TPRO_getLongitude .....	20
TPRO_getSatInfo .....	21
TPRO_getTime .....	21
TPRO_resetFirmware .....	21
TPRO_setHeartbeat .....	22
TPRO_setMatchTime .....	22
TPRO_setPropDelayCorr .....	22
TPRO_setTime .....	23
TPRO_setYear .....	23
TPRO_simEvent .....	23
TPRO_synchControl .....	24
TPRO_synchStatus .....	24
TPRO_waitEvent .....	24
TPRO_waitHeartbeat .....	25
TPRO_waitMatch .....	25

# Chapter One

---

Overview

This guide provides comprehensive information on the Win NT Driver for the KSI TPRO/TSAT-PCI board.

## Introduction

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

## Product Description

The TPRO-PCI 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-PCI 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.

## Caveats

---



### Caution

*If your system is equipped with the TPRO-PCI 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 PCI DEVELOPERS KIT.*
  - 3. Perform a “find file” for **450-10** 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-PCI Win NT Driver

### **Step One**

Go to the CD-ROM drive.

### **Step Two**

Double-click on **setup.exe**. The setup utility launches.

### **Step Three**

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.

### **Step Four**

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
*****/

int main (void)
{
    TPRO_BoardObj *hnd;

    if (TPRO_open (&hnd, "TPROpci0") != TPRO_SUCCESS)
        return (1);

    /**
    *** get date
    **/
    {
        TPRO_DateObj Date;

        if (TPRO_getDate (hnd, &Date) != TPRO_SUCCESS)
            return (1);

        // print results
        printf ("%d\\%d\\%d -- ", Date.month, Date.day, Date.year);
    }

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

        if (TPRO_getTime (hnd, &TproTime) != TPRO_SUCCESS)
            return (1);

        // print results
        printf ("%d:%d:%f\n\n", TproTime.hours, TproTime.minutes,
                TproTime.secsDouble);
    }

    /**

```

```

*** get latitude
**/
{
    TPRO_LatObj Latitude;

    if (TPRO_getLatitude (hnd, &Latitude) != TPRO_SUCCESS)
        return (1);

    // print results
    printf ("Latitude: %d Degrees\t%f minutes\n",
           Latitude.degrees, Latitude.minutes);
}

/**
*** get longitude
**/
{
    TPRO_LatObj Longitude;

    if (TPRO_getLongitude (hnd, &Longitude) != TPRO_SUCCESS)
        return (1);

    // print results
    printf ("Longitude: %d Degrees\t%f minutes\n\n",
           Longitude.degrees, Longitude.minutes);
}

/**
*** get satellite information
**/
{
    TPRO_SatObj SatInfo;

    if (TPRO_getSatInfo (hnd, &SatInfo) != TPRO_SUCCESS)
        return (1);

    // print results
    printf ("Satellites tracked: %d\n", SatInfo.satsTracked);
}

TPRO_close (hnd);

getchar ();
}

```

# Chapter Three

---

The TPRO/TSAT-PCI Win 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_

#ifdef __cplusplus
extern "C" {
#endif

#define DLL_EXPORT __declspec(dllexport)

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

/**
  *** Heartbeat constants
  **/

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

#define SIG_NO_JAM     (0)         // start next 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)

```

```

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

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

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

    unsigned short options;
    unsigned char slotPosition;

    unsigned char firmware[5];
    unsigned char FPGA[5];
    unsigned char driver[7];
} /*-----*/
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;                          /*-- square or pulse ----*/
    unsigned char outputType;                          /*-- jamming option -----*/

    double frequency;                                  /*-- heartbeat freq -----*/
} /*-----*/
TPRO_HeartObj;

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

typedef struct TPRO_TimeObj
{ /*-----*/
    double      secsDouble;                             /*-- seconds floating pt -*/
    unsigned char seconds;                             /*-- seconds whole num ---*/
    unsigned char minutes;                             /*-- minutes -----*/
    unsigned char hours;                               /*-- hours -----*/
    unsigned short days;                               /*-- days -----*/

    unsigned short 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 -----*/
} /*-----*/
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_getAltitude       (TPRO_BoardObj *hnd, TPRO_AltObj *Altp);

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

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

#endif // _defined_TPRO_

```

## Support Routine Descriptions

### 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 sets the driver firmware, fpga revision (if applicable), and the driver revision strings.

Arguments: Pointer to TPRO\_BoardObj handle  
Device name - "TPROpci0"

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 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\_getAltitude

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

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

### **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

### **TPRO\_getLatitude**

```
unsigned char TPRO_getLatitude (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

### **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

### **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

### **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

### **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

### **TPRO\_setHeartbeat**

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

This routine controls the heartbeat 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 - 69)  
 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 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

## **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

## **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 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 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 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

### **TPRO\_waitMatch**

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

This routine will 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

*This page intentionally left blank.*

# *Warranty*

KSI warrants that all products manufactured by KSI conform to published DSPCon 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**

**Toll Free Fax: 866-593-2080**

**Website: [www.ksi-corporation.com](http://www.ksi-corporation.com)**

**E-mail: [info@ksi-corporation.com](mailto:info@ksi-corporation.com)**

**For hardware technical support contact**

**Paulina Domínguez**

**Toll Free Phone: 866-KSI-KSI3**

**Toll Free Fax: 866-593-2080**

**E-mail: [pxd@ksi-corporation.com](mailto:pxd@ksi-corporation.com)**

A Publication of

**KSI**

© MMI

Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced, stored in or transmitted by a retrieval system, or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, without the written permission of KSI.

KSI has worked to verify the accuracy of the information contained in this document as of its publication date; however, such information is subject to change without notice and KSI is not responsible for any errors that may occur in this document.

Trademarks are acknowledged and are the property of their owners.