TSync

PCI TIME CODE PROCESSOR with OPTIONAL GPS

Factory Driver Guide

1565 Jefferson Road
Rochester, NY 14623
Phone: US +1.585.321.5800
Fax: US +1.585.321.5219

spectracom

www.spectracomcorp.com
Part Number 1219-5001-0050
Manual Revision A
December 2013

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 preamplifier are warranted for 5 years, subject to the exceptions listed above.

WARRANTY CLAIMS

Spectracom's obligation under this warranty is limited to infactory 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 1565 Jefferson Road Rochester, NY 14623

+1.585.321.5800 FAX: +1.585.321.5219 www.spectracomcorp.com sales@spectracomcorp.com

Contents

1 Overview

The factory driver for the Spectracom TSync boards provides the interface for multiple users to access the board using the user API library documented herein.

The TSync card with optional GPS is a complete synchronized timecode reader/generator package that supports multiple prioritized timing inputs. When an input is lost, the unit automatically switches to the next input in order of priority.

The disciplined onboard oscillator is phase-locked to an external timing input, providing 5ns resolution time. This 10 MHz oscillator, central to the TSync timing functions, uses the last known reference to increment (“freewheel”) in the absence of a timing input.

The TSync card generates an IRIG AM and DCLS output pair, as well as 10 MHz sine wave and 1PPS outputs.

The board’s four programmable inputs may be used as event capture inputs, dedicated to your time-tagging applications. Four user-programmable alarm and frequency outputs are also provided. Programmable output functions include a periodic pulse or “heartbeat,” square wave, and programmable start/stop time “alarm” output.

Key to TSync functionality is the ability to generate interrupts. Using one of the many available Spectracom driver packages, you may configure your card using interrupt-driven algorithms to support your unique applications.

2 Installing and Uninstalling the Driver

2.1 Linux

The driver is designed to operate with 32-bit or 64-bit Linux kernel version 2.6.x and 3.x running on a PC system with x86 compatible processor(s).

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

Included with the driver are example programs with source, utilizing the API library.

2.1.1 Installing the Driver

NOTE: If the system contains a previously installed version of the driver, that version must first be uninstalled. See **2.1.2 Uninstalling the Driver** below.

This driver is delivered via the archive file:

```
tsync.<rev>.tar.gz
```

- Where `<rev>` is the current driver revision

1) Open a terminal window.
2) Make sure you are logged in as a root user.
3) Copy the driver file to a convenient directory location.
4) Change to the directory in which the driver files were copied.
5) Extract the driver using the following command:

```
> gunzip –c tsync.<rev>.tar.gz | tar –xvf –
```

6) Build the driver by issuing the commands below:

```
> cd tsync
> cd linux
> make clean
> make
> make install
```

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

7) Load the driver by issuing the command:

```
> modprobe tsyncpci
```

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

```
> lsmod
```

9) Verify that the driver "`tsyncpci`" is present.

2.1.2 Uninstalling the Driver

1) Open a terminal window.
2) Make sure you are logged in as a root user.
3) Unload the driver by issuing the following command:

```
> rmmod tsyncpci
```

4) Change to the directory in which the driver files were copied.
5) Unload the driver by issuing the following commands:

```
> cd linux
> make uninstall
```

2.1.3 Generating the Example Programs

The TSync driver provides both a static library (`libtsync.a`) and a shared library (`libtsync.so`). The example programs are built linked with the static library. To use the example programs with the shared library, modify the example "makefile" by replacing the `libtsync.a` with `libtsync.so` and rebuild.

1) Open a terminal window.
2) Change to the directory in which the driver and its sources were extracted.
3) Build the example programs by issuing the commands below:

```
> cd tsync
> cd examples
> make clean
> make
```

To see usage help for any example program, execute the program with no parameters:

```
> ./HW_GetTime
```

2.1.4 Generating the Upgrade Tool

1) Open a terminal window.
2) Change to the directory in which the driver and its sources were extracted.
3) Build the upgrade tool by issuing the commands below:

```
> cd tsync
> cd upgrade
> make clean
> make
```

To see usage help for the upgrade tool, execute the program with no parameters:

```
> ./TSyncUpgrade
```

2.2 Solaris

The driver is designed to operate with 32bit or 64bit Solaris running on a PC system with x86 compatible processor(s).

Included with the driver are example programs with source utilizing the API library.

2.2.1 Installing the Driver

NOTE: If the system contains a previously installed version of the driver, that version must first be uninstalled. See **2.2.2 Uninstalling the Driver** below.

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

```
tsync.pkg
```

This package is delivered via the archive file:

```
tsync.<rev>.<arch>.pkg.tar.gz
```

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

1) Open a terminal window.
2) Make sure you are logged in as a root user.
3) Copy the package file to a convenient directory location.
4) Change to the directory in which the driver package file was copied.
5) Extract the driver using the following command:

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

6) Install the driver package by issuing the following command:

```
> pkgadd -d tsync.pkg
```

2.2.2 Uninstalling the Driver

1) Open a terminal window.
2) Make sure you are logged in as a root user.
3) Uninstall the driver by issuing the following command:

```
> pkgrm tsync
```

2.2.3 File Locations

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

x86 Architecture

File	Destination	64-bit API	32-bit API
tsync (driver)	/usr/kernel/drv	No	Yes
tsync.conf	/usr/kernel/drv	No	Yes
tsync (driver)	/usr/kernel/drv/amd64	Yes	No
tsync.conf	/usr/kernel/drv/amd64	Yes	No
libtpro_32.a	/usr/lib	Yes	Yes
libtsync32.a	/usr/lib	Yes	Yes
libtpro.a	/usr/lib/amd64	Yes	No
libtsync64.a	/usr/lib/amd64	Yes	No
tpro.h	/usr/include	Yes	Yes
tpro_error_codes.h	/usr/include	Yes	Yes
tsync.h	/usr/include	Yes	Yes
tsync_error_codes.h	/usr/include	Yes	Yes
tsync_hw.h	/usr/include	Yes	Yes
tsync_platform_mapping.h	/usr/include	Yes	Yes
tsync_example.h	/usr/include	Yes	Yes

2.2.4 Example Programs

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

`TSyncExamples.i86pc.tar.gz` – 32-bit programs for x86 based machines

To see usage help for any example program, execute the program with no parameters:

```
> ./HW_GetTime
```

2.3 Windows

The driver is designed to operate with the following 32-bit or 64-bit Windows operating systems running on a PC system with x86 compatible processor(s):

Windows 2000
Windows XP
Windows Server 2003
Windows Vista
Windows Server 2008 & R2
Windows 7

2.3.1 Installing the Driver

NOTE: If the system contains a previously installed version of the driver, that version must first be uninstalled. See **2.3.2 Uninstalling the Driver** below.

1) Driver setup should start automatically when the driver CD is loaded (if autorun is enabled). Otherwise, run setup manually by running the following program:

`<CD_Drive>:\\windows\setup.exe`

2) Follow the on-screen prompts. The setup utility will copy the application files. When this process is finished, driver installation is complete.

2.3.2 Uninstalling the Driver

1) Go to `Control Panel` → `Add/Remove Programs`.

NOTE*:* This may vary depending on your operating system. Refer to the appropriate support documentation for your OS if unsure.

2) Select and remove the `TSYNC PCI` program.
3) Follow the on-screen prompts. When this process is finished, driver uninstallation is complete.

2.3.3 File Locations

On 32-bit operating systems, the 32-bit library file and DLL (`Tsync.lib` & `Tsync.dll`) are located in the "`Spectracom\TSYNC PCI\Dev`" directory under "`Program Files`."

On 64-bit operating systems, the 64-bit library file and DLL (`Tsync.lib` & `Tsync.dll`) are located in the "`Spectracom\TSYNC PCI\Dev64`" directory under "`Program Files (x86)`."

Included with the driver are example programs with source utilizing the API library, a control utility exercising legacy API functionality, and a clock daemon tray utility and service that is used to set the host computer's system time.

2.3.4 Example Programs

The driver package includes folders with example programs to interface to the board. The source code and make files for the example programs are included. All of the example programs were compiled using **Visual Studio 2005**. All example programs are 32-bit applications.

To see usage help for any example program, execute the program with no parameters:

```
> HW_GetTime.exe
```

2.3.5 Control Utility

The control utility provides a graphical interface for performing legacy functionality with the board. The control utility is run as follows:

1) From the Windows Start menu, select the "`Programs`" folder.
2) Select the "`Spectracom Corp\TSync PCI`" folder.
3) Select the "`TSync Control Utility`" program.

2.3.6 Clock Daemon

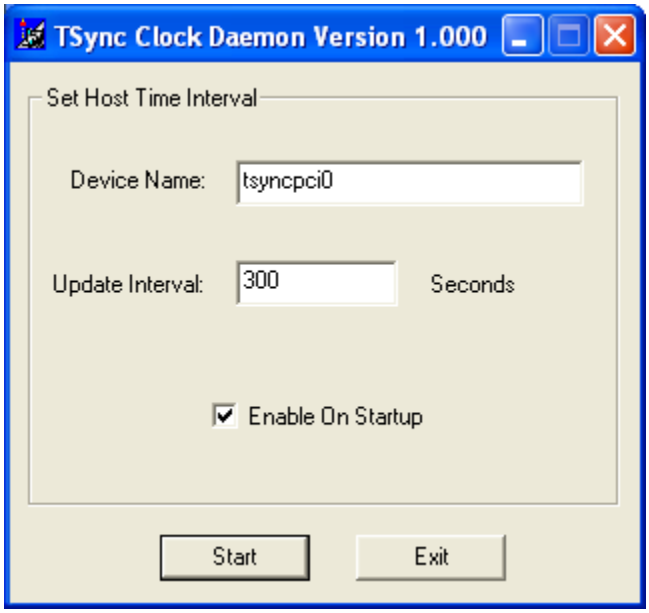
Two clock daemon utilities are provided that can be used to set the computers system clock; "`Clock Daemon.exe`" and "`ClockDaemonService.exe.`" Both will query the board and set the system clock on a periodic basis. Only one should be running at a time.

NOTE: If the board is not synchronized to an external reference, the system clock will not be set.

Clock Daemon.exe:

This program can be run as follows:

1) From the Windows Start menu, select the "`Programs`" folder.
2) Select the "`Spectracom Corp\TSync PCI`" folder.
3) Select the "`Clock Daemon`" program.



Device Name: Enter the board to be used as a time source. The default is "`tsyncpci0.`"

Update Interval: Enter the desired time interval between system queries, in seconds. The default is `300` seconds.

Enable On Startup: Check this box to have system clock synchronization by the clock daemon start automatically when the program starts. Alternatively, the synchronization can be started or stopped manually.

ClockDaemonService.exe:

"`ClockDaemonService.exe`" will set the computer's system clock like the "`Clock Daemon.exe`" but will run as a Windows service. As a service, it can be set to run automatically on Windows startup.

This program can be installed as a service as follows:

1) From the Windows Start menu, select the "Programs" folder.
2) Select the "`Spectracom Corp\TSync PCI`" folder.
3) Select the "`ClockDaemonServiceUtility`" program.

This program can be removed as a service by running "`ClockDaemonService.exe -r`" from a command line.

The controls for device name and update interval are set up by the "`daemon.ini`" located in the "`Spectracom\TSYNC PCI\Control`" folder with the clock daemon executables. This initialization file is shared by both clock daemon executables.

3 Using the Driver to Upgrade

An upgrade tool is provided with the driver to support field upgrades of the configuration and firmware/FPGA loads of the board. The board can be upgraded with the following steps:

1) Open a terminal or command prompt.
2) Change to the directory in which the driver was installed.
3) Change to the `upgrade` directory:

```
> cd upgrade
```

NOTE: On Windows systems, open a command prompt and navigate to `C:\\Program Files\Spectracom\TSYNC PCI\Upgrade`.

.

4) Copy the upgrade image files to the `upgrade` directory. The upgrade image files will consist of a configuration image (`patch_img.bin`) and/or firmware/FPGA images (`rt_fw.bin` and `rt_fpga.bin`).
5) Run the Upgrade Tool:

```
> ./TSyncUpgrade 0
```

Where `0` is the board instance of the TSync board to be upgraded.

6) When the tool completes, a reset of the board is required for the upgrade to take effect. This can be done without resetting the host computer by using the reset example program:

```
> cd ..
> cd examples
> ./SS_Reset 0 0
```

Where the first `0` is the board instance of the TSync board to be reset.

4 Interface to the Driver API

4.1 Header Files

4.1.1 Tsync.h

```
#ifndef _defined_TSYNC_H
#define _defined_TSYNC_H

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

#ifdef __cplusplus
extern "C" {
#endif

#include "tsync_platform_mapping.h"
#include "tsync_agps.h"

#ifndef DLL_EXPORT
#ifndef WIN32
#define DLL_EXPORT /* */
#else
#define DLL_EXPORT __declspec(dllexport)
#endif
#endif

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

    /*
    ** Propagation delay min's and max's for CPCI and PCI boards
    */
#define PMC_CPCI_SYNCP_DELAY_MIN        ( -999 )
#define PMC_CPCI_SYNCP_DELAY_MAX        ( 999 )
#define PCI_SYNCP_DELAY_MIN             ( -1000 )
#define PCI_SYNCP_DELAY_MAX             ( 8999 )

    /*
    ** Position conversion constants
    */
#define PI                              ( 3.1415926535898 )
#define RAD_TO_DEG                      ( 180.0 / PI )
#define DEG_TO_RAD                      ( PI / 180.0 )

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

    /*
```

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


#include "tsync_error_codes.h"

    /*
    ** Handle to the TSync board obj available to the user
    */
    typedef void* TSYNC_BoardHandle;

    typedef uint8_t CI_CAI;
    typedef uint8_t CI_IID;

#define TSYNC_CAPABILITIES_PER_PAGE     ( 16 )
#define TSYNC_DATA_BLOCK_SIZE           ( 1024 )
#define TSYNC_TABLE_ENTRY_NUM           ( 15 )
#define TSYNC_STATE_TABLE_ENTRY_NUM     ( 16 )
#define TSYNC_INIT_RESULT_NUM           ( 16 )
#define TSYNC_SAT_INFO_NUM              ( 32 )
#define TSYNC_IR_SUBFRAME_NUM           ( 10 )
#define TSYNC_IR_CFDATA_NUM             ( 3 )
#define TSYNC_IP_SUBFRAME_NUM           ( 10 )
#define TSYNC_IP_CFDATA_NUM             ( 3 )
#define TSYNC_OSC_MAN_MOD_STRING_SIZE   ( 16 )
#define TSYNC_OSC_DISC_CMD_SIZE         ( 8 )       //TBD!!!
#define TSYNC_OSC_DISC_DATA_SIZE        ( 8 )       //TBD!!!
#define TSYNC_OSC_SER_NUM_STRING_SIZE   ( 32 )
#define TSYNC_PTP_VER_STR_LEN           ( 28 )
#define TSYNC_PTP_DATE_STR_LEN          ( 12 )
#define TSYNC_QUAL_LOG_NUM              ( TSYNC_SAT_INFO_NUM + 1 )
#define TSYNC_PARM_NUM                  ( 4 )
#define TSYNC_GR_AGPS_SIZE              ( 256 )

#define TSYNC_DEFAULT_GPS_INSTANCE      (0)

    /*****************************************************
    **     Define Enumerations
    *****************************************************/

    /*** General ***/
    typedef enum
    {
        SIG_CTL_NONE = 0,           // No Signature Control - always ON
        SIG_CTL_SYNC = 1,           // Signature Control based on SYNC status
        SIG_CTL_REF  = 2,           // Signature Control based on Reference
                                    // status
        SIG_CTL_OFF  = 3,           // Ultimate Signature Control - always OFF
        SIG_CTL_NUM

    } SIG_CTL;

    typedef enum
    {                               // TFOM based on Estimated Time Error (ETE)
        TFOM_MIN       = 0,         // Minimum TFOM value
        TFOM_UNDEFINED = 0,         // TFOM is not defined, ETE is unknown
        TFOM_1         = 1,         //             ETE <=   1 nsec
        TFOM_2         = 2,         //    1 nsec < ETE <=  10 nsec
        TFOM_3         = 3,         //   10 nsec < ETE <= 100 nsec
        TFOM_4         = 4,         //  100 nsec < ETE <=   1 usec
        TFOM_5         = 5,         //    1 usec < ETE <=  10 usec
```

```
    TFOM_6         = 6,         //   10 usec < ETE <= 100 usec
    TFOM_7         = 7,         //  100 usec < ETE <=   1 msec
    TFOM_8         = 8,         //    1 msec < ETE <=  10 msec
    TFOM_9         = 9,         //   10 msec < ETE <= 100 msec
    TFOM_10        = 10,        //  100 msec < ETE <=   1 sec
    TFOM_11        = 11,        //    1 sec  < ETE <=  10 sec
    TFOM_12        = 12,        //   10 sec  < ETE <=  100 sec
    TFOM_13        = 13,        //  100 sec  < ETE <= 1000 sec
    TFOM_14        = 14,        // 1000 sec  < ETE <= 10000 sec
    TFOM_15        = 15,        //             ETE >  10000 sec
    TFOM_NUM,                   // Number of TFOM values
    TFOM_MAX       = (TFOM_15)

} TFOM;

typedef enum
{
    EDGE_MIN     = 0,
    EDGE_FALLING = 0,
    EDGE_RISING  = 1,
    EDGE_BOTH    = 2,
    EDGE_NUM

} EDGE;

typedef enum
{
    LEVEL_LOW  = 0,
    LEVEL_HIGH = 1

} LEVEL;

typedef enum
{
OSC_UNKNOWN  = -1,          // Unknown oscillator type

OSC_TCXO_1   = 0,           //  1ppm
OSC_TCXO_2   = 1,           //  1ppm

OSC_OCXO_1   = 2,           // 50ppb
OSC_OCXO_2   = 3,           //  5ppb
OSC_OCXO_3   = 4,           //  1ppb
OSC_OCXO_4   = 5,           //  1ppb

OSC_RBXO_1   = 6,           // .1ppb
OSC_RBXO_2   = 7,           // .1ppb
OSC_RBXO_3   = 8,           // .1ppb

OSC_RBXO_1_2 = 9,           // .1ppb
OSC_RBXO_1_3 = 10,          // .1ppb
OSC_RBXO_1_4 = 11,          // .1ppb

OSC_NUM

} OSC;

typedef enum
{
    ML_START_TIME_SCALES = 0,           // First time scale in list
    ML_TIME_SCALE_UTC    = 0,           // Universal Coordinated Time
    ML_TIME_SCALE_TAI    = 1,           // International Atomic Time
    ML_TIME_SCALE_GPS    = 2,           // Global Positioning System
    ML_TIME_SCALE_LOCAL  = 3,           // UTC w/local rules for time
```

```
                                        // zone/DST
    ML_NUM_TIME_SCALES   = 4,           // Number of time scales

    ML_TIME_SCALE_MAX    = 15           // Maximum number of timescales

} ML_TIME_SCALE;

typedef enum
{
    ML_TIME_START_TYPES = 0,            // First time type in the list
    ML_TIME_DOYTIME     = 0,            // Year, Day of Year, Hour, Min,
                                        // Sec, nsec
    ML_TIME_BCDTIME     = 1,            // BCD Year, Day of Year, Hour,
                                        // Min, Sec, ms, us
    ML_TIME_SECONDS     = 2,            // Total number of seconds in
                                        // epoch, nsec
    ML_TIME_NUM_TYPES                   // Number of time types

} ML_TIME_TYPE;

typedef enum
{
    ML_DST_REF_LCL = 0,                 // Reference is local time
    ML_DST_REF_UTC = 1                  // Reference is UTC

} ML_DST_REF;

typedef enum
{
    ML_MONTH_NONE = 0,
    ML_MONTH_JAN  = 1,
    ML_MONTH_FEB  = 2,
    ML_MONTH_MAR  = 3,
    ML_MONTH_APR  = 4,
    ML_MONTH_MAY  = 5,
    ML_MONTH_JUN  = 6,
    ML_MONTH_JUL  = 7,
    ML_MONTH_AUG  = 8,
    ML_MONTH_SEP  = 9,
    ML_MONTH_OCT  = 10,
    ML_MONTH_NOV  = 11,
    ML_MONTH_DEC  = 12

} ML_MONTH;

typedef enum
{
    ML_WOM_NONE   = 0,
    ML_WOM_FIRST  = 1,
    ML_WOM_SECOND = 2,
    ML_WOM_THIRD  = 3,
    ML_WOM_FOURTH = 4,
    ML_WOM_LAST   = 5

} ML_WOM;

typedef enum
{
    ML_DOW_SUN = 0,
    ML_DOW_MON = 1,
    ML_DOW_TUE = 2,
    ML_DOW_WED = 3,
    ML_DOW_THU = 4,
```

```
    ML_DOW_FRI = 5,
    ML_DOW_SAT = 6,

} ML_DOW;

typedef enum
{
    ML_HOUR_START_TYPES = 0,            // First hour type in the list
    ML_HOUR_12          = 0,            // 12-hour format
    ML_HOUR_24          = 1,            // 24-hour format
    ML_HOUR_NUM_TYPES                   // Number of hour types

} ML_HOUR;

// Number of different message formats that can be sent each second
#define AL_FMT_MAX              (3)

typedef enum
{
    AL_FMT_SPEC_0     = 0x00000000,     // Spectracom format 0
    AL_FMT_SPEC_1     = 0x00000001,     // Spectracom format 1
    AL_FMT_SPEC_2     = 0x00000002,     // Spectracom format 2
    AL_FMT_SPEC_3     = 0x00000003,     // Spectracom format 3
    AL_FMT_SPEC_4     = 0x00000004,     // Spectracom format 4
    AL_FMT_SPEC_5     = 0x00000005,     // Spectracom format 4
    AL_FMT_SPEC_6     = 0x00000006,     // Spectracom format 6
    AL_FMT_SPEC_7     = 0x00000007,     // Spectracom format 7
    AL_FMT_SPEC_8     = 0x00000008,     // Spectracom format 8
    AL_FMT_SPEC_9     = 0x00000009,     // Spectracom format 9
    AL_FMT_NMEA_GGA   = 0x0000000A,     // NMEA GGA message
    AL_FMT_NMEA_RMC   = 0x0000000B,     // NMEA RMC message
    AL_FMT_NMEA_ZDA   = 0x0000000C,     // NMEA ZDA message
    AL_FMT_BBC_01     = 0x0000000D,     // BBC format 1
    AL_FMT_BBC_02     = 0x0000000E,     // BBC format 2
    AL_FMT_BBC_03     = 0x0000000F,     // BBC format 3 PSTN
    AL_FMT_BBC_04     = 0x00000010,     // BBC format 4
    AL_FMT_153C_BB    = 0x00000011,     // ICD-153C 253  Buffer Box
    AL_FMT_153C_TT    = 0x00000012,     // ICD-153C 5101 Time Transfer
    AL_FMT_153C_CS    = 0x00000013,     // ICD-153C 5040 Current Status
    AL_FMT_ER_0       = 0x00000014,     // EndRun Format
    AL_FMT_ER_1       = 0x00000015,     // EndRunX Extended Format
    AL_FMT_EVT_0      = 0x00000016,     // Event Timestamp Format 0
    AL_FMT_EVT_1      = 0x00000017,     // Event Timestamp Format 0
    AL_FMT_BBC_05     = 0x00000018,     // BBC format 5 (RMC)

    // Add new formats here. Maintain order and values.

    AL_FMT_SPEC_1S    = 0x00000101,     // Format 1S variant

    AL_FMT_UNKNOWN    = 0x0000FFFF,     // Unknown (auto)

} AL_FMT;

typedef enum
{
    AL_OMODE_BC = 0,                    // Broadcast
    AL_OMODE_OT = 1,                    // On-Time
    AL_OMODE_IM = 2,                    // Immediate
    AL_OMODE_EV = 3,                    // Event Immediate

} AL_OUT_MODE;

// The AL_PPS type defines the source of a 1PPS signal for a ASCII Input
```

```
// port.
typedef enum
{
    AL_PPS_OTP = 0,                     // Decoded 1PPS selected
    AL_PPS_SIG = 1,                     // External ATC 1PPS input selected

} AL_PPS;

// The AO_SOURCE enumeration describes the different possible
// sources of an ASCII message
typedef enum
{
    AO_SOURCE_SYSTEM = 0, // System Time is output source (normal)
    AO_SOURCE_EVTBUF = 1  // Event Buffer

} AO_SOURCE;

typedef enum
{
    IL_MODE_AUTO   = 0,
    IL_MODE_MANUAL = 1

} IL_MODE;

typedef enum
{
    IL_FMT_IRIG_A     = 0,
    IL_FMT_IRIG_B     = 1,
    IL_FMT_IRIG_G     = 2,
    IL_FMT_NASA_36    = 3,
    IL_FMT_IRIG_E_100 = 4,
    IL_FMT_IRIG_E_1K  = 5,
    IL_FMT_UNKNOWN    = 7,              // Unknown
    IL_FMT_AUTO                         // Auto-detection

} IL_FMT;

typedef enum
{
    IL_MOD_DCLS        = 0,             // IRIG DCLS only
    IL_MOD_AM          = 1,             // IRIG AM only
    IL_MOD_MAN         = 2,             // IRIG Manchester coding
    IL_MOD_UNKNOWN     = 3,             // Unknown
    IL_MOD_AM_AND_DCLS = 4,             // Port generates both AM and DCLS
    IL_MOD_AM_OR_DCLS  = 5              // Port generate AM or DCLS

} IL_MOD;

typedef enum
{
    IL_FRQ_NONE        = 0,             // No Carrier/Index count interval
    IL_FRQ_100HZ       = 1,
    IL_FRQ_1KHZ        = 2,
    IL_FRQ_10KHZ       = 3,
    IL_FRQ_100KHZ      = 4,
    IL_FRQ_1MHZ        = 5,
    IL_FRQ_UNKNOWN     = 6              // Unknown

} IL_FRQ;

typedef enum
{
    IL_CE_BCDT_CF_SBS      = 0,         // BCD TOY, Ctrl Func, Binary
```

```
                                        // Seconds
    IL_CE_BCDT_CF          = 1,         // BCD TOY, Ctrl Func
    IL_CE_BCDT             = 2,         // BCD TOY
    IL_CE_BCDT_SBS         = 3,         // BCD TOY, Binary Seconds,
    IL_CE_BCDT_BCDY_CF_SBS = 4,         // BCD TOY/Year, Ctrl Func, Binary
                                        // Seconds
    IL_CE_BCDT_BCDY_CF     = 5,         // BCD TOY/Year, Ctrl Func
    IL_CE_BCDT_BCDY        = 6,         // BCD TOY/Year,
    IL_CE_BCDT_BCDY_SBS    = 7,         // BCD TOY/Year, Binary Seconds
    IL_CE_UNKNOWN          = 8          // Unknown - No fields

} IL_CE;

typedef enum
{
    IL_CF_MIN      = 0,                 // Minimum value of CF
    IL_CF_UNKNOWN  = 0,                 // Unknown - All bits ignored
    IL_CF_200_04   = 1,                 // Fields conform to RCC 200-04
    IL_CF_1344     = 2,                 // Fields conform to IEEE
                                        // C37.118-2005
    IL_CF_SPEC     = 3,                 // Fields conform to Spectracom
                                        // format
    IL_CF_SPEC_FAA = 4,                 // Fields conform to Spectracom FAA
                                        // format
    IL_CF_NASA     = 5,                 // Fields conform to NASA formats

    IL_CF_NUM                           // Number of CF types

} IL_CF;

// The QL_FMT type defines the HaveQuick formats.
typedef enum
{
    QL_FMT_HQ_I     = 0,           // STANAG 4246 HaveQuick I
    QL_FMT_HQ_II    = 1,           // STANAG 4246 HaveQuick II
    QL_FMT_HQ_IIA   = 2,           // STANAG 4372 HaveQuick IIA
    QL_FMT_4430_STM = 3,           // STANAG 4430 Standard Time Message
    QL_FMT_4430_XHQ = 4,           // STANAG 4430 Extended HaveQuick
    QL_FMT_GPS_BCD  = 5,           // ICD-GPS-060A Binary Coded Decimal
    QL_FMT_GPS_HQ   = 6,           // ICD-GPS-060A HaveQuick

    QL_FMT_NUM,                    // Number of formats
    QL_FMT_START    = 0,           // Start of format list

    QL_FMT_UNKNOWN  = -1           // Unknown format

} QL_FMT;

// The ESL_FMT type defines the SMPTE/EBU formats.
typedef enum
{
    ESL_FMT_LTCBBC   = 0,               // SMPTE/EBU Format LTC BBC
    ESL_FMT_LTC309M  = 1,               // SMPTE/EBU Format LTC 309m

    ESL_FMT_NUM,                        // Number of formats
    ESL_FMT_START    = 0,               // Start of format list

    ESL_FMT_UNKNOWN  = -1               // Unknown format

} ESL_FMT;

/*** Flash services ***/
typedef enum
```

```
{
    FS_IMG_RT_FW        = 0,        // Run-time firmware image
                                    // (upgradeable)
    FS_IMG_RT_FPGA      = 1,        // Run-time FPGA image
                                    // (upgradeable)
    FS_IMG_DEF_FW       = 2,        // Default firmware image
                                    // (read-only)
    FS_IMG_DEF_FPGA     = 3,        // Default FPGA image
                                    // (read-only)
    FS_IMG_BL           = 4,        // Boot Loader
                                    // (read-only)
    FS_IMG_CFPGA        = 5,        // Compressed FPGA image
                                    // (read-only)
    FS_IMG_EE_PATCH     = 6,        // EEPROM Patch image
                                    // (upgradeable)
    FS_IMG_SER_DEF_ADDR = 7,        // Serial flash default addr iamge
                                    // (upgradeable)
    FS_IMG_SER_RT_FPGA  = 8,        // Run-time serial flash FPGA image
                                    // (upgradeable)
    FS_IMG_SER_DEF_FPGA = 9,        // Default serial flash FPGA image
                                    // (upgradeable)
    FS_IMG_PTP_CUR_FW   = 10,       // Run-Time PTP Module Firmware
                                    // (Upgradeable)
    FS_IMG_PTP_DEF_FW   = 11,       // Default PTP Module Firmware
                                    // (Upgradeable)
         FS_IMG_LICENSING    = 12,       // Licensing image file
                                    // (Upgradeable)
    FS_IMG_NUM_ITEMS,

    FS_IMG_RSVD_BL  = 0xFF          // Reserved for bootloader use

} FS_IMG;

typedef enum
{
    CI_ACC_NONE = 0,
    CI_ACC_GET  = 1,
    CI_ACC_SET  = 2,
    CI_ACC_BOTH = 3

} CI_ACCESS;

/*** Signal Level ***/
typedef enum
{
   SL_10V  = 0, // Signal Level 10V
   SL_5V   = 1, // Signal Level 5V
   SL_3_3V = 2  // Signal Level 3.3V
} SIG_LEVEL;

/*** Electrical type ***/
typedef enum
{
   ET_TTL   = 0,
   ET_RS485 = 1
} ELEC_TYPE;

   /*** Reference Selection ***/
typedef enum
{
   RS_STANDARD = 0,
   RS_EXTENDED = 1,
      RS_NONE     = 2
```

```
} REF_SEL;

/*** Clock services ***/


/*** Log service ***/
typedef enum
{
    LS_ALARM_SYNC     = 0,              // Not in Sync
    LS_ALARM_HOLDOVER = 1,              // In Holdover
    LS_ALARM_FREQ_ERR = 2,              // Frequency Error
    LS_ALARM_SELF_REF = 3,              // Self Reference only
    LS_ALARM_SW_ERR   = 4,              // Software Error
    LS_ALARM_1PPS     = 5,              // 1PPS is not in specification
    LS_ALARM_REF_CHG  = 6,              // Reference Change
    LS_ALARM_HW_ERR   = 7,              // Hardware Error

    LS_ALARM_NUM                        // Number of alarm types

} LS_ALARM;


/*** Oscillator Monitor service ***/
typedef enum
{
    XS_CMD_AUTO    = 0,                 // Automatically start a new window
                                        // of measurement when a window
                                        // completes
    XS_CMD_START   = 1,                 // Begin a single window of
                                        // measurement
    XS_CMD_STOP    = 2,                 // Stop measuring immediately
    XS_CMD_FINISH  = 3,                 // Stop after current window
                                        // completes
    XS_CMD_RESTART = 4                  // Restart window measurement (Does
                                        // not change meter state)

} XS_CMD;

typedef enum
{
    XS_STATE_STOPPED = 0,               // Not measuring
    XS_STATE_RUNNING = 1,               // Measuring
    XS_STATE_ENDING  = 2                // Measuring until end of window

} XS_STATE;

typedef uint32_t TSYNC_METER_HANDLE;


/*** Supervisor service ***/
typedef enum
{
    SS_EVT_SYNC = 0,                    // In sync / Out of sync
    SS_EVT_REF  = 1                     // Valid ref(s) / No valid ref(s)

} SS_EVENT;

typedef enum
{
    SS_RESET_BRD  = 0,                  // Reset the entire board except
                                        // for FPGA
    SS_RESET_MIC  = 1,                  // Reset the microprocessor &
                                        // peripherals
```

```
    SS_RESET_FPGA = 2,                  // Reset the FPGA only
    SS_RESET_NUM

} SS_RESET;

typedef enum
{
    SS_TS_MIN      = 0,                 // Minimum Timestamp index
    SS_TS_TIME_REF = 0,                 // Timestamp on Time Reference
                                        // change
    SS_TS_1PPS_REF = 1,                 // Timestamp on 1PPS Reference
                                        // change
    SS_TS_TFOM     = 2,                 // Timestamp on TFOM value change
    SS_TS_SYNC     = 3,                 // Timestamp on Sync state change
    SS_TS_HOLDOVER = 4,                 // Timestamp on Holdover state
                                        // change
    SS_TS_LOST_REF = 5,                 // Timestamp on entering Holdover

    SS_TS_NUM                           // Number of Timestamps

} SS_TS_SRC;


/*** Reference service ***/
typedef enum
{
    RS_TT_START_TYPES = 0,
    RS_TT_FACT        = 0,              // Factory default table (built at
                                        // runtime)
    RS_TT_USER        = 1,              // User-default table (stored in
                                        // eeprom)
    RS_TT_CURRENT     = 2,              // Current working table

    RS_TT_NUM_TYPES

} RS_TABLE_TYPE;


/*** General Purpose services ***/
typedef enum
{
    OD_MODE_DIRECT      = 0,            // Direct Output Value
    OD_MODE_MATCH_TIME  = 1,            // Match Time Output
    OD_MODE_SQUARE_WAVE = 2,            // Square Wave
    OD_MODE_RESERVED    = 3

} OD_MODE;

/*** Frequency Reference components ***/
typedef enum
{
    FR_MODE_SEC      = 0,       // Reference valid as secondary reference:
                                //   Requires another valid reference
                                //   to synchronize the system before the
                                //   frequency reference can be determined
                                //   to be valid.
    FR_MODE_PRI      = 1,       // Reference valid as primary reference:
                                //   Can be determined to be valid based
                                //   solely on its own presence.  The 1PPS
                                //   of the system should be considered to
                                //   be at some random point in the second.

    FR_MODE_NUM
```

```
} FR_MODE;

/*** LED services ***/
typedef enum
{
    EC_MODE_SYNC     = 0,              // LED indicates sync status
    EC_MODE_HOLDOVER = 1,              // LED indicates holdover status
    EC_MODE_ALARM    = 2,              // LED indicates alarm status
    EC_MODE_1PPS     = 3,              // LED goes on briefly when 1PPS
                                       // occurs
    EC_MODE_MANUAL   = 4,              // LED manually controlled

    EC_MODE_NUM

} EC_MODE;

typedef enum
{
    EC_STATE_OFF   = 0,                // LED off solid
    EC_STATE_ON    = 1,                // LED on solid
    EC_STATE_BLINK = 2,                // LED blinks on/off (2Hz rate)
    EC_STATE_CODE  = 3,                // LED blinks a code (2Hz rate,
                                       // 2sec pause)

    EC_STATE_NUM

} EC_STATE;

typedef enum
{
    LE_ALL = -1,

    LE_0   = 0,
    LE_1   = 1,
    LE_2   = 2,
    LE_3   = 3,
    LE_4   = 4,
    LE_5   = 5,

    LE_LEDS_NUM

} LE_INDEX;


/*** GPS Reference services ***/
typedef enum
{
    GL_DYN_LAND = 0,
    GL_DYN_SEA  = 1,
    GL_DYN_AIR  = 2,
    GL_DYN_STAT = 3

} GL_DYN;

typedef enum
{
    GL_MODE_1SAT = 0,
    GL_MODE_STND = 1,
    GL_MODE_CONT = 2,
    GL_MODE_AVRG = 3,
    GL_MODE_TIME = 4,
    GL_MODE_STBY = 5,
```

```
    GL_MODE_SELF = 6

} GL_MODE;

/*** GPS Reset types ***/
typedef enum
{
    GL_RESET_COLD = 0,          // Clear data in RAM (like power-cycle)
    GL_RESET_WARM = 1,          // Clear ephemeris and osc. uncertainty
    GL_RESET_HOT  = 2,          // No clear, SW reset, rerun self-test
    GL_RESET_POS  = 3,          // Clear position in receiver flash
    GL_RESET_FACT = 4,          // Cold reset + restore factory settings
    GL_RESET_SAVE = 5,          // Reset GPS + save settings

    GL_RESET_SAASM_ZKEY  = 16,  // SGPS Zeroize Keys
    GL_RESET_SAASM_ZCLR  = 17,  // SGPS Zeroize Clear
    GL_RESET_SAASM_ZEMG  = 18,  // SGPS Emergency Zeroize
    GL_RESET_SAASM_RST   = 19,  // Reset SGPS GSSIP interface
    GL_RESET_SAASM_URST  = 20,  // Reset SGPS ALL GSSIP interfaces

    GL_RESET_NONE        = 0xFF  // No Reset required for Web UI Display

} GL_RESET;

/*** GPS Antenna States ***/
typedef enum
{
    GL_ANT_OK    = 0,
    GL_ANT_SHORT = 1,
    GL_ANT_OPEN  = 2,
    GL_ANT_UNK   = 3

} GL_ANT_STATUS;

/*** GPS Set/Get Parameter Type ***/
typedef enum
{
    // TSIP Parameters
    //GL_PARM_TSIP_TBD     = 0x00010000,

    // GSSIP Parameters
    GL_PARM_GSSIP_KEY_STATE = 0x00020000,
    GL_PARM_GSSIP_COM1_EN   = 0x00020001,
    GL_PARM_GSSIP_COM1_CFG  = 0x00020002,
    GL_PARM_GSSIP_PWR_CTRL  = 0x00020003,

    // NMEA Parameters
    //GL_PARM_NMEA_TBD     = 0x00030000,

    // Add more here for other receivers...

} GL_PARM;

/*** GSSIP COM1 Usage Type - Mutualy exclusive uses for COM1 ***/
typedef enum
{
    GL_GSSIP_COM1_HS_IN    = 0,    // Idle, waiting for HotStart Input
    GL_GSSIP_COM1_HS_OUT   = 1,    // Initiate a HotStart Output
    GL_GSSIP_COM1_HVQK     = 2,    // Output HaveQuick messages
    GL_GSSIP_COM1_PPS      = 3,    // Output 1PPS and 5101 message
  //GL_GSSIP_COM1_NMEA     = 4,    // Output NMEA messages (RESERVED)

} GL_GSSIP_COM1;
```

```
/*** GSSIP COM1 Power control value ***/
typedef enum
{
    GL_GSSIP_PWR_OFF   = 0,
    GL_GSSIP_PWR_ON    = 1,
    GL_GSSIP_PWR_CYCLE = 2,

} GL_GSSIP_PWR;

/*** PTP Reference services ***/
typedef enum
{
    PTL_RESET_COLD = 0,
    PTL_RESET_HOT  = 1,
    PTL_RESET_FACT = 2

} PTL_RESET;

typedef enum
{
    PTL_PTP_STATE_INITIALIZING = 0,
    PTL_PTP_STATE_FAULTY       = 1,
    PTL_PTP_STATE_DISABLED     = 2,
    PTL_PTP_STATE_LISTENING    = 3,
    PTL_PTP_STATE_PRE_MASTER   = 4,
    PTL_PTP_STATE_MASTER       = 5,
    PTL_PTP_STATE_PASSIVE      = 6,
    PTL_PTP_STATE_UNCALIBRATED = 7,
    PTL_PTP_STATE_SLAVE        = 8,

    PTL_PTP_STATE_COUNT        = 9

} PTL_PTP_STATE;

typedef enum
{
    PTL_DELAY_MECH_E2E      = 0x01,
    PTL_DELAY_MECH_P2P      = 0x02,
    PTL_DELAY_MECH_DISABLED = 0xFE

} PTL_DELAY_MECH;

typedef enum
{
    PTL_CLK_ACC_MIN            = 0x20,

    PTL_CLK_ACC_WITHIN_25_NS   = 0x20,
    PTL_CLK_ACC_WITHIN_100_NS  = 0x21,
    PTL_CLK_ACC_WITHIN_250_NS  = 0x22,
    PTL_CLK_ACC_WITHIN_1000_NS = 0x23,
    PTL_CLK_ACC_WITHIN_2_5_US  = 0x24,
    PTL_CLK_ACC_WITHIN_10_US   = 0x25,
    PTL_CLK_ACC_WITHIN_25_US   = 0x26,
    PTL_CLK_ACC_WITHIN_100_US  = 0x27,
    PTL_CLK_ACC_WITHIN_250_US  = 0x28,
    PTL_CLK_ACC_WITHIN_1_MS    = 0x29,
    PTL_CLK_ACC_WITHIN_2_5_MS  = 0x2A,
    PTL_CLK_ACC_WITHIN_10_MS   = 0x2B,
    PTL_CLK_ACC_WITHIN_25_MS   = 0x2C,
    PTL_CLK_ACC_WITHIN_100_MS  = 0x2D,
    PTL_CLK_ACC_WITHIN_250_MS  = 0x2E,
    PTL_CLK_ACC_WITHIN_1_S     = 0x2F,
```

```
    PTL_CLK_ACC_WITHIN_10_S    = 0x30,
    PTL_CLK_ACC_BEYOND_10_S    = 0x31,

    PTL_CLK_ACC_MAX            = 0x31,

    PTL_CLK_ACC_UNKNOWN        = 0xFE

} PTL_CLK_ACC;

typedef enum
{
    PTL_TIME_SRC_ATOMIC_CLOCK        = 0x10,
    PTL_TIME_SRC_GPS                 = 0x20,
    PTL_TIME_SRC_TERR_RADIO          = 0x30,
    PTL_TIME_SRC_PTP                 = 0x40,
    PTL_TIME_SRC_NTP                 = 0x50,
    PTL_TIME_SRC_HAND_SET            = 0x60,
    PTL_TIME_SRC_OTHER               = 0x90,
    PTL_TIME_SRC_INTERNAL_OSCILLATOR = 0xA0

} PTL_TIME_SRC;

typedef enum
{
    PTL_RESET_CAUSE_LOSS_LOCK        = 0x00,
    PTL_RESET_CAUSE_LOSS_CLOCK       = 0x01,
    PTL_RESET_CAUSE_EXTERNAL         = 0x02,
    PTL_RESET_CAUSE_POWER            = 0x03,
    PTL_RESET_CAUSE_WATCHDOG         = 0x04,
    PTL_RESET_CAUSE_REQUEST          = 0x05

} PTL_RESET_CAUSE;

typedef enum
{
    PTL_TRANS_PROTO_IPV4             = 0x00,
    PTL_TRANS_PROTO_ETH              = 0x01

} PTL_TRANS_PROTO;


/*** IRIG Reference services ***/


/*** Oscillator services ***/
typedef enum
{
    XO_MODE_DISC  = 0,                  // Disciplining
    XO_MODE_TEST  = 1,                  // Testing
    XO_MODE_RESET = 2                   // Reset

} XO_MODE;


/*** Shared Memory services ***/
typedef enum
{
    MS_DI_NMEA_GGA = 0,                // NMEA messages
    MS_DI_NMEA_GLL = 1,
    MS_DI_NMEA_GSA = 2,
    MS_DI_NMEA_GSV = 3,
    MS_DI_NMEA_RMC = 4,
    MS_DI_NMEA_VTG = 5,
```

```
    MS_DI_NMEA_ZDA = 6,

    MS_DI_NUM_ITEMS,                    // Total number of data items

    MS_DI_ALL      = -1                 // Refers to all data items

} MS_DI_INDEX;


/*** IRIG Output services ***/
typedef enum
{
    IO_PORT_0    = 0,
    IO_PORT_MIN  = 0,
    IO_PORT_1    = 1,
    IO_PORT_2    = 2,
    IO_PORT_3    = 3,
    IO_PORT_4    = 4,
    IO_PORT_5    = 5,
    IO_PORT_6    = 6,
    IO_PORT_7    = 7,

    // Add more IRIG Outputs here

    IO_PORT_NUM  = 8                    // 8 Maximum range of unsigned int

} IO_PORT;

/*** E1/T1 Output services ***/

typedef enum
{
    ETO_OUT_OPT_DIFF   = 0,             // Differential output option
    ETO_OUT_OPT_SINGLE = 1              // Single-ended output option

} ETO_OUT_OPT;

typedef enum
{
    ETO_MODE_T1      = 0,               // T1 output mode
    ETO_MODE_E1      = 1,               // E1 output mode
    ETO_MODE_DISABLE = 2                // Disable output

} ETO_MODE;

typedef enum
{
    ETO_T1_ENC_B8ZS = 0,                // B8ZS T1 encoding
    ETO_T1_ENC_AMI  = 1                 // AMI T1 encoding

} ETO_T1_ENC;

typedef enum
{
    ETO_T1_FRM_D4SF        = 0,         // T1 super frame
    ETO_T1_FRM_ESF_CRC6    = 1,         // T1 extended super frame w/CRC6
    ETO_T1_FRM_ESF_NO_CRC6 = 2,         // T1 extended super frame w/o CRC6
    ETO_T1_FRM_AIS         = 3          // T1 alarm Indication Signal

} ETO_T1_FRM;

typedef enum
{
```

```
    ETO_E1_FRM_CRC4    = 0,             // E1 framing w/CRC4
    ETO_E1_FRM_NO_CRC4 = 1,             // E1 framing w/o CRC4
    ETO_E1_FRM_AIS     = 2              // E1 alarm indication signal

} ETO_E1_FRM;

typedef enum
{
    ETO_T1_SSM_PRS  = 0,    // Primary Reference Source
    ETO_T1_SSM_STU  = 1,    // Synchronized - Traceability Unknown
    ETO_T1_SSM_ST2  = 2,    // Stratum 2
    ETO_T1_SSM_TNC  = 3,    // Transit Node Clock
    ETO_T1_SSM_ST3E = 4,    // Stratum 3E
    ETO_T1_SSM_ST3  = 5,    // Stratum 3
    ETO_T1_SSM_SMC  = 6,    // SONET Minimum Clock
    ETO_T1_SSM_ST4  = 7,    // Stratum 4
    ETO_T1_SSM_RES  = 8,    // Reserved for Network Synchronization
    ETO_T1_SSM_DUS  = 9     // Do Not Use for Synchronization

} ETO_T1_SSM;

typedef enum
{
    ETO_E1_SSM_UNK   = 0,   // Synchronized - Traceability Unknown
    ETO_E1_SSM_PRC   = 1,   // Primary Reference Clock (G.811)
    ETO_E1_SSM_SSU_A = 2,   // Synchronization Supply Unit (G.812 Type I)
    ETO_E1_SSM_SSU_B = 3,   // Synchronization Supply Unit (G.812 Type VI)
    ETO_E1_SSM_SEC   = 4,   // Synchronous Equipment Clock (G.813)
    ETO_E1_SSM_DNU   = 5    // Do Not Use

} ETO_E1_SSM;

typedef struct
{
    ETO_OUT_OPT outOpt;             // Output option board type
    ETO_MODE    mode;               // Operational mode
    ETO_T1_ENC  t1Encode;           // T1 encoding type
    ETO_T1_FRM  t1Frame;            // T1 framing type
    ETO_E1_FRM  e1Frame;            // E1 framing type
    int         ssmEn;              // SSM Enable
    ETO_T1_SSM  t1Ssm;              // T1 SSM value
    ETO_E1_SSM  e1Ssm;              // E1 SSM value

} ETO_CFG;

/*** UART configuration ***/
typedef enum
{
    UD_BR_1200   = 1200,
    UD_BR_2400   = 2400,
    UD_BR_4800   = 4800,
    UD_BR_9600   = 9600,
    UD_BR_19200  = 19200,
    UD_BR_38400  = 38400,
    UD_BR_57600  = 57600,
    UD_BR_76800  = 76800,
    UD_BR_115200 = 115200

} UD_BR;

typedef enum
{
    UD_DATA_8 = 0,                  // 8 Data bits
```

```
    UD_DATA_7 = 1,                  // 7 Data bits
    UD_DATA_6 = 2,                  // 6 Data bits
    UD_DATA_5 = 3                   // 5 Data bits

} UD_DATA;

typedef enum
{
    UD_STOP_1  = 0,                 // 1 Stop bit
    UD_STOP_15 = 1,                 // 1.5 Stop bits
    UD_STOP_2  = 2                  // 2 Stop bits

} UD_STOP;

typedef enum
{
    UD_PAR_NONE = 0,                // Parity none
    UD_PAR_ODD  = 1,                // Odd parity
    UD_PAR_EVEN = 2                 // Even parity

} UD_PAR;

typedef struct
{
    UD_BR      br;
    UD_DATA    numbits;
    UD_STOP    stopbits;
    UD_PAR     parity;

} UD_CFG;

/*** Display services ***/
typedef enum
{
    DP_MODE_NORMAL = 0,             // Normal operational mode
    DP_MODE_PHOTO  = 1,             // Photo mode
    DP_MODE_TEST   = 2,             // Display test mode

} DP_MODE;


/*** License Type ***/
typedef enum
{
    DCS_NONE      = 0,
         DCS_MULTIGNSS = 1, // Multi GNSS Option
    DCS_SKYLIGHT  = 2, // SkyLight option

    // Add License type here

    DCS_NOTANOPT = 0xFF // Not an Option type used for the end of list
} DCS_LICENSE_TYPE;

   /*** Constellation Type ***/
typedef enum
{
    GR_NONE_CONST           = 0x00000000, //
         GR_GPS_CONST            = 0x00000001, // GPS Constellation
    GR_GLONASS_CONST        = 0x00000002, // GLONASS Constellation
    GR_GALILEO_CONST        = 0x00000004, // GALILEO Constellation
    GR_COMPASS_BEIDOU_CONST = 0x00000008, // COMPASS/BEIDOU Constellation
         GR_SBAS_CONST           = 0x00000010, // SBAS Constellation
```

```
        // Add Constallation type here
        GR_NUM_CONST
    } GR_CONSTELLATION_TYPE;



    /*** Hardware services ***/
#include "tsync_hw.h"

    /*=========================================================================
    TSYNC MATCH OBJECT
    =========================================================================*/

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


    /*=========================================================================
    TSYNC SATINFO OBJECT
    =========================================================================*/

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

    /*=========================================================================
    TSYNC HEARTBEAT OBJECT
    =========================================================================*/

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

    /*=========================================================================
    TSYNC TIME OBJECT
    =========================================================================*/

    typedef struct TSYNC_TimeObj
    {
        unsigned int years;
        unsigned int doy;
        unsigned int hours;
        unsigned int minutes;
        unsigned int seconds;
        unsigned int ns;
    }
    TSYNC_TimeObj;

    /*=========================================================================
```

```
TSYNC TIME BCD OBJECT
==========================================================================*/

typedef struct TSYNC_TimeBCDObj
{
    unsigned int years;
    unsigned int doy;
    unsigned int hours;
    unsigned int minutes;
    unsigned int seconds;
    unsigned int ms;
    unsigned int us;
}
TSYNC_TimeBCDObj;

/*==========================================================================
TSYNC TIME SECONDS OBJECT
==========================================================================*/

typedef struct TSYNC_TimeSecondsObj
{
    unsigned int seconds;
    unsigned int ns;
}
TSYNC_TimeSecondsObj;

/*==========================================================================
TSYNC HW TIME OBJECT
==========================================================================*/

typedef struct TSYNC_HWTimeObj
{
    TSYNC_TimeObj time;
    unsigned int  bSync;
}
TSYNC_HWTimeObj;

/*==========================================================================
TSYNC HW TIME SECONDS OBJECT
==========================================================================*/

typedef struct TSYNC_HWTimeSecondsObj
{
    TSYNC_TimeSecondsObj time;
    unsigned int         bSync;
}
TSYNC_HWTimeSecondsObj;

/*==========================================================================
TSYNC WAIT OBJECT
==========================================================================*/

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

/*==========================================================================
```

```
TSYNC MEM OBJECT FOR PEEK/POKE
==============================================================================*/

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

/*==============================================================================
TSYNC GET STATUS OBJECT
==============================================================================*/

typedef struct TSYNC_StateObj
{
    unsigned int image;
    unsigned int step;
    unsigned int complete;
}
TSYNC_StateObj;

/*==============================================================================
TSYNC IMAGE ID OBJECT
==============================================================================*/

typedef struct
{
    FS_IMG type;       // Image type
    unsigned int slot; // Image slot
} TSYNC_FSImageIdObj;

/*==============================================================================
TSYNC IMAGE HEADER OBJECT
==============================================================================*/

typedef struct
{
    unsigned int mark; // Image marker
    FS_IMG type; // Image type
    unsigned int len; // Image length
} TSYNC_FSImageHeaderObj;

/*==============================================================================
TSYNC DATA UPDATE OBJECT
==============================================================================*/

typedef struct
{
    FS_IMG type; // Image type
    unsigned char data[TSYNC_DATA_BLOCK_SIZE]; // data block
} TSYNC_UpdateDataObj;

/*==============================================================================
TSYNC END UPDATE OBJECT
==============================================================================*/

typedef struct
{
    FS_IMG type; // Image type
    unsigned char ver[4]; // version
    unsigned int crc; // crc
} TSYNC_UpdateEndObj;
```

```
/*=============================================================================
TSYNC Capability OBJECT
=============================================================================*/

typedef struct TSYNC_CapabilityObj
{
    CI_CAI cai;
    CI_IID iid;
    CI_ACCESS access;
}
TSYNC_CapabilityObj;

/*=============================================================================
TSYNC Capability Page OBJECT
=============================================================================*/

typedef struct TSYNC_CapabilityPageObj
{
    unsigned int pageNum;
    int more;
    TSYNC_CapabilityObj caps[TSYNC_CAPABILITIES_PER_PAGE];
}
TSYNC_CapabilityPageObj;

/*=============================================================================
TSYNC TIME SCALE OBJECT
=============================================================================*/

typedef struct TSYNC_TimeScaleObj
{
    ML_TIME_SCALE scale;
}
TSYNC_TimeScaleObj;

/*=============================================================================
TSYNC TIME SCALE OFFSET OBJECT
=============================================================================*/

typedef struct TSYNC_TimeScaleOffsetObj
{
    ML_TIME_SCALE scale;
    int offset;
}
TSYNC_TimeScaleOffsetObj;

/*=============================================================================
TSYNC TIME SUBSECOND ADJUSTMENT OBJECT
=============================================================================*/

typedef struct TSYNC_TimeSubsecAdjObj
{
    int adjust;
}
TSYNC_TimeSubsecAdjObj;

/*=============================================================================
TSYNC TIME SUBSECOND ADJUSTMENT OBJECT
=============================================================================*/

typedef struct TSYNC_TimeDiscontObj
{
    TSYNC_TimeObj   newTime;
```

```
    TSYNC_TimeObj   effectiveTime;
    int             bActive;
}
TSYNC_TimeDiscontObj;

/*=============================================================================
TSYNC TIME SUBSECOND ADJUSTMENT OBJECT
=============================================================================*/

typedef struct TSYNC_TimeLeapSecondObj
{
    int             offset;
    TSYNC_TimeObj   utcDate;
}
TSYNC_TimeLeapSecondObj;

/*=============================================================================
TSYNC TIME ZONE OFFSET OBJECT
=============================================================================*/

typedef struct TSYNC_TimeZoneOffsetObj
{
    int             tzOffset;
}
TSYNC_TimeZoneOffsetObj;

/*=============================================================================
TSYNC TIME DST POINT OBJECT
=============================================================================*/

typedef struct TSYNC_TimeDSTPointObj
{
    ML_MONTH            month;
    ML_WOM              wom;
    ML_DOW              dow;
    int                 hour;
}
TSYNC_TimeDSTPointObj;

/*=============================================================================
TSYNC TIME DST RULE OBJECT
=============================================================================*/

typedef struct TSYNC_TimeDSTRuleObj
{
    ML_DST_REF              ref;
    TSYNC_TimeDSTPointObj   in;
    TSYNC_TimeDSTPointObj   out;
    uint32_t                offset;
}
TSYNC_TimeDSTRuleObj;

/*=============================================================================
TSYNC TIME YEAR OBJECT
=============================================================================*/

typedef struct TSYNC_TimeYearObj
{
    int             year;
}
TSYNC_TimeYearObj;

/*=============================================================================
```

```
TSYNC TIME DST STATE OBJECT
==========================================================================*/

typedef struct TSYNC_TimeDSTStateObj
{
    int state;
}
TSYNC_TimeDSTStateObj;

/*==========================================================================
TSYNC FS IMAGE OBJECT
==========================================================================*/

typedef struct TSYNC_FSImageObj
{
    FS_IMG         image;
}
TSYNC_FSImageObj;

/*==========================================================================
TSYNC FS CRC OBJECT
==========================================================================*/

typedef struct TSYNC_FSCRCObj
{
    unsigned int         crc;
}
TSYNC_FSCRCObj;

/*==========================================================================
TSYNC FS VERSION OBJECT
==========================================================================*/

typedef struct
{
    char version[4];
} TSYNC_FSVersionObj;

/*==========================================================================
TSYNC ERROR LOG OBJECT
==========================================================================*/

typedef struct
{
    char message[120];
} TSYNC_ErrorLogObj;

/*==========================================================================
TSYNC ALARM OBJECT
==========================================================================*/

typedef struct
{
    LS_ALARM index;
} TSYNC_AlarmObj;

/*==========================================================================
TSYNC FLAG OBJECT
==========================================================================*/

typedef struct
{
    uint32_t flag;
```

```
} TSYNC_FlagObj;

/*=========================================================================
TSYNC LS VERSION OBJECT
=========================================================================*/

typedef struct
{
    char version[7];
} TSYNC_FirmwareVersionObj;

/*=========================================================================
TSYNC LS SERIAL NO OBJECT
=========================================================================*/

typedef struct
{
    char serno[33];
} TSYNC_SerialNoObj;

/*=========================================================================
TSYNC METER HANDLE OBJECT
=========================================================================*/

typedef struct
{
    TSYNC_METER_HANDLE hnd;
} TSYNC_MeterHandle;

/*=========================================================================
TSYNC METER WIN SIZE OBJECT
=========================================================================*/

typedef struct
{
    TSYNC_METER_HANDLE  hnd;
    unsigned int        size;
} TSYNC_MeterWinSizeObj;

/*=========================================================================
TSYNC METER DATA OBJECT
=========================================================================*/

typedef struct
{
    TSYNC_METER_HANDLE  hnd;
    XS_STATE state; // Meter state
    unsigned int size; // Window size
    unsigned int elapsed; // Elapsed duration through the
                                    // window
    float frAccum; // Accumulated frequency error of
                                    // current window
    float frPrev; // Total frequency error of
                                    // previous window
    float phStart; // Starting phase error of current
                                    // window
    float phAccum; // Accumulated phase error of
                                    // current window
    float phPrev; // Total phase error of previous
                                    // window
} TSYNC_MeterDataObj;

/*=========================================================================
```

```
TSYNC METER COMMAND OBJECT
==========================================================================*/

typedef struct
{
    TSYNC_METER_HANDLE  hnd;
    XS_CMD              command;
} TSYNC_MeterCommandObj;

/*==========================================================================
TSYNC SUPERVISOR REFERENCE OBJECT
==========================================================================*/

typedef struct
{
    char time[5];
    char pps[5];
} TSYNC_ReferenceObj;

/*==========================================================================
TSYNC REFERENCE ID OBJECT
==========================================================================*/

typedef struct
{
    char refid[5];
} TSYNC_RefIdObj;

/*==========================================================================
TSYNC SUPERVISOR RESET OBJECT
==========================================================================*/

typedef struct
{
    SS_RESET type;
} TSYNC_ResetObj;

/*==========================================================================
TSYNC REFERENCE MONITOR TABLE TYPE OBJECT
==========================================================================*/

typedef struct
{
    RS_TABLE_TYPE type;
} TSYNC_TableTypeObj;

/*==========================================================================
TSYNC REFERENCE MONITOR TABLE ENTRY OBJECT
==========================================================================*/

typedef struct
{
    int enab;
    unsigned int prio;
    char time[5];
    char pps[5];
} TSYNC_TableEntryObj;

/*==========================================================================
TSYNC REFERENCE MONITOR TABLE OBJECT
==========================================================================*/

typedef struct
```

```
{
    TSYNC_TableEntryObj rows[TSYNC_TABLE_ENTRY_NUM];
} TSYNC_ReferenceTableObj;

/*=============================================================================
TSYNC REFERENCE MONITOR TABLE ENTRY STATE OBJECT
=============================================================================*/

typedef struct
{
    char source[5];
    int timeValid;
    int ppsValid;
} TSYNC_TableEntryStateObj;

/*=============================================================================
TSYNC REFERENCE MONITOR STATE TABLE OBJECT
=============================================================================*/

typedef struct
{
    TSYNC_TableEntryStateObj rows[TSYNC_STATE_TABLE_ENTRY_NUM];
} TSYNC_ReferenceStateTableObj;

/*=============================================================================
TSYNC INITIALIZER MODULE RESULT OBJECT
=============================================================================*/

typedef struct
{
    char module[33];
    TSYNC_ERROR result;
} TSYNC_InitModuleResult;

/*=============================================================================
TSYNC INITIALIZER STATUS OBJECT
=============================================================================*/

typedef struct
{
    unsigned int pageNum;
    int more;
    TSYNC_InitModuleResult results[TSYNC_INIT_RESULT_NUM];
} TSYNC_InitStatusResult;

/*=============================================================================
TSYNC ASCII EVENT MESSAGE STATUS OBJECT
=============================================================================*/

typedef struct
{
    char msg[40];

} TSYNC_AsciiEvtMsg;

/*=============================================================================
TSYNC SUPERVISOR TFOM OBJECT
=============================================================================*/

typedef struct
{
    TFOM tfom;
} TSYNC_TFOMObj;
```

```
/*=============================================================================
TSYNC GPS LLA OBJECT
=============================================================================*/

typedef struct
{
    double lat;
    double lon;
    double alt;
} TSYNC_LLAObj;

/*=============================================================================
TSYNC GPS FIX DATA OBJECT
=============================================================================*/

typedef struct
{
    int nSats;
    float pdop;
    float hdop;
    float vdop;
    float tdop;
    int fom;
    int tfom;
    int herr;
    int verr;
} TSYNC_FixDataObj;

/*=============================================================================
TSYNC GPS SAT INFO OBJECT
=============================================================================*/

typedef struct
{
    unsigned int chnum;
    unsigned int svid;
    unsigned int str;
    int bTraim;
    int bInfix;
    unsigned int flags;
} TSYNC_SatInfoObj;

/*=============================================================================
TSYNC GPS SAT DATA OBJECT
=============================================================================*/

typedef struct
{
    TSYNC_SatInfoObj info[TSYNC_SAT_INFO_NUM];
} TSYNC_SatDataObj;

/*=============================================================================
TSYNC GPS Qualification Log OBJECT
=============================================================================*/

typedef struct
{
    unsigned int val[TSYNC_QUAL_LOG_NUM];
} TSYNC_QualLogObj;

/*=============================================================================
TSYNC MANUFACTURER MODEL OBJECT
```

```
==========================================================================*/

typedef struct
{
    char mfr[17];
    char mdl[17];
} TSYNC_ManModObj;

/*==========================================================================
TSYNC GPS RECEIVER INFO OBJECT
==========================================================================*/

typedef struct
{
    unsigned int len;
    char info[257];
} TSYNC_ReceiverInfoObj;

/*==========================================================================
TSYNC GPS CUSTOM MESSAGE OBJECT
==========================================================================*/

typedef struct
{
    unsigned int len;
    char msg[257];
} TSYNC_CustomMessageObj;

/*==========================================================================
TSYNC GPS RECEIVER PARM OBJECT
==========================================================================*/

typedef struct
{
    unsigned int len;
    GL_PARM      parm;
    unsigned int cfg[TSYNC_PARM_NUM];
} TSYNC_ReceiverParmObj;

/*==========================================================================
TSYNC GPS RECEIVER A-GPS ALMANAC OBJECT
==========================================================================*/

typedef struct
{
    AGPS_ALMANAC alm;
} TSYNC_AlmObj;


/*==========================================================================
TSYNC GPS RECEIVER A-GPS EPHEMERIS OBJECT
==========================================================================*/

typedef struct
{
    AGPS_EPHEMERIS ephm;
} TSYNC_EphmObj;


/*==========================================================================
TSYNC GPS RECEIVER A-GPS SERVER STATE OBJECT
==========================================================================*/
```

```
typedef struct
{
    int state;
} TSYNC_AgpsServerStateObj;


/*=============================================================================
PTP MODULE INFO OBJECT
=============================================================================*/

typedef struct
{
    unsigned int ptpVerisonNumber;
    unsigned int softwareVersion;
    char         hardwareVersion;
    char         filler[3];
    char         softDate[TSYNC_PTP_DATE_STR_LEN];
    char         softTime[9];
} TSYNC_PTPModuleInfoObj;

/*=============================================================================
PTP ETHERNET INTERFACE INFO OBJECT
=============================================================================*/

typedef struct
{
    int dhcpEnabled;
    unsigned char staticIpAddr[4];
    unsigned char netMask[4];
    unsigned char defaultGateway[4];
} TSYNC_PTPEthernetItfObj;

/*=============================================================================
PTP CLOCK SETTINGS OBJECT
=============================================================================*/

typedef struct
{
    int ptpClockRunning;
    int usingExternalClock;
} TSYNC_PTPClkSettingsObj;

/*=============================================================================
PTP UNIT SETTINGS OBJECT
=============================================================================*/

typedef struct
{
    unsigned char clockIdentity[8];
    int oneStepMode;
    int slaveOnly;
    unsigned int domainNumber;
    unsigned int priority1;
    unsigned int priority2;
    int forcedHoldover;
} TSYNC_PTPUnitSettingsObj;

/*=============================================================================
PTP PORT STATE OBJECT
=============================================================================*/

typedef struct
{
```

```
        unsigned int portNumber;
        int portEnabled;
        PTL_PTP_STATE portState;
        int linkConnected;
        int slaveLogAnn;
        int slaveLogSync;
        int slaveLogDelayReq;
        int slaveOneStepMode;
    } TSYNC_PTPPortStateObj;

    /*=============================================================================
    PTP PORT SETTINGS OBJECT
    =============================================================================*/

    typedef struct
    {
        unsigned int portNumber;
        unsigned int annRcptTimeout;
        int logAnnInterval;
        int logSyncInterval;
        int logDelayReqInterval;
        int logPeerDelayReqInterval;
        PTL_DELAY_MECH  delayMechanism;
             unsigned int syncTimeout;
             unsigned int delayRespTimeout;
    } TSYNC_PTPPortSettingsObj;

    /*=============================================================================
    PTP UNICAST SETTINGS MASTER ADD OBJECT YS
    =============================================================================*/
typedef struct
    {
         unsigned char  clockIdentity[8];
              unsigned int   portNumber;
              unsigned char  staticIpAddr[4];
              char           logQuerySalveInterval;
              char               filler1;
              unsigned short duationSlaveContracts;
              char           logAnnSlaveInterval;
              char           logSyncMasterInterval;
              char           logDelayReqSlaveInterval;
              char               filler2;
    } TSYNC_PTPUnctMasterAddObj;

       /*=============================================================================
    PTP UNICAST SLAVE PROPERTIES OBJECT YS
    =============================================================================*/
typedef struct
    {
         int            negoEnabled;
              char           contractAnnState[16];
              unsigned short duationAnnContracts;
              unsigned short delayAnnContracts;
              char           logAnnMsgInterval;
              char           filler1[3];
              char           contractSyncState[16];
              unsigned short duationSyncContracts;
              unsigned short delaySyncContracts;
              char           logSyncMsgInterval;
              char               filler2[3];
              char           contractDelayRespState[16];
              unsigned short duationDelayRespContracts;
              unsigned short delayDelayRespContracts;
```

```
            char           logDelayRespMsgInterval;
            char               filler3[3];
    } TSYNC_PTPUnctSlavePropObj;

       /*=============================================================================
    PTP UNICAST MASTER PROPERTIES OBJECT YS
    =============================================================================*/
typedef struct
    {
         int           negoEnabled;
               unsigned char nbSalveConnected;
               char           filler[3];

    } TSYNC_PTPUnctMasterPropObj;

    /*=============================================================================
    PTP CLOCK QUALITY OBJECT
    =============================================================================*/

    typedef struct
    {
        unsigned int clockClass;
        PTL_CLK_ACC clockAccuracy;
        unsigned int offsetScaledLogVariance;
    } TSYNC_PTPClkQualityObj;

    /*=============================================================================
    PTP TIME PROPERTIES OBJECT
    =============================================================================*/

    typedef struct
    {
        int utcOffset;
        int utcOffsetValid;
        int forwardLeap;
        int backwardLeap;
        int timeTraceable;
        int freqTraceable;
        int ptpTimescale;
        PTL_TIME_SRC timeSource;
    } TSYNC_PTPTimePropObj;

    /*=============================================================================
    PTP PARENT PROPERTIES OBJECT
    =============================================================================*/

    typedef struct
    {
        unsigned char clockIdentity[8];
        unsigned int portNumber;
        int statsCalculated;
        unsigned int observedOSLV;
        unsigned int observedCPCR;
    } TSYNC_PTPParentPropObj;

    /*=============================================================================
    PTP GRANDMASTER PROPERTIES OBJECT
    =============================================================================*/

    typedef struct
    {
        unsigned char clockIdentity[8];
        TSYNC_PTPClkQualityObj clockQuality;
```

```
    unsigned int priority1;
    unsigned int priority2;
} TSYNC_PTPGrandmasterPropObj;

/*=============================================================================
PTP TOD ENABLE OBJECT
=============================================================================*/

typedef struct
{
    int todEnabled;
    unsigned int timeScale;
} TSYNC_PTPTODSettingsObj;

/*=============================================================================
PTP MAC ADDR OBJECT
=============================================================================*/

typedef struct
{
    unsigned char macAddress[8];
} TSYNC_PTPMacAddrObj;

/*=============================================================================
PTP MAC ADDR w/ PASSWORD OBJECT
=============================================================================*/

typedef struct
{
    unsigned char macAddress[8];
    unsigned char pw[12];
} TSYNC_PTPMacAddrPwObj;

/*=============================================================================
PTP USER DESCRIPTION OBJECT
=============================================================================*/

typedef struct
{
    unsigned char deviceName[16];
    unsigned char deviceLocation[16];
} TSYNC_PTPUserDescObj;

/*=============================================================================
PTP DEBUG OUTPUT OBJECT
=============================================================================*/

typedef struct
{
    int debugOutput;
    int debugFilterOutput;
} TSYNC_PTPDebugOutputObj;

/*=============================================================================
PTP ETHERNET STATUS OBJECT
=============================================================================*/

typedef struct
{
    unsigned char ipAddress[4];
    unsigned char netMask[4];
    unsigned char gateway[4];
} TSYNC_PTPEthernetStatusObj;
```

```
/*=============================================================================
 PTP SYNC-E STATUS OBJECT
 ============================================================================*/

 typedef struct
 {
     int enableSyncE;
     int xmitSyncEClock;
          int esmcEnable;
          unsigned char ssmCode;
          char filler[3];
 } TSYNC_PTPSyncEStatusObj;

/*=============================================================================
 PTP ITF OBJECT
 ============================================================================*/

 typedef struct
 {
     PTL_TRANS_PROTO transProto;
     int             masterUnicast;
     int             slaveUnicast;
     unsigned char   ttl;
     char            filler[3];
     int             mgtPortEna;
 } TSYNC_PTPItfObj;
   /*=============================================================================
 PTP VLAN OBJECT       YS
 ============================================================================*/

 typedef struct
 {
          int vLanIntEnable;
          unsigned short vLanID;
          unsigned char priorityCodePoint;
          char          filler;
 } TSYNC_PTPVLANObj;

/*=============================================================================
 PTP FTP ITF OBJECT
 ============================================================================*/

 typedef struct
 {
     int ftpEnable;
 } TSYNC_PTPFtpItfObj;

/*=============================================================================
 PTP STATISTICS OBJECT
 ============================================================================*/

 typedef struct
 {
     int parentStats;
     int clockStats;
 } TSYNC_PTPStatisticsObj;

/*=============================================================================
 PTP CLOCK PROPERTIES OBJECT
 ============================================================================*/

 typedef struct
```

```
{
    long long offFromMaster;
    long long meanPathDelay;
    unsigned short stepsRemoved;
} TSYNC_PTPClockPropertiesObj;

/*=============================================================================
TSYNC IRIG MESSAGE OBJECT
=============================================================================*/

typedef struct
{
    unsigned short subframes[TSYNC_IR_SUBFRAME_NUM];
} TSYNC_IRIGMessageObj;

/*=============================================================================
TSYNC IRIG CONTROL FIELD OBJECT
=============================================================================*/

typedef struct
{
    unsigned short cfData[TSYNC_IR_CFDATA_NUM];
} TSYNC_IRIGCfDataObj;

/*=============================================================================
TSYNC VARIABLE-FREQUENCY OUTPUT CONFIGURATION OBJECT
=============================================================================*/

typedef struct
{
    float min;
    float max;
    float step;
} TSYNC_VPCfgObj;

/*=============================================================================
TSYNC TIME LOCAL CLOCK OBJECT
=============================================================================*/

typedef struct
{
    TSYNC_TimeDSTRuleObj      rule;
    int                       tz;
}
TSYNC_LocalClockObj;

/*=============================================================================
TSYNC ASCII OUTPUT SUBSECOND OBJECT
=============================================================================*/

typedef struct
{
    int             bEnabled;
    unsigned int    pos;
    unsigned int    num;
}
TSYNC_SubSecondObj;

/*=============================================================================
TSYNC OSCILLATOR DISCIPLINING OBJECT
=============================================================================*/

typedef struct
```

```
{
    unsigned char cmd[TSYNC_OSC_DISC_CMD_SIZE];
    unsigned char data[TSYNC_OSC_DISC_DATA_SIZE];
}
TSYNC_OscDiscObj;

/*=============================================================================
TSYNC OSCILLATOR CUSTOM MESSAGE OBJECT
=============================================================================*/

typedef struct
{
    unsigned int len;
    char msg[65];
}
TSYNC_OscCustomMessageObj;

/*=============================================================================
TSYNC SHARED MEMORY OBJECT
=============================================================================*/

typedef struct
{
    unsigned int    seqNum;
    char            data[256];
}
TSYNC_SharedMemoryObj;

/*=============================================================================
TSYNC GENERAL PURPOSE SQUARE OBJECT
=============================================================================*/

typedef struct
{
    int             off;
    unsigned int    per;
    unsigned int    pw;
    EDGE            ae;
}
TSYNC_GPOSquareObj;

/*=============================================================================
TSYNC TIME DATA OBJECT
=============================================================================*/

typedef struct
{
    TSYNC_HWTimeObj   data[TSYNC_TIMESTAMP_DATA_NUM];
}
TSYNC_HWTimeDataObj;

/*=============================================================================
TSYNC OPTION CARD CIS HEADER OBJECT
=============================================================================*/

typedef struct TSYNC_CardInfoStructHdrObj
{
    unsigned int pldId;
    unsigned int pldVer;
    unsigned int rsv1;
    unsigned int rsv2;
    unsigned int num;
}
```

```
TSYNC_CardInfoStructHdrObj;

/*=============================================================================
TSYNC OPTION CARD HEADER OBJECT
=============================================================================*/

typedef struct TSYNC_OptCardHdrObj
{
    unsigned int               id;
    unsigned int               ver;
    TSYNC_CardInfoStructHdrObj cish;
}
TSYNC_OptCardHdrObj;

/*=============================================================================
TSYNC OPTION CARD SLOT/INDEX OBJECT
=============================================================================*/

typedef struct TSYNC_OptCardSlotIdxObj
{
    int          slot;
    unsigned int idx;
}
TSYNC_OptCardSlotIdxObj;

/*=============================================================================
TSYNC OPTION CARD SLOT/FEATURE/LOCAL INSTANCE OBJECT
=============================================================================*/

typedef struct TSYNC_OptCardSlotFeatInstObj
{
    int          slot;
    unsigned int featId;
    unsigned int inst;
}
TSYNC_OptCardSlotFeatInstObj;

/*=============================================================================
TSYNC OPTION CARD HEADER OBJECT
=============================================================================*/

typedef struct TSYNC_OptCardFeatInstObj
{
    unsigned int featId;
    unsigned int inst;
}
TSYNC_OptCardFeatInstObj;

/*=============================================================================
GPL: Clock Identity
=============================================================================*/

typedef struct TSYNC_GprClockId
{
    unsigned char cid[8];
}
TSYNC_GplClockId;

/*=============================================================================
GPL: Priority
=============================================================================*/

typedef struct TSYNC_GplPriority
```

```
{
    unsigned char priority1;
    unsigned char priority2;
}
TSYNC_GplPriority;

/*=============================================================================
GPL: Clock Quality
=============================================================================*/

typedef struct TSYNC_GplClockQual
{
   unsigned int   clockClass;
   unsigned int   clockAcc;
    unsigned int   oSLV;
}
TSYNC_GplClockQual;

/*=============================================================================
GPL: Statistics
=============================================================================*/

typedef struct TSYNC_GplStatistics
{
    int             offFromMaster;
    int             meanPathDelay;
    unsigned short  stepsRemoved;
}
TSYNC_GplStatistics;

/*=============================================================================
GPL: Parent Data
=============================================================================*/

typedef struct TSYNC_GplParentData
{
    TSYNC_GplClockId    parentCid;
    unsigned int        parentPortId;
    unsigned int        parentStatsCalc;
    unsigned int        parentObsOSLV;
    unsigned int        parentObsCPCR;
    TSYNC_GplClockQual  gmQual;
    TSYNC_GplClockId    gmCid;
    unsigned char       gmPortid;
    TSYNC_GplPriority   gmpri;
}
TSYNC_GplParentData;

/*=============================================================================
GPL: Time Properties
=============================================================================*/

typedef struct TSYNC_GplTimeProp
{
    int             utcOffset;
    unsigned int    utcOffsetValid;
    unsigned int    forwardLeap;
    unsigned int    backwardLeap;
    unsigned int    timeTraceable;
    unsigned int    freqTraceable;
    unsigned int    ptpTimescale;
    unsigned int    timeSource;
}
```

```
TSYNC_GplTimeProp;

/*=============================================================================
GPL: User Description
=============================================================================*/

typedef struct TSYNC_GplUserDesc
{
    unsigned char deviceName[16];
    unsigned char deviceLocation[16];
}
TSYNC_GplUserDesc;

/*=============================================================================
GPL: Unicast Master Add Table
=============================================================================*/

typedef struct TSYNC_GplUnctMasterAdd
{
    unsigned char  masterCid[8];
    unsigned int   masterPid;
    unsigned char  masterIPV4[4];
    unsigned char  logQuerySlaveInterval;
    unsigned short durationSlaveContracts;
    unsigned char  logAnnSlaveInterval;
    unsigned char  logSyncMasterInterval;
    unsigned char  logDelayReqSlaveInterval;
}
TSYNC_GplUnctMasterAdd;

/*=============================================================================
GPL: Port Identity
=============================================================================*/

typedef struct TSYNC_GplPortId
{
    unsigned char  cid[8];
    unsigned int   pid;
}
TSYNC_GplPortId;

/*=============================================================================
GPL: Unicast Slave Properties
=============================================================================*/

typedef struct TSYNC_GplUnctSlaveProp
{
    unsigned int   negoEnabled;
    unsigned char  contractAnnState[16];
    unsigned short durationAnnContracts;
    unsigned short delayAnnContracts;
    unsigned int   logAnnMsgInterval;
    unsigned char  contractSyncState[16];
    unsigned short durationSyncContracts;
    unsigned short delaySyncContracts;
    unsigned int   logSyncMsgInterval;
    unsigned char  contractDelayRespState[16];
    unsigned short durationDelayRespContracts;
    unsigned short delayDelayRespContracts;
    unsigned int   logDelayRespMsgInterval;
}
TSYNC_GplUnctSlaveProp;
```

```
/*=============================================================================
GPL: Unicast Master Properties
=============================================================================*/

typedef struct TSYNC_GplUnctMasterProp
{
    unsigned int   negoEnabled;
    unsigned int   nbSlaveConnected;
}
TSYNC_GplUnctMasterProp;

/*=============================================================================
GPL: Unicast Master Configuration
=============================================================================*/

typedef struct TSYNC_GplUnctMasterCfg
{
    unsigned short minSyncInt;
    unsigned short maxSyncDur;
    unsigned short minAnnInt;
    unsigned short maxAnnDur;
    unsigned short minDRqInt;
    unsigned short maxDRqDur;
    unsigned short maxSlaves;
}
TSYNC_GplUnctMasterCfg;

/*=============================================================================
GPL: Message Rates
=============================================================================*/

typedef struct TSYNC_GplMsgRates
{
    unsigned int logAnnInterval;
    unsigned int logSyncInterval;
    unsigned int logDelayReqInterval;
}
TSYNC_GplMsgRates;

/*=============================================================================
GPL: Message Timeouts
=============================================================================*/

typedef struct TSYNC_GplMsgTo
{
    unsigned int annRcptTimeout;
    unsigned int syncTimeout;
    unsigned int delayRespTimeout;
}
TSYNC_GplMsgTo;

/*=============================================================================
GPL: IPV4 Configuration
=============================================================================*/

typedef struct TSYNC_GplIpV4
{
    unsigned char ipV4Addr[4];
    unsigned char netmask[4];
    unsigned char gateway[4];
}
TSYNC_GplIpV4;
```

```
/*=============================================================================
GPL: MAC Address
=============================================================================*/

typedef struct TSYNC_GplMacAddr
{
    unsigned char  macAddr[6];
}
TSYNC_GplMacAddr;

/*=============================================================================
GPL: Sync Ethernet
=============================================================================*/

typedef struct TSYNC_GplSyncEth
{
    unsigned int  enableSyncE;
    unsigned int  esmcEnable;
    unsigned int  esmcSigCtl;
    unsigned int  ssmCode;
}
TSYNC_GplSyncEth;

/*=============================================================================
GPL: VLAN
=============================================================================*/

typedef struct TSYNC_GplVlan
{
    unsigned int   vLanIntEnable;
    unsigned int   vLanID;
    unsigned int   priorityCodePoint;
}
TSYNC_GplVlan;

/*=============================================================================
GPL: Module Info
=============================================================================*/

typedef struct
{
    unsigned int ptpVerisonNumber;
    unsigned int softwareVersion;
    unsigned int hardwareVersion;
    char         softDate[12];
    char         softTime[9];
}
TSYNC_GplModuleInfo;

/*=============================================================================
GPL: Control
=============================================================================*/

typedef struct TSYNC_GplControl
{
    unsigned int   networkEn;
    unsigned int   ptpEn;
}
TSYNC_GplControl;

/*=============================================================================
GPL: Slave Stats
=============================================================================*/
```

```
    typedef struct TSYNC_GplSlaveStats
    {
        unsigned int   slaveNum;
        unsigned int   ptpEn;
        unsigned char  cid[8];
        unsigned int   pid;
        unsigned char  ipV4Addr[4];
        unsigned char  macAddr[6];
        unsigned short syncPeriod;
        unsigned short annPeriod;
        unsigned short dRqPeriod;
        unsigned short syncDur;
        unsigned short syncRem;
        unsigned short annDur;
        unsigned short annRem;
        unsigned short dRqDur;
        unsigned short dRqRem;

    }
    TSYNC_GplSlaveStats;

    /*=============================================================================
    GPL: Slave Summary
    =============================================================================*/
    typedef struct TSYNC_GplSlaveSummary
    {
        unsigned int dRqPerSec;
        unsigned int numUniSlaveConn;
    }
    TSYNC_GplSlaveSummary;

    /*=============================================================================
    GPL: BCASt MECH
    =============================================================================*/

    typedef struct TSYNC_GplBcastMech
    {
        unsigned int   mSync;
        unsigned int   mDelReq;
        unsigned int   uSync;
        unsigned int   uDelReq;
    }
    TSYNC_GplBcastMech;


#define DCS_TYPE_LICENSE_MAX 32
    // The TSYNC_FileOptionObj type is used to store a license option datatypedef
struct
    typedef struct TSYNC_FileOptionObj
    {
        unsigned char   length;        // data length
             unsigned char   filler[3];
             unsigned char   type[DCS_TYPE_LICENSE_MAX];     // License option type
    }
       TSYNC_FileOptionObj;


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

    /* General ================================================================*/
```

```
/*
* Function:    TSYNC_open()
* Description: Open the TSYNC device.
*
* Parameters:
*   IN: *hw          - Handle
*       *deviceName - Name of the device.
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_open(
    TSYNC_BoardHandle *hnd,
    char              *deviceName);

/*
* Function:    TSYNC_close()
* Description: Close the TSYNC device.
*
* Parameters:
*   IN: *hw          - Handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_close(
    TSYNC_BoardHandle hnd);
/*
* Function:    TSYNC_get()
* Description: Generic get accessor.
*
* Parameters:
*   IN:  hnd               - Board handle
*        dest              - destination of the call
*        iid               - item id
*        inPayload         - transaction specific payload
*        inLength          - number of bytes in inPayload
*        maxOutLength      - number bytes allowed in outPayload
*   OUT: outPayload        - passed back transaction specific data
*        actualOutLength   - the actual number of bytes passe back
*                            in outPayload
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_get(
    TSYNC_BoardHandle  handle,
    DEST_ID            dest,
    ITEM_ID            iid,
    void              *inPayload,
    uint32_t           inLength,
    void              *outPayload,
    uint32_t           maxOutLength,
    uint32_t          *actualOutLength);

/*
* Function:    TSYNC_set()
* Description: Generic set accessor.
*
* Parameters:
*   IN:  hnd               - Board handle
*        dest              - destination of the call
```

```
*        iid               - item id
*        inPayload         - transaction specific payload
*        inLength          - number of bytes in inPayload
*        maxOutLength      - number bytes allowed in outPayload
*   OUT: outPayload        - passed back transaction specific data
*        actualOutLength   - the actual number of bytes passe back
*                            in outPayload
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_set(
    TSYNC_BoardHandle  handle,
    DEST_ID            dest,
    ITEM_ID            iid,
    void              *inPayload,
    uint32_t           inLength,
    void              *outPayload,
    uint32_t           maxOutLength,
    uint32_t          *actualOutLength);

/*
* Function:    TSYNC_waitFor()
* Description: Blocking call to wait for specified interrupt.
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - the interrupt type to wait for (GPI/GPO are
*                            indexed)
*        index             - the index of the interrupt (0 for non-indexed
*                            interrupts)
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_waitFor(
    TSYNC_BoardHandle handle,
    INT_TYPE          intType,
    uint32_t          index);

/* Host Agent ================================================================*/

/*
* Function:    TSYNC_HA_getCaps()
* Description: Gets a single page of capability results.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pageNum       - Component Agent Identifier
*   OUT: *pObj         - Pointer to the page of capability results
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HA_getCaps(
    TSYNC_BoardHandle        hnd,
    unsigned int             pageNum,
    TSYNC_CapabilityPageObj *pObj);

/* Discovery and Configuration Service ======================================*/

/*
* Function:    TSYNC_DCS_getCardInfo()
```

```
* Description: Gets information about an option card in a specified slot.
*
* Parameters:
*   IN:  hnd    - Board handle
*        nSlot  - Option card slot
*   OUT: *pObj  - Option card information result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DCS_getCardInfo(
    TSYNC_BoardHandle    hnd,
    unsigned int         nSlot,
    TSYNC_OptCardHdrObj *pObj);

/*
* Function:    TSYNC_DCS_getFeatureByIdx()
* Description: Gets the feature ID and instance for a specified feature
*              index on an option card in a specified slot.
*
* Parameters:
*   IN:  hnd      - Board handle
*        *pInObj  - Slot and index of feature
*   OUT: *pOutObj - Feature ID and instance result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DCS_getFeatureByIdx(
    TSYNC_BoardHandle         hnd,
    TSYNC_OptCardSlotIdxObj  *pInObj,
    TSYNC_OptCardFeatInstObj *pOutObj);

/*
* Function:    TSYNC_DCS_getInstance()
* Description: Gets the system instance of a specified local instance of a
*              specified feature id on an option card in a specified slot.
*
* Parameters:
*   IN:  hnd        - Board handle
*        *pInObj    - Slot, id, and local instance of feature
*   OUT: *nInstance - System instance result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DCS_getInstance(
    TSYNC_BoardHandle             hnd,
    TSYNC_OptCardSlotFeatInstObj *pObj,
    int                          *nInstance);

/*
* Function:    TSYNC_DCS_getSlot()
* Description: Gets the slot number of the option card that contains the
*              specified system instance of a specified feature id.
*
* Parameters:
*   IN:  hnd    - Board handle
*        *pObj  - Id and system instance of feature
*   OUT: *nSlot - Option card slot result
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
    DLL_EXPORT
    TSYNC_ERROR TSYNC_DCS_getSlot(
        TSYNC_BoardHandle         hnd,
        TSYNC_OptCardFeatInstObj *pObj,
        int                      *nSlot);

/*
    * Function:    TSYNC_DCS_getOptions()
    * Description: Gets access of available licensed options
    *
    * Parameters:
    *   IN:  hnd    - Board handle
    *        *pObj  - Id and system instance of feature
    *   OUT: *fileOption - Option
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_DCS_getOptions(
        TSYNC_BoardHandle         hnd,
        TSYNC_FileOptionObj      *fileOption);

/*
    * Function:    TSYNC_DCS_checkOption()
    * Description: Gets access of the availability of a specified
       *              licensed options
    *
    * Parameters:
    *   IN:  hnd    - Board handle
    *        *pObj  - Id and system instance of feature
             licenseType - Type of license LCS_MULTIGNSS or LCS_SKYLIGHT
    *   OUT: *licenseStatus - License available or not
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_DCS_checkOption(
        TSYNC_BoardHandle         hnd,
             int                       licenseType,
        int                      *licenseStatus);

/*
    * Function:    TSYNC_DCS_deleteOption()
    * Description: Delete specified licensed options
    *
    * Parameters:
    *   IN:  hnd    - Board handle
    *        *pObj  - Id and system instance of feature
             licenseType - Type of license LCS_MULTIGNSS or LCS_SKYLIGHT
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_DCS_deleteOption(
        TSYNC_BoardHandle         hnd,
             int                       licenseType);

       /* Upgrade Service ========================================================*/

    /*
    * Function:    TSYNC_US_getState()
    * Description: Retreive the update status.
    *
```

```
* Parameters:
*   IN:  *hw           - Handle
*        *obj          - Pointer to the state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_getState(
    TSYNC_BoardHandle  hnd,
    TSYNC_StateObj    *obj);

/*
* Function:    TSYNC_US_start()
* Description: Begin an image update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*        *obj          - Pointer to the update header information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_start(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageHeaderObj *obj);

/*
* Function:    TSYNC_US_startOC()
* Description: Begin an option card image update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*        *obj1         - Pointer to the update image information
*        *obj2         - Pointer to the update header information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_startOC(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageIdObj     *obj1,
    TSYNC_FSImageHeaderObj *obj2);

/*
* Function:    TSYNC_US_data()
* Description: Send a single data block in the update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*   OUT: *obj          - Pointer to the update data information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_data(
    TSYNC_BoardHandle    hnd,
    TSYNC_UpdateDataObj *obj);

/*
* Function:    TSYNC_US_data()
* Description: Finish the update sequence.
*
* Parameters:
```

```
*   IN:  *hw           - Handle
*   OUT: *obj          - Pointer to the update end information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_end(
    TSYNC_BoardHandle   hnd,
    TSYNC_UpdateEndObj *obj);

/*
* Function:    TSYNC_US_cancel()
* Description: Cancel the update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*   OUT: imageType     - the type of update sequence being cancelled
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_US_cancel(
    TSYNC_BoardHandle hnd,
    FS_IMG            imageType);

/* Clock Service ============================================================*/

/*
* Function:    TSYNC_CS_getTime()
* Description: Get the DOY time from the firmware.  TSYNC_HW_getTime is
*              recommended for faster time reads.
*
* Parameters:
*   IN:  *hw         - Handle
*   OUT: *Timep      - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getTime(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);

/*
* Function:    TSYNC_CS_setTime()
* Description: Set the DOY time.
*
* Parameters:
*   IN:  *hw         - Handle
*        *Timep      - Pointer to the time information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setTime(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);

/*
* Function:    TSYNC_CS_getNextSec()
* Description: Get the DOY time for the next second from the firmware.
*
* Parameters:
```

```
*   IN:  *hw         - Handle
*   OUT: *Timep      - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getNextSec(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);

/*
* Function:    TSYNC_CS_getTimeScale()
* Description: Get the board's current time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT TSYNC_ERROR
    TSYNC_CS_getTimeScale(
    TSYNC_BoardHandle   hnd,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_CS_setTimeScale()
* Description: Set the board's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT TSYNC_ERROR
    TSYNC_CS_setTimeScale(
    TSYNC_BoardHandle   hnd,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_CS_getTimeScaleOff()
* Description: Get the specified time scale's offset from UTC.  Offset is
*              in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the time scale offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT TSYNC_ERROR
    TSYNC_CS_getTimeScaleOff(
    TSYNC_BoardHandle         hnd,
    TSYNC_TimeScaleOffsetObj *pObj);

/*
* Function:    TSYNC_CS_setTimeScaleOff()
* Description: Set the specified time scale's offset from UTC.  Offset is in
*              seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        pObj          - Pointer to the time scale offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT TSYNC_ERROR
    TSYNC_CS_setTimeScaleOff(
    TSYNC_BoardHandle         hnd,
    TSYNC_TimeScaleOffsetObj *pObj);

/*
* Function:    TSYNC_CS_subsecAdj()
* Description: Make a one-time adjustment to the 1PPS on-time point.
*              Adjustment is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the subsecond adjustment information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT TSYNC_ERROR
    TSYNC_CS_subsecAdj(
    TSYNC_BoardHandle       hnd,
    TSYNC_TimeSubsecAdjObj *pObj);

/*
* Function:    TSYNC_CS_getTimeDiscont()
* Description: Get the user time discontinuity.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the time discontinuity result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getTimeDiscont(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDiscontObj *pObj);

/*
* Function:    TSYNC_CS_setTimeDiscont()
* Description: Set the user time discontinuity.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the time discontinuity information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setTimeDiscont(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDiscontObj *pObj);

/*
* Function:    TSYNC_CS_getLeapSec()
* Description: Get the current leap second information.  Offset is in
*              seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the leap seconds result
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getLeapSec(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeLeapSecondObj *pObj);

/*
* Function:    TSYNC_CS_setLeapSec()
* Description: Set a new leap second.  Offset is in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the leap seconds information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setLeapSec(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeLeapSecondObj *pObj);

/*
* Function:    TSYNC_CS_getTimeZoneOff()
* Description: Get the current time zone offset from UTC.  Offset is in
*              seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the time zone offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getTimeZoneOff(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeZoneOffsetObj *pObj);

/*
* Function:    TSYNC_CS_setTimeZoneOff()
* Description: Set the current time zone offset from UTC.  Offset is in
*              seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the time zone offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setTimeZoneOff(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeZoneOffsetObj *pObj);

/*
* Function:    TSYNC_CS_getDstRule()
* Description: Get the current DST rule.  DST Offset is in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the DST rule result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getDstRule(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDSTRuleObj *pObj);

/*
* Function:    TSYNC_CS_setDstRule()
* Description: Set the DST rule.  DST offset is in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the time zone offset rule information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setDstRule(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDSTRuleObj *pObj);

/*
* Function:    TSYNC_CS_getYear()
* Description: Get the current year.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the year result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getYear(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeYearObj *pObj);

/*
* Function:    TSYNC_CS_setYear()
* Description: Set the year.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the year information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setYear(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeYearObj *pObj);

/*
* Function:    TSYNC_CS_getDstState()
* Description: Get the current DST state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the DST state information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
TSYNC_ERROR TSYNC_CS_getDstState(
    TSYNC_BoardHandle      hnd,
    TSYNC_TimeDSTStateObj *pObj);

/*
* Function:    TSYNC_CS_setDstState()
* Description: Set the DST state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the DST state information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setDstState(
    TSYNC_BoardHandle      hnd,
    TSYNC_TimeDSTStateObj *pObj);

/*
* Function:    TSYNC_CS_getTimeSec()
* Description: Get the time in seconds and nanoseconds format.
*              TSYNC_HW_getTimeSec is recommended for faster time reads.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nSeconds      - The seconds result
*        nNanos        - The nanoseconds result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getTimeSec(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nSeconds,
    unsigned int      *nNanos);

/*
* Function:    TSYNC_CS_setTimeSec()
* Description: Set the time in seconds and nanoseconds format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nSeconds      - The seconds information
*        nNanos        - The nanoseconds information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setTimeSec(
    TSYNC_BoardHandle hnd,
    unsigned int      nSeconds,
    unsigned int      nNanos);

/*
* Function:    TSYNC_CS_getTimeBcd()
* Description: Get the time in BCD format.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_getTimeBcd(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeBCDObj  *pObj);

/*
* Function:    TSYNC_CS_setTimeBcd()
* Description: Set the time in BCD format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the time information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_CS_setTimeBcd(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeBCDObj  *pObj);

/* Flash Manager Service ================================================*/

/*
* Function:    TSYNC_FS_getCrc()
* Description: Get the CRC for a particular flash image.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the image information
*   OUT: pObj2         - Pointer to the crc result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FS_getCrc(
    TSYNC_BoardHandle  hnd,
    TSYNC_FSImageObj  *pObj,
    TSYNC_FSCRCObj    *pObj2);

/*
* Function:    TSYNC_FS_calcCrc()
* Description: Calculate the CRC for a particular flash image.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the image information
*   OUT: pObj2         - Pointer to the crc result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FS_calcCrc(
    TSYNC_BoardHandle  hnd,
    TSYNC_FSImageObj  *pObj,
    TSYNC_FSCRCObj    *pObj2);

/*
* Function:    TSYNC_FS_getHeader()
* Description: Get the image header for a particular flash image.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        pObj          - Pointer to the image information
*   OUT: pObj2         - Pointer to the image header result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FS_getHeader(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageObj       *pObj,
    TSYNC_FSImageHeaderObj *pObj2);

/*
* Function:    TSYNC_FS_getVersion()
* Description: Get the image version for a particular flash image.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the image information
*   OUT: pObj2         - Pointer to the image version result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FS_getVersion(
    TSYNC_BoardHandle   hnd,
    TSYNC_FSImageObj   *pObj,
    TSYNC_FSVersionObj *pObj2);

/* Log Service ============================================================*/

/*
* Function:    TSYNC_LS_getErrorLog()
* Description: Get the error log.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the error log result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_LS_getErrorLog(
    TSYNC_BoardHandle  hnd,
    TSYNC_ErrorLogObj *pObj);

/*
* Function:    TSYNC_LS_getAlarm()
* Description: Get the alarm state for the specified alarm.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the alarm index information
*   OUT: pObj2         - Pointer to the alarm flag result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_LS_getAlarm(
    TSYNC_BoardHandle  hnd,
    TSYNC_AlarmObj    *pObj,
    TSYNC_FlagObj     *pObj2);

/*
```

```
* Function:    TSYNC_LS_setAlarm()
* Description: Clear the specified alarm.  (Set 0 to clear)
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the alarm index information
*        pObj2         - Pointer to the alarm flag result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_LS_setAlarm(
    TSYNC_BoardHandle  hnd,
    TSYNC_AlarmObj    *pObj,
    TSYNC_FlagObj     *pObj2);

/*
* Function:    TSYNC_LS_getVersion()
* Description: Get the firmware version string.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the firmware version result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_LS_getVersion(
    TSYNC_BoardHandle         hnd,
    TSYNC_FirmwareVersionObj *pObj);

/*
* Function:    TSYNC_LS_getSerialNo()
* Description: Get the serial number string.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the serial number result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_LS_getSerialNo(
    TSYNC_BoardHandle  hnd,
    TSYNC_SerialNoObj *pObj);

/* Oscillator Monitor Service ================================================*/

/*
* Function:    TSYNC_XS_register()
* Description: Reserve a meter.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the meter handle result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_register(
    TSYNC_BoardHandle  hnd,
    TSYNC_MeterHandle *pObj);
```

```
/*
* Function:    TSYNC_XS_unregister()
* Description: Free the specified meter.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the meter handle information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_unregister(
    TSYNC_BoardHandle  hnd,
    TSYNC_MeterHandle *pObj);

/*
* Function:    TSYNC_XS_getWindowSize()
* Description: Get the specified meter's error data.  Error data is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the window size result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_getWindowSize (
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterWinSizeObj *pObj);

/*
* Function:    TSYNC_XS_setWindowSize()
* Description: Set the specified meter's window size.  Size is in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the window size information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_setWindowSize(
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterWinSizeObj *pObj);

/*
* Function:    TSYNC_XS_getMeterData()
* Description: Get the meter data.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the meter data result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_getMeterData(
    TSYNC_BoardHandle   hnd,
    TSYNC_MeterDataObj *pObj);

/*
* Function:    TSYNC_XS_meterCmd()
```

```
* Description: Send a command to a meter.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the meter command information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XS_meterCmd(
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterCommandObj *pObj);

/* Superviser Service ====================================================*/

/*
* Function:    TSYNC_SS_reset()
* Description: Reset the TSYNC board.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the reset type information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_reset(
    TSYNC_BoardHandle  hnd,
    TSYNC_ResetObj    *pObj);

/*
* Function:    TSYNC_SS_getRef()
* Description: Get currently selected time and 1PPS reference.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getRef(
    TSYNC_BoardHandle   hnd,
    TSYNC_ReferenceObj *pObj);

/*
* Function:    TSYNC_SS_getMaxTfom()
* Description: Get the maximum TFOM.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: tfom          - Pointer to the maximum TFOM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getMaxTfom(
    TSYNC_BoardHandle  hnd,
    TFOM              *tfom);

/*
* Function:    TSYNC_SS_setMaxTfom()
* Description: Set the maximum TFOM.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        tfom          - Pointer to the maximum TFOM information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_setMaxTfom(
    TSYNC_BoardHandle hnd,
    TFOM              tfom);

/*
* Function:    TSYNC_SS_getTfom()
* Description: Get the current TFOM.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: tfom          - Pointer to the current TFOM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getTfom(
    TSYNC_BoardHandle  hnd,
    TFOM              *tfom);

/*
* Function:    TSYNC_SS_getSync()
* Description: Get the current sync state.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: bSync         - Pointer to the current sync state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getSync(
    TSYNC_BoardHandle  hnd,
    int               *bSync);

/*
* Function:    TSYNC_SS_getHoldover()
* Description: Get the current holdover state.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: bHoldover     - Pointer to the current holdover state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getHoldover(
    TSYNC_BoardHandle  hnd,
    int               *bHoldover);

/*
* Function:    TSYNC_SS_getHoldoverTO()
* Description: Get the current holdover timeout.
*
* Parameters:
*   IN:  hnd               - Board handle
```

```
*   OUT: nHoldoverTimeout  - Pointer to the current timeout state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getHoldoverTO(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nHoldoverTimeout);

/*
* Function:    TSYNC_SS_setHoldoverTO()
* Description: Set the current holdover timeout.
*
* Parameters:
*   IN:  hnd               - Board handle
*        nHoldoverTimeout  - Pointer to the current timeout state
*                            information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_setHoldoverTO(
    TSYNC_BoardHandle hnd,
    unsigned int      nHoldoverTimeout);

/*
* Function:    TSYNC_SS_getTimestamp()
* Description: Get the specified state change timestamp.
*
* Parameters:
*   IN:  hnd           - Board handle
*        src           - The timestamp source
*   OUT: pObj          - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getTimestamp(
    TSYNC_BoardHandle  hnd,
    SS_TS_SRC          src,
    TSYNC_TimeObj     *pObj);

/*
* Function:    TSYNC_SS_getTimestampBcd()
* Description: Get the specified state change timestamp in BCD format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        src           - The timestamp source
*   OUT: pObj          - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getTimestampBcd(
    TSYNC_BoardHandle  hnd,
    SS_TS_SRC          src,
    TSYNC_TimeBCDObj  *pObj);

/*
* Function:    TSYNC_SS_getTimestampSec()
* Description: Get the specified state change timestamp in seconds format.
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*        src           - The timestamp source
*   OUT: nSeconds      - Pointer to the seconds time result
*        nNanos        - Pointer to the nanoseconds time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getTimestampSec(
    TSYNC_BoardHandle  hnd,
    SS_TS_SRC          src,
    unsigned int      *nSeconds,
    unsigned int      *nNanos);

/*
* Function:    TSYNC_SS_getUptime()
* Description: Get the board's total uptime in minutes.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nUptime       - Pointer to the uptime result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getUptime(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nUptime);

/*
* Function:    TSYNC_SS_getFreeRun()
* Description: Get the current freerun state.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: bFreerun      - The freerun result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_SS_getFreeRun(
    TSYNC_BoardHandle  hnd,
    int               *bFreerun);

/* Reference Monitor Service ================================================*/

/*
* Function:    TSYNC_RS_getTable()
* Description: Get specified reference priority table.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the table type information
*   OUT: pObj2         - Pointer to the reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getTable(
    TSYNC_BoardHandle        hnd,
    TSYNC_TableTypeObj      *pObj,
    TSYNC_ReferenceTableObj *pObj2);
```

```
/*
* Function:    TSYNC_RS_getBestRef()
* Description: Get current best working priority table entry.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the table entry result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getBestRef(
    TSYNC_BoardHandle    hnd,
    TSYNC_TableEntryObj *pObj);

/*
* Function:    TSYNC_RS_getEntry()
* Description: Get working priority table entry by index.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Table entry index
*   OUT: pObj          - Pointer to the table entry result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getEntry(
    TSYNC_BoardHandle    hnd,
    int                  index,
    TSYNC_TableEntryObj *pObj);

/*
* Function:    TSYNC_RS_addEntry()
* Description: Add an entry to the working priority table.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the table entry information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_addEntry(
    TSYNC_BoardHandle    hnd,
    TSYNC_TableEntryObj *pObj);

/*
* Function:    TSYNC_RS_setFactDef()
* Description: Reset the working priority table to the factory priority
*              table.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_setFactDef(
    TSYNC_BoardHandle hnd);

/*
```

```
* Function:    TSYNC_RS_setUserDef()
* Description: Reset the reference table to the user default settings, if
*              saved user priority table exists.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_setUserDef(
    TSYNC_BoardHandle hnd);

/*
* Function:    TSYNC_RS_saveUserDef()
* Description: Save the working priority table to the user priority table.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_saveUserDef(
    TSYNC_BoardHandle hnd);

/*
* Function:    TSYNC_RS_deleteEntry()
* Description: Delete a working priority table entry by index.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Index of table entry to delete
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_deleteEntry(
    TSYNC_BoardHandle hnd,
    unsigned int      index);

/*
* Function:    TSYNC_RS_getPriority()
* Description: Get specified working priority table entry's priority.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Table entry index
*   OUT: priority      - Pointer to the priority result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getPriority(
    TSYNC_BoardHandle  hnd,
    unsigned int       index,
    unsigned int      *priority);

/*
* Function:    TSYNC_RS_setPriority()
* Description: Set specified working priority table entry's priority.
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*        index         - Table entry index
*        priority      - Pointer to the priority information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_setPriority(
    TSYNC_BoardHandle hnd,
    unsigned int      index,
    unsigned int      priority);

/*
* Function:    TSYNC_RS_getEnable()
* Description: Get specified working priority table entry's enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Table entry index
*   OUT: enabled       - Pointer to the enabled result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getEnable(
    TSYNC_BoardHandle  hnd,
    unsigned int       index,
    unsigned int      *enabled);

/*
* Function:    TSYNC_RS_setEnable()
* Description: Set specified working priority table entry's enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Table entry index
*        enabled       - Pointer to the enabled information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_setEnable(
    TSYNC_BoardHandle hnd,
    unsigned int      index,
    unsigned int      enabled);

/*
* Function:    TSYNC_RS_getStateTable()
* Description: Get the reference validity state table.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the reference state table result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_RS_getStateTable(
    TSYNC_BoardHandle             hnd,
    TSYNC_ReferenceStateTableObj *pObj);

/* Initializer Service ================================================*/
```

```
/*
* Function:    TSYNC_IN_getStatus()
* Description: Get the board's initialization results.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pageNum       - Table entry index
*   OUT: pObj          - Pointer to the initialization status result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IN_getStatus(
    TSYNC_BoardHandle       hnd,
    unsigned int            pageNum,
    TSYNC_InitStatusResult *pObj);

/* General-Purpose Input Component ======================================*/

/*
* Function:    TSYNC_GI_getValue()
* Description: Get the specified GPI's current input value.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*   OUT: bEnabled      - The value result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getValue(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    int               *bEnabled);

/*
* Function:    TSYNC_GI_getEdge()
* Description: Get the GPI's trigger edge used when detecting
*              input changes.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*   OUT: edge          - The edge result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getEdge(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    EDGE              *edge);

/*
* Function:    TSYNC_GI_setEdge()
* Description: Set the GPI's trigger edge used when detecting
*              input changes.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*        edge          - The edge information
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_setEdge(
    TSYNC_BoardHandle hnd,
    ID_PIN            index,
    EDGE              edge);

/*
* Function:    TSYNC_GI_getTsEnable()
* Description: Get the GPI's timestamp enable state
*              used when time stamping input changes.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*   OUT: bEnable       - The enabled result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getTsEnable(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    int               *bEnable);

/*
* Function:    TSYNC_GI_setTsEnable()
* Description: Set the GPI's timestamp enable state
*              used when time stamping input changes.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*        bEnable       - The enabled information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_setTsEnable(
    TSYNC_BoardHandle hnd,
    ID_PIN            index,
    int               bEnable);

/*
* Function:    TSYNC_GI_getNumInst()
* Description: Get number of GPIO Inputs present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_GI_getTsType()
* Description: Get Timestamp Type of a specified GI instance.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*   OUT: bType         - The timestamp type
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getTsType(
    TSYNC_BoardHandle hnd,
    ID_PIN index,
    int *bType);

/*
* Function:    TSYNC_GI_getAsciiTs()
* Description: Get the latest ASCII Timestamp.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*   OUT: pObj          - The latest ASCII timestamp
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_getAsciiTs(
    TSYNC_BoardHandle hnd,
    ID_PIN index,
    TSYNC_AsciiEvtMsg *pObj);

/*
* Function:    TSYNC_GI_tsClear()
* Description: Clears the ASCII TImestamp buffer.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The input index
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GI_tsClear(
    TSYNC_BoardHandle hnd,
    ID_PIN index);


/* Host Reference Component ================================================*/

/*
* Function:    TSYNC_HR_getValidity()
* Description: Get the reference validity from the Host.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_HR_setValidity()
* Description: Set the reference validity of the Host.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bTimeValid    - The time reference information
*        bPpsValid     - The pps reference information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_setValidity(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               bTimeValid,
    int               bPpsValid);

/*
* Function:    TSYNC_HR_setTime()
* Description: Set the time from the Host.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        TSYNC_TimeObj - The time information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_setTime(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_TimeObj     *pObj);

/*
* Function:    TSYNC_HR_getLocal()
* Description: Get the ASCII reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd                - Board handle
*        nInstance          - The instance number
*   OUT: TSYNC_LocalClockObj - The local clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_HR_setLocal()
```

```
* Description: Set the ASCII reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd                - Board handle
*        nInstance          - The instance number
*        TSYNC_LocalClockObj - The local clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_HR_getTimeScale()
* Description: Get the Host reference's time scale.
*
* Parameters:
*   IN:  hnd                - Board handle
*        nInstance          - The instance number
*   OUT: TSYNC_LocalClockObj - The time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_HR_setTimeScale()
* Description: Set the Host reference's time scale.
*
* Parameters:
*   IN:  hnd                - Board handle
*        nInstance          - The instance number
*        TSYNC_LocalClockObj - The time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_HR_getRefId()
* Description: Get reference identifier for a Host reference instance.
*
* Parameters:
*   IN:  hnd            - Board handle
*   OUT: TSYNC_RefIdObj - The reference identifier result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_getRefId(
    TSYNC_BoardHandle  hnd,
```

```
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/*
* Function:    TSYNC_HR_getNumInst()
* Description: Get number of Host reference instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/* 1PPS Reference Component ================================================*/

/*
* Function:    TSYNC_PR_getOffset()
* Description: Get the 1PPS reference's input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_PR_setOffset()
* Description: Get the 1PPS reference's 1PPS input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_PR_getEdge()
* Description: Get the 1PPS reference's active edge setting.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
```

```
*   OUT: edge          - The edge result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_getEdge(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    EDGE              *edge);

/*
* Function:    TSYNC_PR_setEdge()
* Description: Set the 1PPS reference's active edge setting.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        edge          - The edge information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_setEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    EDGE              edge);

/*
* Function:    TSYNC_PR_getValidity()
* Description: Get the 1PPS validity structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_PR_getNumInst()
* Description: Get number of PPS reference instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_PR_getRefId()
```

```
* Description: Get reference identifier for a PPS reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/* LED Control Component ================================================*/

/*
* Function:    TSYNC_EC_getMode()
* Description: Get the LED usage mode state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The LED index
*   OUT: mode          - The usage mode
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EC_getMode(
    TSYNC_BoardHandle  hnd,
    LE_INDEX           index,
    EC_MODE            *mode);

/*
* Function:    TSYNC_EC_setMode()
* Description: Set the LED usage mode state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The LED index
*        mode          - The usage mode
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EC_setMode(
    TSYNC_BoardHandle hnd,
    LE_INDEX          index,
    EC_MODE           mode);

/*
* Function:    TSYNC_EC_getState()
* Description: Get the LED display state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The LED index
*   OUT: state         - The display state
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EC_getState(
    TSYNC_BoardHandle  hnd,
```

```
    LE_INDEX           index,
    EC_STATE          *state);

/*
* Function:    TSYNC_EC_setState()
* Description: Set the LED display state.  Settable only in manual LED mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - The LED index
*        state         - The display state
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EC_setState(
    TSYNC_BoardHandle hnd,
    LE_INDEX          index,
    EC_STATE          state);

/* ASCII Reference Component ================================================*/

/*
* Function:    TSYNC_AR_getOffset()
* Description: Get the ASCII reference's input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_AR_setOffset()
* Description: Get the ASCII reference's 1PPS input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_AR_getValidity()
* Description: Get the ASCII reference validity structure.
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_AR_getUartCfg()
* Description: Get the ASCII reference UART configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pCfg          - The UART configuration
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getUartCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    UD_CFG            *pCfg);

/*
* Function:    TSYNC_AR_setUartCfg()
* Description: Set the ASCII reference UART configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pCfg          - The UART configuration
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setUartCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    UD_CFG            *pCfg);

/*
* Function:    TSYNC_AR_getLeapFlag()
* Description: Get the ASCII reference leap pending flag.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bLeap         - The leap pending flag
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getLeapFlag(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
```

```
    unsigned int      *bLeap);

/*
* Function:    TSYNC_AR_getLocal()
* Description: Get the ASCII reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_AR_setLocal()
* Description: Set the ASCII reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_AR_getTimeScale()
* Description: Get the ASCII reference's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_AR_setTimeScale()
* Description: Set the ASCII reference's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_AR_getRefId()
* Description: Get reference identifier for an ASCII reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/*
* Function:    TSYNC_AR_getFormat()
* Description: Get the ASCII time code format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_FMT            *format);

/*
* Function:    TSYNC_AR_setFormat()
* Description: Set the ASCII time code format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    AL_FMT            format);

/*
* Function:    TSYNC_AR_getNumInst()
* Description: Get number of ASCII reference instances present in the
*              system.
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_AR_getPpsSrc()
* Description: Get the ASCII PPS Source.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: AL_SRC        - The 1PPS Source
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_getPpsSrc(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_PPS            *pps);

/*
* Function:    TSYNC_AR_setPpsSrc()
* Description: Set the ASCII PPS Source.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The 1PPS Source
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AR_setPpsSrc(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    AL_PPS            pps);

/* Frequency Reference Component ========================================*/

/*
* Function:    TSYNC_FR_getValidity()
* Description: Get the frequency validity structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
```

```
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_FR_getFreq()
* Description: Get the frequency of a frequency reference.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *freq);

/*
* Function:    TSYNC_FR_setFreq()
* Description: Set the frequency of a frequency reference.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        freq          - The frequency information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_setFreq(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      freq);

/*
* Function:    TSYNC_FR_getMode()
* Description: Get the reference mode of a frequency reference.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The reference mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    FR_MODE           *mode);

/*
* Function:    TSYNC_FR_setMode()
* Description: Set the reference mode of a frequency reference.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mode          - The reference mode information
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_setMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    FR_MODE           mode);

/*
* Function:    TSYNC_FR_getNumInst()
* Description: Get number of frequency reference instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_FR_getRefId()
* Description: Get reference identifier for a frequency reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/* GPS Reference Component ================================================*/

/*
* Function:    TSYNC_GR_getOffset()
* Description: Get the GPS reference's 1PPS input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_GR_setOffset()
* Description: Set the GPS reference's 1PPS input offset.  Offset is in
*              nanoseconds from -500 msec to +500 msec.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_GR_getValidity()
* Description: Get the GPS validity structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_GR_getPosition()

* Description: Get the GPS position.  Latitude and longitude are in radians.
*              Altitude is in meters.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the position result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getPosition(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_LLAObj      *pObj);

/*
* Function:    TSYNC_GR_setPosition()
* Description: Set the GPS position.  Latitude and longitude are in radians.
*              Altitude is in meters.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the position information
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setPosition(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_LLAObj      *pObj);

/*
* Function:    TSYNC_GR_getMode()
* Description: Get the GPS receiver mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The receiver mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    GL_MODE           *mode,
    GL_DYN            *dyn);

/*
* Function:    TSYNC_GR_setMode()
* Description: Set the GPS receiver mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mode          - The receiver mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    GL_MODE           mode,
    GL_DYN            dyn);

/*
* Function:    TSYNC_GR_getDynamics()
* Description: Get the GPS dynamics mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The dynamics mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getDynamics(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    GL_DYN            *mode);

/*
* Function:    TSYNC_GR_setDynamics()
* Description: Set the GPS dynamics mode.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mode          - The dynamics mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setDynamics(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    GL_DYN            mode);

/*
* Function:    TSYNC_GR_getFixData()
* Description: Get the GPS position fix data.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the position fix data result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getFixData(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_FixDataObj  *pObj);

/*
* Function:    TSYNC_GR_getSatData()
* Description: Get the GPS satellite data.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the satellite data result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getSatData(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_SatDataObj  *pObj);

/*
* Function:    TSYNC_GR_getSurveyProg()
* Description: Get the GPS survey progress.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nProgress     - The survey progress result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getSurveyProg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
```

```
    unsigned int      *nProgress);

/*
* Function:    TSYNC_GR_getMfrMdl()
* Description: Get the GPS receiver manufacturer and model.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the manufacturer and model result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getMfrMdl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_ManModObj   *pObj);

/*
* Function:    TSYNC_GR_getRcvInfo()
* Description: Get the GPS receiver info.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the receiver info result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getRcvInfo(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_ReceiverInfoObj *pObj);

/*
* Function:    TSYNC_GR_getCustom()
* Description: Get the last unhandled GPS message.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the custom message result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getCustom(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_CustomMessageObj *pObj);

/*
* Function:    TSYNC_GR_setCustom()
* Description: Send a custom message to the GPS.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the custom message result
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setCustom(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_CustomMessageObj *pObj);

/*
* Function:    TSYNC_GR_getNumInst()
* Description: Get number of GPS references instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_GR_delPos()
* Description: Clear any position information that is stored in persistent
*              memory inside the GPS receiver.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_delPos(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance);

/*
* Function:    TSYNC_GR_getRefId()
* Description: Get reference identifier for a GPS reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/*
* Function:    TSYNC_GR_Reset()
* Description: Reset the GPS receiver.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        reset         - The reset type
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
    DLL_EXPORT
    TSYNC_ERROR TSYNC_GR_reset(
        TSYNC_BoardHandle hnd,
        unsigned int      nInstance,
        GL_RESET          reset);

    /*
    * Function:    TSYNC_GR_getAntenna()
    * Description: Get the GPS receiver antenna status.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        status        - Pointer to the status result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_GR_getAntenna(
        TSYNC_BoardHandle hnd,
        unsigned int nInstance,
        GL_ANT_STATUS *status);
/*
    * Function:    TSYNC_GR_getConstSel()
    * Description: Get the Constellation selection.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - Pointer to the Constellation selection
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_GR_getConstSel(
        TSYNC_BoardHandle       hnd,
        unsigned int            nInstance,
        unsigned int            *constellation);

    /*
    * Function:    TSYNC_GR_setConstSel()
    * Description: Send the Constellation selection.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Pointer to the Constellation selection
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_GR_setConstSel(
        TSYNC_BoardHandle       hnd,
        unsigned int            nInstance,
        unsigned int            constellation);
    /*
    * Function:    TSYNC_GR_getParameter()
    * Description: Get a GPS Receiver specific parameter or control response.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - Pointer to the GPS specific message
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getParameter(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_ReceiverParmObj   *pObj);

/*
* Function:    TSYNC_GR_setParameter()
* Description: Send a GPS Receiver specific parameter or control message.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the GPS specific message
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_setParameter(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_ReceiverParmObj   *pObj);

/*
* Function:    TSYNC_GR_getQualLog()
* Description: Get the GPS satellite Qualification Log for the last hour.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the qualification log data result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GR_getQualLog(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_QualLogObj  *pObj);

/*
* Function:    TSYNC_GR_getAlm()
* Description: Get the almanac
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the returned almanac
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_getAlm(
       TSYNC_BoardHandle      hnd,
       unsigned int           nInstance,
       TSYNC_AlmObj          *pObj);

/*
* Function:    TSYNC_GR_setAlm()
* Description: Set the almanac
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the almanac to be sent
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_setAlm(
       TSYNC_BoardHandle      hnd,
       unsigned int           nInstance,
       TSYNC_AlmObj          *pObj);

/*
* Function:    TSYNC_GR_getEphm()
* Description: Get the ephemeris
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the returned ephemeris
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_getEphm(
       TSYNC_BoardHandle      hnd,
       unsigned int           nInstance,
       TSYNC_EphmObj          *pObj);

/*
* Function:    TSYNC_GR_setEphm()
* Description: Set the ephemeris
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the ephemeris to be sent
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_setEphm(
       TSYNC_BoardHandle      hnd,
       unsigned int           nInstance,
       TSYNC_EphmObj          *pObj);

/*
* Function:    TSYNC_GR_setAgpsTime()
* Description: Apply the current time in the timing system to the GPS
*              receiver.  Coarsly accurate time, together with position,
*              allows the receiver to use A-GPS almanac and ephemeris data.
*              The timing system time can be set by NTP or by hand.  The
*              UTC-GPS offset also needs to be set before setting time.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
   TSYNC_ERROR TSYNC_GR_setAgpsTime(
       TSYNC_BoardHandle      hnd,
       unsigned int           nInstance);

/*
* Function:    TSYNC_GR_getAgpsServerState()
* Description: Get whether the GPS component is collecting A-GPS data
*              or not.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the returned A-GPS server state
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_getAgpsServerState(
       TSYNC_BoardHandle         hnd,
       unsigned int              nInstance,
       TSYNC_AgpsServerStateObj *pObj);

/*
* Function:    TSYNC_GR_setAgpsServerState()
* Description: Set the GPS component to collect A-GPS data or not.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the A-GPS server state to be sent
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
   TSYNC_ERROR TSYNC_GR_setAgpsServerState(
       TSYNC_BoardHandle         hnd,
       unsigned int              nInstance,
       TSYNC_AgpsServerStateObj *pObj);

/* IRIG Reference Component ================================================*/

/*
* Function:    TSYNC_IR_getOffset()
* Description: Get the IRIG reference's 1PPS input offset.  Offset is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_IR_setOffset()
* Description: Set the IRIG reference's 1PPS input offset.  Offset is in
*              nanoseconds from -500 msec to +500 msec.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_IR_getValidity()
* Description: Get the IRIG validity structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_IR_getMode()
* Description: Get the IRIG mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The receiver mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_MODE           *mode);

/*
* Function:    TSYNC_IR_setMode()
* Description: Set the IRIG mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mode          - The receiver mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setMode(
```

```
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_MODE           mode);

/*
* Function:    TSYNC_IR_getFormat()
* Description: Get the IRIG format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FMT            *format);

/*
* Function:    TSYNC_IR_setFormat()
* Description: Set the IRIG format.  Settable only when in manual mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FMT            format);

/*
* Function:    TSYNC_IR_getMod()
* Description: Get the IRIG modulation.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mod           - The modulation result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getMod(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_MOD            *mod);

/*
* Function:    TSYNC_IR_setMod()
* Description: Set the IRIG modulation.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mod           - The modulation
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setMod(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_MOD            mod);

/*
* Function:    TSYNC_IR_getFreq()
* Description: Get the IRIG carrier frequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FRQ            *freq);

/*
* Function:    TSYNC_IR_setFreq()
* Description: Set the IRIG carrier frequency.  Settable only when in
*              manual mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        freq          - The frequency information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setFreq(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FRQ            freq);

/*
* Function:    TSYNC_IR_getCodedExpr()
* Description: Get the IRIG Coded Expression.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: ce            - The Coded Expression result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getCodedExpr(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CE             *ce);

/*
* Function:    TSYNC_IR_setCodedExpr()
```

```
* Description: Set the IRIG Coded Expression.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        ce            - The Coded Expression information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setCodedExpr(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_CE             ce);

/*
* Function:    TSYNC_IR_getCtrlField()
* Description: Get the IRIG Control Field.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: cf            - The Control Field result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getCtrlField(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CF             *cf);

/*
* Function:    TSYNC_IR_setCtrlField()
* Description: Set the IRIG Control Field.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        cf            - The Control Field information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setCtrlField(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_CF             cf);

/*
* Function:    TSYNC_IR_getMessage()
* Description: Get the latest IRIG input message.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the message result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getMessage(
    TSYNC_BoardHandle     hnd,
```

```
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);

/*
* Function:    TSYNC_IR_setMessage()
* Description: Set the latest IRIG input message.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the message information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setMessage(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);

/*
* Function:    TSYNC_IR_getNumInst()
* Description: Get number of IRIG references instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_IR_getCfData()
* Description: Get the latest IRIG Control Field data received.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the control field information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getCfData(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_IRIGCfDataObj *pObj);

/*
* Function:    TSYNC_IR_getLocal()
* Description: Get the IRIG reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_IR_setLocal()
* Description: Set the IRIG reference's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_IR_getTimeScale()
* Description: Get the IRIG reference's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success

*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_IR_setTimeScale()
* Description: Set the IRIG reference's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_IR_getRefId()
```

```
* Description: Get reference identifier for an IRIG reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/* STANAG/HaveQuick Reference Component ==================================*/

/*
* Function:    TSYNC_QR_getOffset()
* Description: Get the STANAG/HaveQuick reference's 1PPS input offset.
*              Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_QR_setOffset()
* Description: Set the STANAG/HaveQuick reference's 1PPS input offset.
*              Offset is in nanoseconds from -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_QR_getExtOffset()
* Description: Get the STANAG/HaveQuick extended reference's 1PPS input offset.
*              Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getExtOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_QR_setExtOffset()
* Description: Set the STANAG/HaveQuick extended reference's 1PPS input offset.
*              Offset is in nanoseconds from -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setExtOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_QR_getValidity()
* Description: Get the STANAG/HaveQuick validity structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTimeValid    - The time reference result
*        bPpsValid     - The pps reference result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);

/*
* Function:    TSYNC_QR_getFormat()
* Description: Get the STANAG/HaveQuick format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    QL_FMT            *format);

/*
* Function:    TSYNC_QR_setFormat()
* Description: Set the STANAG/HaveQuick format.
```

```
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *        format        - The format information
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_QR_setFormat(
     TSYNC_BoardHandle hnd,
     unsigned int      nInstance,
     QL_FMT            format);

/*
 * Function:    TSYNC_QR_getExtFormat()
 * Description: Get the STANAG/HaveQuick format.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *   OUT: format        - The format result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_QR_getExtFormat(
     TSYNC_BoardHandle  hnd,
     unsigned int       nInstance,
     QL_FMT            *format);

 /*
 * Function:    TSYNC_QR_setExtFormat()
 * Description: Set the STANAG/HaveQuick format.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *        format        - The format information
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_QR_setExtFormat(
     TSYNC_BoardHandle hnd,
     unsigned int      nInstance,
     QL_FMT            format);

 /*
 * Function:    TSYNC_QR_getLocal()
 * Description: Get the STANAG/HaveQuick reference's local time zone and DST
 *              rule.  Timezone and DST offsets are in seconds.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *   OUT: pObj          - Pointer to the Local Clock result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_QR_getLocal(
     TSYNC_BoardHandle    hnd,
```

```
        unsigned int         nInstance,
        TSYNC_LocalClockObj *pObj);

    /*
    * Function:    TSYNC_QR_setLocal()
    * Description: Set the STANAG/HaveQuick reference's local time zone and DST
    *              rule.  Timezone and DST offsets are in seconds.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Pointer to the Local Clock information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_setLocal(
        TSYNC_BoardHandle    hnd,
        unsigned int         nInstance,
        TSYNC_LocalClockObj *pObj);

    /*
    * Function:    TSYNC_QR_getTimeScale()
    * Description: Get the STANAG/HaveQuick reference's time scale.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - Pointer to the time scale result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_getTimeScale(
        TSYNC_BoardHandle   hnd,
        unsigned int        nInstance,
        TSYNC_TimeScaleObj *pObj);

    /*
    * Function:    TSYNC_QR_setTimeScale()
    * Description: Set the STANAG/HaveQuick reference's time scale.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Pointer to the time scale information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_setTimeScale(
        TSYNC_BoardHandle   hnd,
        unsigned int        nInstance,
        TSYNC_TimeScaleObj *pObj);

/*
    * Function:    TSYNC_QR_getExtTimeScale()
    * Description: Get the STANAG/HaveQuick Extended reference's time scale.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - Pointer to the time scale result
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getExtTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_QR_setExtTimeScale()
* Description: Set the STANAG/HaveQuick Extended reference's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setExtTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

   /*
* Function:    TSYNC_QR_getRefId()
* Description: Get reference identifier for a STANAG/HaveQuick reference
*              instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/*
* Function:    TSYNC_QR_getNumInst()
* Description: Get number of STANAG/HaveQuick reference instances present
*              in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_QR_getTfd()
* Description: Set the STANAG/HaveQuick reference's time fault discrete.
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: tfd           - The time fault discrete information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getTfd(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *tfd);

/*
* Function:    TSYNC_QR_setTfd()
* Description: Set the STANAG/HaveQuick reference's time fault discrete.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        tfd           - The time fault discrete result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setTfd(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      tfd);

/*
* Function:    TSYNC_QR_getTfdState()
* Description: Set the STANAG/HaveQuick reference's time fault discrete state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: tfd           - The time fault discrete state information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getTfdState(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *tfd);

/*
* Function:    TSYNC_QR_setTfdState()
* Description: Set the STANAG/HaveQuick reference's time fault discrete state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        tfd           - The time fault discrete state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setTfdState(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      tfd);
```

```
    /*
* Function:    TSYNC_QR_getBs()
* Description: Set the STANAG/HaveQuick reference's Bit synchnonization.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bs            - The Bit synchnonization information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getBs(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *bs);

/*
* Function:    TSYNC_QR_setRefSelection()
* Description: Set the STANAG/HaveQuick reference's Ref Selection.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        refsel        - The Reference selection result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setRefSelection(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      refsel);
    /*
* Function:    TSYNC_QR_getRefSelection()
* Description: Set the STANAG/HaveQuick reference's Ref Selection.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: refsel        - The Reference Selection information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getRefSelection(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *refsel);

    /*
* Function:    TSYNC_QR_getSelectedRef()
* Description: Set the STANAG/HaveQuick reference's Selected Ref.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: refsel        - The Reference Selection information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getSelectedRef(
```

```
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *refsel);

/*
* Function:    TSYNC_QR_setBs()
* Description: Set the STANAG/HaveQuick reference's Bit synchnonization.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bs            - The Bit synchnonization result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setBs(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      bs);

/*
* Function:    TSYNC_QR_getTfom()
* Description: Get the STANAG/HaveQuick reference's TFOM.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: tfom          - The TFOM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getTfom(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *tfom);
/*
* Function:    TSYNC_QR_getExtTfom()
* Description: Get the STANAG/HaveQuick extended reference's TFOM.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: tfom          - The TFOM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getExtTfom(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int     *tfom);

/*
* Function:    TSYNC_QR_getElecType()
* Description: Get the reference's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*   OUT: et            - The electrical type result
*
```

```
 * Returns: (TSYNC_SUCCESS) Success
 */
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getElecType(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE         *et);

/*
* Function:    TSYNC_QR_setElecType()
* Description: Set the reference's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        et            - The electrical type information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setElecType (
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE          et);

/*
 * Function:    TSYNC_QR_getExtElecType()
 * Description: Get the extended reference's electrical type.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstances    - The instance number
 *   OUT: et            - The electrical type result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getExtElecType(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE         *et);

/*
* Function:    TSYNC_QR_setExtElecType()
* Description: Set the extended reference's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        et            - The electrical type information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setExtElecType (
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE          et);


   /*
* Function:    TSYNC_QR_getPpsElecType()
* Description: Get the 1PPS reference's electrical type.
```

```
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstances    - The instance number
    *   OUT: et            - The electrical type result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_getPpsElecType(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstances,
        ELEC_TYPE         *et);

    /*
    * Function:    TSYNC_QR_setPpsElecType()
    * Description: Set the 1PPS reference's electrical type.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstances    - The instance number
    *        et            - The electrical type information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_setPpsElecType (
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstances,
        ELEC_TYPE          et);

/*
    * Function:    TSYNC_QR_getPpsOffset()
    * Description: Get the 1PPS reference's input offset.  Offset is in
    *              nanoseconds.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: nOffset       - The offset result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_getPpsOffset(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstance,
        int               *nOffset);

    /*
    * Function:    TSYNC_QR_setPpsOffset()
    * Description: Get the 1PPS reference's 1PPS input offset.  Offset is in
    *              nanoseconds.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        nOffset       - The offset
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QR_setPpsOffset(
```

```
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_QR_getPpsEdge()
* Description: Get the 1PPS reference's active edge setting.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: edge          - The edge result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_getPpsEdge(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    EDGE              *edge);

/*
* Function:    TSYNC_QR_setPpsEdge()
* Description: Set the 1PPS reference's active edge setting.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        edge          - The edge information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QR_setPpsEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    EDGE              edge);

/* ASCII Output Component ================================================*/

/*
* Function:    TSYNC_AP_getSigCtrl()
* Description: Get the ASCII output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_AP_setSigCtrl()
* Description: Set the ASCII output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_AP_getLocal()
* Description: Get the ASCII output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);


/*
* Function:    TSYNC_AP_setLocal()
* Description: Set the ASCII output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_AP_getTimeScale()
* Description: Get the ASCII output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
```

```
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_AP_setTimeScale()
* Description: Set the ASCII output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_AP_getFormat()
* Description: Get the ASCII output time code format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_FMT            *format);

/*
* Function:    TSYNC_AP_setFormat()
* Description: Set the ASCII output time code format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_FMT            *format);

/*
* Function:    TSYNC_AP_getMode()
* Description: Get the ASCII output mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The mode
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_OUT_MODE       *mode);

/*
* Function:    TSYNC_AP_setMode()
* Description: Set the ASCII output mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mode          - The mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AL_OUT_MODE        mode);

/*
* Function:    TSYNC_AP_getReqChar()
* Description: Get the ASCII output request character.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        req           - The request character result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getReqChar(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    char              *req);

/*
* Function:    TSYNC_AP_setReqChar()
* Description: Set the ASCII output request character.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        req           - The request character information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setReqChar(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    char               req);

/*
* Function:    TSYNC_AP_getUartCfg()
* Description: Get the ASCII output UART configuration.
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pCfg          - The UART configuration
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getUartCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    UD_CFG            *pCfg);

/*
* Function:    TSYNC_AP_setUartCfg()
* Description: Set the ASCII output UART configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pCfg          - The UART configuration
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_setUartCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    UD_CFG            *pCfg);

/*
* Function:    TSYNC_AP_getNumInst()
* Description: Get number of ASCII output instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_AP_getSource()
* Description: Get the ASCII output's Message Source.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: source        - The Message Source result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_AP_getSource(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    AO_SOURCE         *source);

/* IRIG Output Component ================================================*/
```

```
/*
* Function:    TSYNC_IP_getSigCtrl()
* Description: Get the IRIG output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_IP_setSigCtrl()
* Description: Set the IRIG output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_IP_getOffset()
* Description: Get the IRIG output's offset.  Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_IP_setOffset()
* Description: Set the IRIG output's offset.  Offset is in nanoseconds from
*              -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_IP_getLocal()
* Description: Get the IRIG output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_IP_setLocal()
* Description: Set the IRIG output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_IP_getFormat()
* Description: Get the IRIG output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FMT            *format);

/*
* Function:    TSYNC_IP_setFormat()
* Description: Set the IRIG output format.
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FMT            format);

/*
* Function:    TSYNC_IP_getAmplitude()
* Description: Get the IRIG output amplitude.
*              Amplitude is in range of 3 - 255.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: amp           - The amplitude result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getAmplitude(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *amp);

/*
* Function:    TSYNC_IP_setAmplitude()
* Description: Set the IRIG output amplitude.
*              Amplitude is in range of 3 - 255.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        amp           - The amplitude information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setAmplitude(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      amp);

/*
* Function:    TSYNC_IP_getMod()
* Description: Get the IRIG outputmodulation.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mod           - The modulation result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getMod(
    TSYNC_BoardHandle  hnd,
```

```
    unsigned int       nInstance,
    IL_MOD            *mod);

/*
* Function:    TSYNC_IP_setMod()
* Description: Set the IRIG outputmodulation.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        mod           - The modulation
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setMod(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_MOD            mod);

/*
* Function:    TSYNC_IP_getFreq()
* Description: Get the IRIG outputfrequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FRQ            *freq);

/*
* Function:    TSYNC_IP_setFreq()
* Description: Set the IRIG outputfrequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        freq          - The frequency information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setFreq(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FRQ            freq);

/*
* Function:    TSYNC_IP_getCodedExpr()
* Description: Get the IRIG output Coded Expression.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: ce            - The Coded Expression result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getCodedExpr(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CE             *ce);

/*
* Function:    TSYNC_IP_setCodedExpr()
* Description: Set the IRIG output Coded Expression.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        ce            - The Coded Expression information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setCodedExpr(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_CE             ce);

/*
* Function:    TSYNC_IP_getCtrlField()
* Description: Get the IRIG output Control Field.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: cf            - The Control Field result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getCtrlField(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CF             *cf);

/*
* Function:    TSYNC_IP_setCtrlField()
* Description: Set the IRIG output Control Field.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        cf            - The Control Field information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setCtrlField(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_CF             cf);

/*
* Function:    TSYNC_IP_getMessage()
* Description: Get the latest IRIG output message.
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the message information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getMessage(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);

/*
* Function:    TSYNC_IP_setMessage()
* Description: Set the IRIG output message.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the message information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setMessage(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);

/*
* Function:    TSYNC_IP_getCfData()
* Description: Get the latest IRIG output control field data.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the control field information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getCfData(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_IRIGCfDataObj *pObj);

/*
* Function:    TSYNC_IP_setCfData()
* Description: Set the IRIG output control field data manually.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the control field information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setCfData(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_IRIGCfDataObj *pObj);
```

```
/*
* Function:    TSYNC_IP_getNumInst()
* Description: Get number of IRIG output instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_IP_getPhase()
* Description: Get the IRIG output's current phase adjustment.  This
*              feature is only available to Spectracom for testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: phase         - The phase adjustment result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getPhase(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *phase);

/*
* Function:    TSYNC_IP_setPhase()
* Description: Set the IRIG output's phase adjustment.  This feature is
*              only available to Spectracom for testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setPhase(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      phase);

/*
* Function:    TSYNC_IP_getPhaseErr()
* Description: Get the IRIG output's current phase error.  This feature is
*              only available to Spectracom for testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: phErr         - The phase error result
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getPhaseErr(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *phErr);

/*
* Function:    TSYNC_IP_getTimeScale()
* Description: Get the IRIG output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_IP_setTimeScale()
* Description: Set the IRIG output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_IP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/* PTP Reference Component ============================================*/

/*
* Function:    TSYNC_PTR_getModuleInfo()
* Description: Gets the PTP module's version and build date.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The module's version and build date
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getModuleInfo(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_PTPModuleInfoObj *pObj);

/*
* Function:    TSYNC_PTR_getEthernetItf()
* Description: Gets Ethernet settings for the PTP module.
```

```
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The module's Ethernet interface settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getEthernetItf(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPEthernetItfObj *pObj);

    /*
    * Function:    TSYNC_PTR_setEthernetItf()
    * Description: Sets Ethernet settings for the PTP module.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - The module's new Ethernet interface settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setEthernetItf(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPEthernetItfObj *pObj);


// ----------------------------YS ----------------------------------------------------
-----
/*
    * Function:    TSYNC_PTR_getSyncEthItf()
    * Description: Gets Ethernet settings for the PTP module.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The module's Sync Ethernet interface settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getSyncEthItf(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPSyncEStatusObj *pObj);

    /*
    * Function:    TSYNC_PTR_setSyncEthItf()
    * Description: Sets sync Ethernet settings for the PTP module.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - The module's new Sync Ethernet interface settings
```

```
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setSyncEthItf(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPSyncEStatusObj *pObj);


//-------------------------------------------------------------------------------------
-----
    /*
    * Function:    TSYNC_PTR_getClockSettings()
    * Description: Gets the module's clock settings.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The module's new clock settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getClockSettings(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPClkSettingsObj *pObj);

    /*
    * Function:    TSYNC_PTR_setClockSettings()
    * Description: Sets the module's clock settings.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - The module's clock settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setClockSettings(
        TSYNC_BoardHandle        hnd,
        unsigned int             nInstance,
        TSYNC_PTPClkSettingsObj *pObj);


    /*
    * Function:    TSYNC_PTR_getUnitSettings()
    * Description: Gets general PTP settings for the module.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The module's general PTP settings
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
```

```
TSYNC_ERROR TSYNC_PTR_getUnitSettings(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_PTPUnitSettingsObj *pObj);

/*
* Function:    TSYNC_PTR_setUnitSettings()
* Description: Sets general PTP settings for the module.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - The module's new general PTP settings
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setUnitSettings(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_PTPUnitSettingsObj *pObj);

/*
* Function:    TSYNC_PTR_getPortState()
* Description: Gets the state of a port on the module.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The state of a port on the module
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getPortState(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPPortStateObj *pObj);

/*
* Function:    TSYNC_PTR_setPortState()
* Description: Gets the state of a port on the module.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The state of a port on the module
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setPortState(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPPortStateObj *pObj);


/*
* Function:    TSYNC_PTR_getPortSettings()
* Description: Gets various settings of a port on the module.
```

```
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - Various settings of a port on the module
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getPortSettings(
        TSYNC_BoardHandle         hnd,
        unsigned int              nInstance,
        TSYNC_PTPPortSettingsObj *pObj);

    /*
    * Function:    TSYNC_PTR_setPortSettings()
    * Description: Sets various settings of a port on the module.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Various settings of a port on the module
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setPortSettings(
        TSYNC_BoardHandle         hnd,
        unsigned int              nInstance,
        TSYNC_PTPPortSettingsObj *pObj);


//-------------------------------------YS-------------------------------------
/*
    * Function:    TSYNC_PTR_setVLAN()
    * Description: Sets the VLAN interface settings.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Various settings of a port on the module
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setVLAN(
        TSYNC_BoardHandle         hnd,
        unsigned int              nInstance,
        TSYNC_PTPVLANObj *pObj);
       /*
    * Function:    TSYNC_PTR_getVLAN()
    * Description: Gets the VLAN interface settings.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The Master clock's properties.
    *
    * Returns: (TSYNC_SUCCESS) Success
```

```
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getVLAN(
        TSYNC_BoardHandle       hnd,
        unsigned int            nInstance,
        TSYNC_PTPVLANObj *pObj);

      /*
    * Function:    TSYNC_PTR_setUnctMasterAdd()
    * Description: Sets the unicast parameters provided by the clock and adds a master
cmocl to the list of master clocks.
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        pObj          - Various settings of a port on the module
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_setUnctMasterAdd(
        TSYNC_BoardHandle         hnd,
        unsigned int              nInstance,
        TSYNC_PTPUnctMasterAddObj *pObj);
   /*
    * Function:    TSYNC_PTR_getUnctMasterAdd()
    * Description: Gets the Master clock's propertirs  .
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The Master clock's properties.
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getUnctMasterAdd(
        TSYNC_BoardHandle       hnd,
        unsigned int            nInstance,
        TSYNC_PTPUnctMasterAddObj *pObj);

      /*
    * Function:    TSYNC_PTR_getUnctSlaveProp()
    * Description: Gets the unicast mode properties of a Slave-Only unit  .
    *
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: pObj          - The unicast mode properties of a Slave-Only unit.
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_PTR_getUnctSlaveProp(
        TSYNC_BoardHandle       hnd,
        unsigned int            nInstance,
         TSYNC_PTPUnctSlavePropObj *pObj);

      /*
    * Function:    TSYNC_PTR_getUnctMasterProp()
```

```
* Description: Gets the unicast mode properties of a Master-Only unit  .
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The unicast mode properties of a Master-Only unit.
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getUnctMasterProp(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
     TSYNC_PTPUnctMasterPropObj *pObj);


/*
* Function:    TSYNC_PTR_getClkQuality()
* Description: Gets the module's reported clock quality information.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The module's reported clock quality information.
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getClkQuality(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_PTPClkQualityObj *pObj);

/*
* Function:    TSYNC_PTR_getTimeProperties()
* Description: Gets the module's reported time properties.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The module's reported time properties
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getTimeProperties(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_PTPTimePropObj *pObj);

/*
* Function:    TSYNC_PTR_getParentProperties()
* Description: Gets the module's parent properties dataset.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The module's parent properties dataset
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getParentProperties(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_PTPParentPropObj *pObj);

/*
* Function:    TSYNC_PTR_getGrandmasterProperties()
* Description: Gets the module's grandmaster properties dataset.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - The module's grandmaster properties dataset
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getGrandmasterProperties(
    TSYNC_BoardHandle            hnd,
    unsigned int                 nInstance,
    TSYNC_PTPGrandmasterPropObj *pObj);

/*
* Function:    TSYNC_PTR_getTODEnabled()
* Description: Gets whether the module outputs a TOD.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bTodEnabled   - Whether the module outputs a TOD
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getTODSettings(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPTODSettingsObj *pObj);

/*
* Function:    TSYNC_PTR_setTODEnabled()
* Description: Sets whether the module outputs a TOD.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bTodEnabled   - Whether the module outputs a TOD
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setTODSettings(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPTODSettingsObj *pObj);

/*
```

```
* Function:    TSYNC_PTR_getPPSEnabled()
* Description: Gets whether the module outputs a PPS when it's a slave.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bPpsEnabled   - Whether the module outputs a PPS when it's a slave
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getPPSEnabled(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int              *bPpsEnabled);

/*
* Function:    TSYNC_PTR_setPPSEnabled()
* Description: Sets whether the module outputs a PPS when it's a slave.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bPpsEnabled   - Whether the module outputs a PPS when it's a slave
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setPPSEnabled(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               bPpsEnabled);

/*
* Function:    TSYNC_PTR_getPPSRisingEdge()
* Description: Gets if the module's output PPS is rising-edge or not.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*   OUT: bPpsRisingEdge - Whether the module's PPS is risin-edge or not
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getPPSRisingEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int              *bPpsRisingEdge);

/*
* Function:    TSYNC_PTR_setPPSRisingEdge()
* Description: Sets if the module's output PPS is rising-edge or not.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        bPpsRisingEdge - Whether the module's PPS is risin-edge or not
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setPPSRisingEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               bPpsRisingEdge);

/*
* Function:    TSYNC_PTR_saveSettingsToROM()
* Description: Saves any settings that have been changed in the PTP module
*              to the module's ROM.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_saveSettingsToROM(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance);

/*
* Function:    TSYNC_PTR_resetModule()
* Description: Resets the PTP module, based on the type of reset requested.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - PTP instance requested
*        resetType     - The type of reset requested.
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_resetModule(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    PTL_RESET         resetType);

/*
* Function:    TSYNC_PTR_getNumInst()
* Description: Gets the number of PTP modules present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of PTP modules present in the system
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getNumInst(
    TSYNC_BoardHandle hnd,
    unsigned int     *nInstances);

/*
* Function:    TSYNC_PTR_reinitModule()
* Description: Reinitializes PTP Module.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
 TSYNC_ERROR TSYNC_PTR_reinitModule(
     TSYNC_BoardHandle hnd,
     unsigned int      nInstance);

/*
 * Function:    TSYNC_PTR_getValidity()
 * Description: Get the PTR validity structure.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *   OUT: bTimeValid    - The time reference result
 *        bPpsValid     - The pps reference result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_PTR_getValidity(
     TSYNC_BoardHandle  hnd,
     unsigned int       nInstance,
     int               *bTimeValid,
     int               *bPpsValid);

/*
 * Function:    TSYNC_PTR_getMode()
 * Description: Gets the module's current operational mode.
 *
 * Parameters:
 *   IN:  hnd            - Board handle
 *        nInstance      - The instance number
 *   OUT: bMode          - Current Operational Mode
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_PTR_getMode(
     TSYNC_BoardHandle hnd,
     unsigned int      nInstance,
     int              *bMode);

 /*
 * Function:    TSYNC_PTR_setMode()
 * Description: Sets the module's current operational mode.
 *
 * Parameters:
 *   IN:  hnd            - Board handle
 *        nInstance      - The instance number
 *        bMode          - New Operational Mode
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
 DLL_EXPORT
 TSYNC_ERROR TSYNC_PTR_setMode(
     TSYNC_BoardHandle hnd,
     unsigned int      nInstance,
     int               bMode);


 /*
 * Function:    TSYNC_PTR_getMacAddr()
 * Description: Sets the module's current MAC Address.
 *
 * Parameters:
```

```
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - MAC Address object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getMacAddr(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_PTPMacAddrObj  *pObj);


/*
* Function:    TSYNC_PTR_setMacAddr()
* Description: Sets the module's current MAC Address.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - MAC Address object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setMacAddr(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPMacAddrPwObj   *pObj);

/*
* Function:    TSYNC_PTR_getMasterActive()
* Description: Gets the Master Active state.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - MAC Address object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getMasterActive(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int              *bMasterActive);

/*
* Function:    TSYNC_PTR_getModuleStatus()
* Description: Gets the Module Status
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        bModuleStatus  - Module Status (Reset Cause)
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getModuleStatus(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int              *bModuleStatus);
```

```
/*
* Function:    TSYNC_PTR_getUserDesc()
* Description: Gets the User Description strings
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - User Description object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getUserDesc(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPUserDescObj    *pObj);

/*
* Function:    TSYNC_PTR_setUserDesc()
* Description: Sets the User Description strings
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - User Description object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setUserDesc(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPUserDescObj    *pObj);

/*
* Function:    TSYNC_PTR_getRefId()
* Description: Get reference identifier for a PTP reference instance.
*
* Parameters:
*   IN:  hnd           - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getRefId(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);

/*
* Function:    TSYNC_PTR_getDebugOutput()
* Description: Gets Debug Output information.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - Debug Output object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
TSYNC_ERROR TSYNC_PTR_getDebugOutput(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPDebugOutputObj *pObj);

/*
* Function:    TSYNC_PTR_setDebugOutput()
* Description: Sets Debug Output information.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - Debug Output object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setDebugOutput(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPDebugOutputObj *pObj);

/*
* Function:    TSYNC_PTR_getEthernetStatus()
* Description: Gets current Ethernet Status.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - Ethernet Status object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getEthernetStatus(
    TSYNC_BoardHandle           hnd,
    unsigned int                nInstance,
    TSYNC_PTPEthernetStatusObj *pObj);

/*
* Function:    TSYNC_PTR_getSyncEItf()
* Description: Gets Sync-E Interface information.
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - Sync-E Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getSyncEItf(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPSyncEStatusObj *pObj);

/*
* Function:    TSYNC_PTR_setSyncEItf()
* Description: Sets Sync-E Interface information.
*
* Parameters:
```

```
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - Sync-E Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setSyncEItf(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPSyncEStatusObj *pObj);

/*
* Function:    TSYNC_PTR_getPTPItf()
* Description: Gets PTP Protocol information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - PTP Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getPTPItf(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_PTPItfObj    *pObj);

/*
* Function:    TSYNC_PTR_setPTPItf()
* Description: Sets PTP Protocol information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - PTP Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setPTPItf(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_PTPItfObj    *pObj);

/*
* Function:    TSYNC_PTR_getFTPItf()
* Description: Gets FTP Protocol information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - FTP Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getFTPItf(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
```

```
    TSYNC_PTPFtpItfObj    *pObj);

/*
* Function:    TSYNC_PTR_setFTPItf()
* Description: Sets FTP Protocol information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - FTP Interface object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setFTPItf(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_PTPFtpItfObj    *pObj);

/*
* Function:    TSYNC_PTR_getPTPStatistics()
* Description: Gets PTP Statistics Control information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*
*  OUT:  pObj           - PTP Statistics object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getPTPStatistics(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPStatisticsObj  *pObj);

/*
* Function:    TSYNC_PTR_setPTPStatistics()
* Description: Sets PTP Statistics Control information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - PTP Statistics object
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setPTPStatistics(
    TSYNC_BoardHandle        hnd,
    unsigned int             nInstance,
    TSYNC_PTPStatisticsObj  *pObj);

/*
* Function:    TSYNC_PTR_getClockProperties()
* Description: Gets PTP Clock Properties information
*
* Parameters:
*   IN:  hnd            - Board handle
*        nInstance      - The instance number
*        pObj           - PTP Clock Properties object
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getClockProperties(
    TSYNC_BoardHandle            hnd,
    unsigned int                 nInstance,
    TSYNC_PTPClockPropertiesObj *pObj);

/*
* Function:    TSYNC_PTR_getState()
* Description: Retreive the PTR update status.
*
* Parameters:
*   IN:  *hw           - Handle
*        *obj          - Pointer to the state result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getState(
    TSYNC_BoardHandle  hnd,
    TSYNC_StateObj    *obj);

/*
* Function:    TSYNC_PTR_start()
* Description: Begin a PTR update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*        *obj1         - Pointer to the update image information
*        *obj2         - Pointer to the update header information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_start(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageIdObj     *obj1,
    TSYNC_FSImageHeaderObj *obj2);

/*
* Function:    TSYNC_PTR_data()
* Description: Send a single data block in the update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*   OUT: *obj          - Pointer to the update data information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_data(
    TSYNC_BoardHandle    hnd,
    TSYNC_UpdateDataObj *obj);

/*
* Function:    TSYNC_PTR_end()
* Description: Finish the update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*   OUT: *obj          - Pointer to the update end information
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_end(
    TSYNC_BoardHandle   hnd,
    TSYNC_UpdateEndObj *obj);

/*
* Function:    TSYNC_PTR_cancel()
* Description: Cancel the update sequence.
*
* Parameters:
*   IN:  *hw           - Handle
*   OUT: imageType     - the type of update sequence being cancelled
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_cancel(
    TSYNC_BoardHandle hnd,
    FS_IMG            imageType);

/*
* Function:    TSYNC_PTR_getBootImg()
* Description: Gets the image that the module will boot into on a Module Reset.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bBootImg      - Whether the module outputs a PPS when it's a slave
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_getBootImg(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int              *bBootImg);

/*
* Function:    TSYNC_PTR_setBootImg()
* Description: Sets the image that the module will boot into on a Module Reset.
*
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bBootImg      - Whether the module outputs a PPS when it's a slave
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PTR_setBootImg(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               bBootImg);


/* STANAG/HaveQuick Output Component ====================================*/

/*
* Function:    TSYNC_QP_getSigCtrl()
```

```
    * Description: Get the HaveQuick output's signature control state.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: sig           - The signature control result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_getSigCtrl(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstance,
        SIG_CTL           *sig);

    /*
    * Function:    TSYNC_QP_setSigCtrl()
    * Description: Set the HaveQuick output's signature control state.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        sig           - The signature control information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_setSigCtrl(
        TSYNC_BoardHandle hnd,
        unsigned int      nInstance,
        SIG_CTL           sig);

/*
    * Function:    TSYNC_QP_getExtSigCtrl()
    * Description: Get the HaveQuick extended output's signature control state.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: sig           - The signature control result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_getExtSigCtrl(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstance,
        SIG_CTL           *sig);

    /*
    * Function:    TSYNC_QP_setExtSigCtrl()
    * Description: Set the HaveQuick extended output's signature control state.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        sig           - The signature control information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_setExtSigCtrl(
        TSYNC_BoardHandle hnd,
```

```
        unsigned int      nInstance,
        SIG_CTL           sig);

    /*
    * Function:    TSYNC_QP_getOffset()
    * Description: Get the HaveQuick output's offset.  Offset is in nanoseconds.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: nOffset       - The offset result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_getOffset(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstance,
        int               *nOffset);

    /*
    * Function:    TSYNC_QP_setOffset()
    * Description: Set the HaveQuick output's offset.  Offset is in nanoseconds
    *              from -500 msec to +500 msec.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *        nOffset       - The offset information
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_setOffset(
        TSYNC_BoardHandle hnd,
        unsigned int      nInstance,
        int               nOffset);
/*
    * Function:    TSYNC_QP_getExtOffset()
    * Description: Get the HaveQuick Extended output's offset.  Offset is in
nanoseconds.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
    *        nInstance     - The instance number
    *   OUT: nOffset       - The offset result
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_QP_getExtOffset(
        TSYNC_BoardHandle  hnd,
        unsigned int       nInstance,
        int               *nOffset);

    /*
    * Function:    TSYNC_QP_setExtOffset()
    * Description: Set the HaveQuick Extended output's offset.  Offset is in
nanoseconds
    *              from -500 msec to +500 msec.
    *
    * Parameters:
    *   IN:  hnd           - Board handle
```

```
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setExtOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_QP_getLocal()
* Description: Get the HaveQuick output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_QP_setLocal()
* Description: Set the HaveQuick output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_QP_getBs()
* Description: Get the HaveQuick output Bit Synchronization.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: bs            - The Bit Synchronization result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getBs(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int       *bs);
```

```
/*
* Function:    TSYNC_QP_setBs()
* Description: Set the HaveQuick output Bit Synchronization.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        bs            - The Bit Synchronization information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setBs(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      bs);

/*
* Function:    TSYNC_QP_getFormat()
* Description: Get the HaveQuick output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    QL_FMT            *format);

/*
* Function:    TSYNC_QP_setFormat()
* Description: Set the HaveQuick output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number

*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    QL_FMT            format);

/*
* Function:    TSYNC_QP_getExtFormat()
* Description: Get the Extended HaveQuick output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getExtFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    QL_FMT            *format);

/*
* Function:    TSYNC_QP_setExtFormat()
* Description: Set the HaveQuick Extended output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setExtFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    QL_FMT            format);

/*
* Function:    TSYNC_QP_getTimeScale()
* Description: Get the HaveQuick output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_QP_setTimeScale()
* Description: Set the HaveQuick output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
 * Function:    TSYNC_QP_getExtTimeScale()
 * Description: Get the HaveQuick extended output's time scale.
 *
 * Parameters:
```

```
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getExtTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_QP_setExtTimeScale()
* Description: Set the HaveQuick extended output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setExtTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_QP_getNumInst()
* Description: Get number of HaveQuick output instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
/*
* Function:    TSYNC_QP_getTfd()
* Description: Get the HaveQuick output's time fault discrete.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The number of instances result
*   OUT  tfd           - Pointer to the time fault discrete result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getTfd(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    unsigned int      *tfd);
/*
* Function:    TSYNC_QP_setTfd()
* Description: Set the HaveQuick output time fault discrete.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        tfd            - The time fault discrete information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setTfd(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      tfd);

/*
* Function:    TSYNC_QP_getTfdState()
* Description: Get the HaveQuick output's time fault discrete state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The number of instances result
*   OUT  tfd           - Pointer to the time fault discrete result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getTfdState(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    unsigned int      *tfd);
/*
* Function:    TSYNC_QP_setTfdState()
* Description: Set the HaveQuick output time fault discrete state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        tfd            - The time fault discrete information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setTfdState(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      tfd);

/*
* Function:    TSYNC_QP_getTfom()
* Description: Get the HaveQuick output's TFOM.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The number of instances result
*        tfd           - Pointer to the TFOM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getTfom(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    unsigned int      *tfom);
```

```
/*
 * Function:    TSYNC_QP_getRequiredTfom()
 * Description: Get the HaveQuick output's required TFOM.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstances    - The number of instances result
 *        tfd           - Pointer to the TFOM result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getRequiredTfom(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    unsigned int      *tfom);

/*
 * Function:    TSYNC_QP_setRequiredTfom()
 * Description: Set the HaveQuick output required tfom.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstance     - The instance number
 *        tfom          - The required tfom information
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setRequiredTfom(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      tfom);

/*
 * Function:    TSYNC_QP_getExtTfom()
 * Description: Get the HaveQuick extended output's TFOM.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstances    - The number of instances result
 *        tfd           - Pointer to the TFOM result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getExtTfom(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    unsigned int      *tfom);

/*
 * Function:    TSYNC_QP_getLevel()
 * Description: Get the output's level.
 *
 * Parameters:
 *   IN:  hnd           - Board handle
 *        nInstances    - The instance number
 *   OUT: sl            - The signal level result
 *
 * Returns: (TSYNC_SUCCESS) Success
 */
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getLevel(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    SIG_LEVEL         *sl);

/*
* Function:    TSYNC_QP_setLevel()
* Description: Set the output's level.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        pw            - The signal level information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setLevel(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    SIG_LEVEL          sl);

/*
* Function:    TSYNC_QP_getElecType()
* Description: Get the output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*   OUT: et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getElecType(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE         *et);

/*
* Function:    TSYNC_QP_setElecType()
* Description: Set the output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setElecType (
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE          et);

/*
* Function:    TSYNC_QP_getExtElecType()
* Description: Get the extended output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        nInstances    - The instance number

*   OUT: et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getExtElecType(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE         *et);

/*
* Function:    TSYNC_QP_setExtElecType()
* Description: Set the extended output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setExtElecType (
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE          et);
   /*
* Function:    TSYNC_QP_getPpsSigCtrl()
* Description: Get the PPS output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getPpsSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_QP_setPpsSigCtrl()
* Description: Set the PPS output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setPpsSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
```

```
* Function:    TSYNC_QP_getPpsElecType()
* Description: Get the PPS output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*   OUT: et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getPpsElecType(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE         *et);

/*
* Function:    TSYNC_QP_setPpsElecType()
* Description: Set the PPS output's electrical type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstances    - The instance number
*        et            - The electrical type result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setPpsElecType (
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstances,
    ELEC_TYPE          et);

/*
* Function:    TSYNC_QP_getPpsOffset()
* Description: Get the PPS output's offset.  Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getPpsOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_QP_setPpsOffset()
* Description: Get the PPS output's offset.  Offset is in nanoseconds from
*              -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
TSYNC_ERROR TSYNC_QP_setPpsOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_QP_getPpsEdge()
* Description: Get the PPS output's edge.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: edge          - The edge result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getPpsEdge(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    EDGE              *edge);

/*
* Function:    TSYNC_QP_setPpsEdge()
* Description: Set the PPS output's edge.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        edge          - The edge information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setPpsEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    EDGE              edge);

/*
* Function:    TSYNC_QP_getPpsPw()
* Description: Get the PPS output's pulse width.  Pulse width is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pw            - The pulse width result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_getPpsPw(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *pw);

/*
* Function:    TSYNC_QP_setPpsPw()
* Description: Set the PPS output's pulse width.  Pulse width is in
*              nanoseconds from 10 nsec to 999,999,990 nsec.
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pw            - The pulse width information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_QP_setPpsPw(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      pw);


/* SMPTE/EBU Output Component ====================================*/

/*
* Function:    TSYNC_EP_getSigCtrl()
* Description: Get the SMPTE/EBU output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_EP_setSigCtrl()
* Description: Set the SMPTE/EBU output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_EP_getOffset()
* Description: Get the SMPTE/EBU output's offset.  Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getOffset(
    TSYNC_BoardHandle  hnd,
```

```
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_EP_setOffset()
* Description: Set the SMPTE/EBU output's offset.  Offset is in nanoseconds
*              from -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_EP_getLocal()
* Description: Get the SMPTE/EBU output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_EP_setLocal()
* Description: Set the SMPTE/EBU output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_EP_getFormat()
* Description: Get the SMPTE/EBU output format.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    ESL_FMT           *format);

/*
* Function:    TSYNC_EP_setFormat()
* Description: Set the SMPTE/EBU output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    ESL_FMT           format);

/*
* Function:    TSYNC_EP_getTimeScale()
* Description: Get the SMPTE/EBU output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_EP_setTimeScale()
* Description: Set the SMPTE/EBU output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
```

```
* Function:    TSYNC_EP_getAmplitude()
* Description: Get the SMPTE/EBU output's amplitude.
*              Amplitude is in range of 0 - 255.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: amp           - The amplitude result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getAmplitude(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *amp);

/*
* Function:    TSYNC_EP_setAmplitude()
* Description: Set the SMPTE/EBU output's amplitude.
*              Amplitude is in range of 0 - 255.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        amp           - The amplitude information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_setAmplitude(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      amp);

/*
* Function:    TSYNC_EP_getNumInst()
* Description: Get number of SMPTE/EBU output instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_EP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/* Display Output Component ================================================*/

/*
* Function:    TSYNC_DP_getLocal()
* Description: Get the display output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the Local Clock result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_DP_setLocal()
* Description: Set the display output's local time zone and DST rule.
*              Timezone and DST offsets are in seconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the Local Clock information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);

/*
* Function:    TSYNC_DP_getFormat()
* Description: Get the display output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: format        - The format result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    ML_HOUR           *format);

/*
* Function:    TSYNC_DP_setFormat()
* Description: Set the display output format.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    ML_HOUR           format);

/*
* Function:    TSYNC_DP_getTimeScale()
* Description: Get the display output's time scale.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pObj          - Pointer to the time scale result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_DP_setTimeScale()
* Description: Set the display output's time scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pObj          - Pointer to the time scale information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);

/*
* Function:    TSYNC_DP_getNumInst()
* Description: Get number of display output instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_DP_getMode()
* Description: Get the display mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: mode          - The mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    DP_MODE           *mode);
```

```
/*
* Function:    TSYNC_DP_setMode()
* Description: Set the display mode.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        format        - The format information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_DP_setMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    DP_MODE           mode);

/* 1PPS Output Component ==============================================*/

/*
* Function:    TSYNC_PP_getSigCtrl()
* Description: Get the PPS output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_PP_setSigCtrl()
* Description: Set the PPS output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_PP_getFreq()
* Description: Get the PPS output's frequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    float             *freq);

/*
* Function:    TSYNC_PP_getOffset()
* Description: Get the PPS output's offset.  Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: nOffset       - The offset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);

/*
* Function:    TSYNC_PP_setOffset()
* Description: Get the PPS output's offset.  Offset is in nanoseconds from
*              -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        nOffset       - The offset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);

/*
* Function:    TSYNC_PP_getEdge()
* Description: Get the PPS output's edge.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: edge          - The edge result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getEdge(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    EDGE              *edge);

/*
* Function:    TSYNC_PP_setEdge()
* Description: Set the PPS output's edge.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        edge          - The edge information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_setEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    EDGE              edge);

/*
* Function:    TSYNC_PP_getPulseWidth()
* Description: Get the PPS output's pulse width.  Pulse width is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pw            - The pulse width result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getPulseWidth(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *pw);

/*
* Function:    TSYNC_PP_setPulseWidth()
* Description: Set the PPS output's pulse width.  Pulse width is in
*              nanoseconds from 10 nsec to 999,999,990 nsec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        pw            - The pulse width information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_setPulseWidth(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      pw);

/*
* Function:    TSYNC_PP_getNumInst()
* Description: Get number of PPS output instances present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_PP_getNumInst(
    TSYNC_BoardHandle  hnd,
```

```
    unsigned int      *nInstances);

/* Variable-Frequency Output Component ===================================*/

/*
* Function:    TSYNC_VP_getSigCtrl()
* Description: Get the Variable-Frequency output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_VP_setSigCtrl()
* Description: Set the Variable-Frequency output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_VP_getFreq()
* Description: Get the Variable-Frequency output frequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    float             *freq);

/*
* Function:    TSYNC_VP_setFreq()
* Description: Set the Variable-Frequency output frequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        freq          - The frequency information
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_setFreq(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    float             freq);

/*
* Function:    TSYNC_VP_getPhase()
* Description: Get the Variable-Phase output frequency.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: phase         - The phase result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getPhase(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    float            *phase);

/*
* Function:    TSYNC_VP_setPhase()
* Description: Set the Variable-Frequency output phase.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        phase         - The phase information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_setPhase(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    float             phase);

/*
* Function:    TSYNC_VP_getCfg()
* Description: Get the Variable-Frequency output configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: cfg           - The configuration result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getCfg(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_VPCfgObj   *pObj);

/*
* Function:    TSYNC_VP_getNumInst()
* Description: Get number of Variable-Frequency output instances present in
```

```
*              the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_VP_getLock()
* Description: Get number of Variable-Frequency output instances present in
*              the system.
*
* Parameters:
*   IN:  hnd           - Board handle
   *        nInstance     - The instance number
*   OUT: lock          - PLL lock enabled/disabled
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_VP_getLock(
    TSYNC_BoardHandle  hnd,
         unsigned int       nInstance,
    int                     *lock);

/* E1/T1 Output Component ================================================*/

/*
* Function:    TSYNC_ETP_getSigCtrl()
* Description: Get the E1/T1 output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_ETP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_ETP_setSigCtrl()
* Description: Set the E1/T1 output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_ETP_setSigCtrl(
```

```
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_ETP_getCfg()
* Description: Get the E1/T1 output configuration parameters.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pCfg          - Pointer to the configuration parameter structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_ETP_getCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    ETO_CFG           *pCfg);

/*
* Function:    TSYNC_ETP_setMode()
* Description: Set the E1/T1 output configuration parameters.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: pCfg          - Pointer to the configuration parameter structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_ETP_setCfg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    ETO_CFG           *pCfg);

/*
* Function:    TSYNC_ETP_getNumInst()
* Description: Get number of E1/T1 output set instances in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_ETP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/* Oscillator Component ================================================*/

/*
* Function:    TSYNC_XO_getDiscState()
* Description: Get the external oscillator's disciplining state.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: disc          - The disciplining state result
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getDiscState(
    TSYNC_BoardHandle  hnd,
    int               *disc);

/*
* Function:    TSYNC_XO_getMode()
* Description: Get the external oscillator's mode used when
*              disciplining or testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: mode          - The mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getMode(
    TSYNC_BoardHandle  hnd,
    XO_MODE           *mode);

/*
* Function:    TSYNC_XO_setMode()
* Description: Set the external oscillator's mode used when
*              disciplining or testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*        mode          - The mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_setMode(
    TSYNC_BoardHandle hnd,
    XO_MODE           mode);

/*
* Function:    TSYNC_XO_getDac()
* Description: Get the external oscillator's DAC setting for testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: dac           - The DAC result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getDac(
    TSYNC_BoardHandle  hnd,
    unsigned short    *dac);

/*
* Function:    TSYNC_XO_setDac()
* Description: Set the external oscillator's DACsetting for testing.
*
* Parameters:
*   IN:  hnd           - Board handle
*        dac           - The DAC information
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_setDac(
    TSYNC_BoardHandle hnd,
    unsigned short    dac);

/*
* Function:    TSYNC_XO_getAlarm()
* Description: Get the external oscillator's alarm state.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: alarm         - The alarm result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getAlarm(
    TSYNC_BoardHandle  hnd,
    unsigned int      *alarm);

/*
* Function:    TSYNC_XO_getSerNum()
* Description: Get the external oscillator's serial number.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: sernum        - The serial number result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getSerNum(
    TSYNC_BoardHandle  hnd,
    TSYNC_SerialNoObj *sernum);

/*
* Function:    TSYNC_XO_getMfrMdl()
* Description: Get the external oscillator's manufacturer and model.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the man/mod result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getMfrMdl(
    TSYNC_BoardHandle  hnd,
    TSYNC_ManModObj   *pObj);

/*
* Function:    TSYNC_XO_setMessage()
* Description: Send a Custom Message to the external oscillator.  Result is
*              passed back in same buffer.  Valid only when in test mode.
*
* Parameters:
*   IN:    hnd         - Board handle
*   INOUT: pObj        - Pointer to the message information and result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
```

```
TSYNC_ERROR TSYNC_XO_setMessage(
    TSYNC_BoardHandle       hnd,
    TSYNC_OscCustomMessageObj *pObj);

/*
* Function:    TSYNC_XO_getCmd()
* Description: Get a disciplining dataset from the external oscillator.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - Pointer to the dataset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getCmd(
    TSYNC_BoardHandle  hnd,
    TSYNC_OscDiscObj  *pObj);

/*
* Function:    TSYNC_XO_setCmd()
* Description: Send a disciplining command and dataset to the
*              external oscillator.
*
* Parameters:
*   IN:  hnd           - Board handle
*        pObj          - Pointer to the command information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_setCmd(
    TSYNC_BoardHandle  hnd,
    TSYNC_OscDiscObj  *pObj);

/*
* Function:    TSYNC_XO_getPhaseErr()
* Description: Get the estimated phase error of the external oscillator.
*
* Parameters:
*   IN:  hnd           - Board handle
*        error         - Phase error result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getPhaseErr(
    TSYNC_BoardHandle  hnd,
    int               *error);

/*
* Function:    TSYNC_XO_getFreqErr()
* Description: Get the frequency phase error of the external oscillator.
*
* Parameters:
*   IN:  hnd           - Board handle
*        error         - Frequency error result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getFreqErr(
    TSYNC_BoardHandle  hnd,
```

```
    float             *error);

/*
* Function:    TSYNC_XO_getCalVal()
* Description: Get the calibration value of the external oscillator.
*
* Parameters:
*   IN:  hnd           - Board handle
*        cal           - calibration value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getCalVal(
    TSYNC_BoardHandle  hnd,
    float             *cal);

/*
* Function:    TSYNC_XO_getOscType()
* Description: Get the system oscillator type.
*
* Parameters:
*   IN:  hnd           - Board handle
*        err           - Oscillator type
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_XO_getOscType(
    TSYNC_BoardHandle hnd,
    OSC *oscType);


/* Fixed-Frequency Output Component ======================================*/

/*
* Function:    TSYNC_FP_getSigCtrl()
* Description: Get the Fixed Frequency output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_FP_setSigCtrl()
* Description: Set the Fixed Frequency output's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_FP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    SIG_CTL           sig);

/*
* Function:    TSYNC_FP_getFreq()
* Description: Get the Fixed Frequency output's frequency.  Frequency is in
*              Hertz.
*
* Parameters:
*   IN:  hnd           - Board handle
*        nInstance     - The instance number
*   OUT: freq          - The frequency result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FP_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    float             *freq);

/*
* Function:    TSYNC_FP_getNumInst()
* Description: Get number of fixed-freq output instances present in the
*              system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_FP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/* Shared-Memory Service ================================================*/

/*
* Function:    TSYNC_MS_reset()
* Description: Reset a shared memory data set item.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Index of the shared memory set
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_MS_reset(
    TSYNC_BoardHandle hnd,
    MS_DI_INDEX       index);

/*
* Function:    TSYNC_MS_getData()
* Description: Get a shared memory data set item.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        index         - Index of the shared memory set
*   OUT: pObj          - Pointer to the shared memory dataset result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_MS_getData(
    TSYNC_BoardHandle      hnd,
    MS_DI_INDEX            index,
    TSYNC_SharedMemoryObj *pObj);

/*
* Function:    TSYNC_MS_setData()
* Description: Set a shared memory data set item.
*
* Parameters:
*   IN:  hnd           - Board handle
*        index         - Index of the shared memory set
*        pObj          - Pointer to the shared memory dataset information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_MS_setData(
    TSYNC_BoardHandle      hnd,
    MS_DI_INDEX            index,
    TSYNC_SharedMemoryObj *pObj);

/* General-Purpose Output Component ====================================*/

/*
* Function:    TSYNC_GO_getSigCtrl()
* Description: Get the GPO's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: sig           - The signature control result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    SIG_CTL           *sig);

/*
* Function:    TSYNC_GO_setSigCtrl()
* Description: Set the GPO's signature control state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        sig           - The signature control information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSigCtrl(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    SIG_CTL           sig);
```

```
/*
* Function:    TSYNC_GO_getEnable()
* Description: Get the GPO's enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: bEnable       - The output enable result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getEnable(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bEnable);

/*
* Function:    TSYNC_GO_setEnable()
* Description: Set the GPO's enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        bEnable       - The output enable information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setEnable(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               bEnable);

/*
* Function:    TSYNC_GO_getValue()
* Description: Get the GPO's current output value.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: bValue        - The value result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getValue(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bValue);

/*
* Function:    TSYNC_GO_getMode()
* Description: Get the GPO's mode state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: mode          - The mode result
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getMode(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    OD_MODE           *mode);

/*
* Function:    TSYNC_GO_setMode()
* Description: Set the GPO's mode state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        mode          - The mode information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setMode(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    OD_MODE           mode);

/*
* Function:    TSYNC_GO_getDvmValue()
* Description: Get the GPO's Direct Value Mode (DVM) value state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: bValue        - The DVM result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getDvmValue(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bValue);

/*
* Function:    TSYNC_GO_setDvmValue()
* Description: Set the GPO's Direct Value Mode (DVM) value state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        bValue        - The DVM information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setDvmValue(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               bValue);

/*
* Function:    TSYNC_GO_getMatchEnable()
* Description: Get the GPO's match enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*        gpo           - GPO index
*        lvl           - Low or High Match Time
*   OUT: bEnable       - Match enable
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getMatchEnable(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    LEVEL              lvl,
    int               *bEnable);

/*
* Function:    TSYNC_GO_setMatchEnable()
* Description: Set the GPO's match enable state.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        lvl           - Low or High Match Time
*        bEnable       - Match enable
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setMatchEnable(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    LEVEL             lvl,
    int               bEnable);

/*
* Function:    TSYNC_GO_getSquareWave()
* Description: Get the GPO's square wave output configuration structure.
*              Offset, period, and pulse width are in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: pObj          - Pointer to the configuration result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSquareWave(
    TSYNC_BoardHandle   hnd,
    OD_PIN              gpo,
    TSYNC_GPOSquareObj *pObj);

/*
* Function:    TSYNC_GO_setSquareWave()
* Description: Get the GPO's square wave output configuration structure.
*              Offset and pulse width are in nanoseconds.  Period is in
*              nanoseconds or microseconds depending on scale bit (msb).
*              Offset is from -500 msec to +500 msec. Period is from 100
*              nsec to 20 sec in nanosecond scale, and from 100 usec to
*              20,000 sec in microsecond scale.  Pulse width is from 10
*              nsec to 999,999,990 nsec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
```

```
*        pObj          - Pointer to the configuration information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSquareWave(
    TSYNC_BoardHandle   hnd,
    OD_PIN              gpo,
    TSYNC_GPOSquareObj *pObj);

/*
* Function:    TSYNC_GO_getSWOffset()
* Description: Get the GPO's square wave offset.  Offset is in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        off           - Pointer to the square wave offset
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWOffset(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *off);

/*
* Function:    TSYNC_GO_setSWOffset()
* Description: Set the GPO's square wave offset.  Offset is in nanoseconds
*              from -500 msec to +500 msec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        off           - Square wave offset
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWOffset(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               off);

/*
* Function:    TSYNC_GO_getSWPeriod()
* Description: Get the GPO's square wave period.  Period is in nanoseconds
*              or microseconds depending on scale bit (msb).
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: per           - Pointer to the square wave period
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWPeriod(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    unsigned int      *per);
```

```
/*
* Function:    TSYNC_GO_setSWPeriod()
* Description: Set the GPO's square wave period.  Period is in nanoseconds
*              or microseconds depending on scale bit (msb).  Period is
*              from 100 nsec to 20 sec in nanosecond scale, and from 100
*              usec to 20,000 sec in microsecond scale.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*   OUT: per           - Square wave period
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWPeriod(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    unsigned int      per);

/*
* Function:    TSYNC_GO_getSWPW()
* Description: Get the GPO's square wave pulse width.  Pulse width is in
*              nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        pw            - Pointer to the square wave pulse width
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWPW(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    unsigned int      *pw);

/*
* Function:    TSYNC_GO_setSWPW()
* Description: Set the GPO's square wave pulse width.  Pulse width is in
*              nanoseconds from 20 nsec to 900,000,000 nsec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        pw            - Square wave pulse width
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWPW(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    unsigned int      pw);

/*
* Function:    TSYNC_GO_getSWEdge()
* Description: Get the GPO's square wave active edge.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
```

```
*        edge          - Pointer to the square wave active edge
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWEdge(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    EDGE              *edge);

/*
* Function:    TSYNC_GO_setSWEdge()
* Description: Set the GPO's square wave active edge.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        edge          - Square wave active edge
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWEdge(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    EDGE              edge);

/*
* Function:    TSYNC_GO_getSWPerCorr()
* Description: Get the GPO's square wave period correction.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        num           - Pointer to the square wave period correction
*                        numerator
*        den           - Pointer to the square wave period correction
*                        denominator
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWPerCorr(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    unsigned int      *num,
    unsigned int      *den);

/*
* Function:    TSYNC_GO_setSWPerCorr()
* Description: Set the GPO's square wave period correction.  The numerator
*              and denominator are in the range of 0 to 255, where the
*              numerator must be less than the denominator, and a
*              numerator of 0 indicating no period correction.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        num           - Square wave period correction numerator
*        den           - Square wave period correction denominator
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWPerCorr(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    unsigned int      num,
    unsigned int      den);

/*
* Function:    TSYNC_GO_getSWAlignCnt()
* Description: Get the GPO's square wave alignment count.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        cnt           - Pointer to the square wave alignment count
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWAlignCnt(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    unsigned int      *cnt);

/*
* Function:    TSYNC_GO_setSWAlignCnt()
* Description: Set the GPO's square wave alignment count.  The alginment
*              count is in seconds from 0 secs to 1 minute, where 0
*              disables PPS alignment beyond the initial alignment.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        cnt           - Square wave alignment count
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWAlignCnt(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    unsigned int      cnt);

/*
* Function:    TSYNC_GO_getSWTmAlgnEn()
* Description: Get the GPO's square wave time alignment enable.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        bEnable       - Pointer to the square wave time alignment enable
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWTmAlgnEn(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bEnable);

/*
* Function:    TSYNC_GO_setSWTmAlgnEn()
* Description: Set the GPO's square wave time alignment enable.  The time
```

```
*              alignment enable changes the function of the alignment
*              counter to align the square wave whenever the current time's
*              seconds value is a multiple of the alignment count.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        bEnable       - Square wave time alignment enable
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWTmAlgnEn(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               bEnable);

/*
* Function:    TSYNC_GO_setSWInit()
* Description: Set the GPO's square wave initialization.  This will
*              initialize, align, and restart the square wave on the next
*              PPS.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWInit(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo);

/*
* Function:    TSYNC_GO_getNumInst()
* Description: Get number of GPIO Outputs present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of instances result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);

/*
* Function:    TSYNC_GO_getSWOtpPW()
* Description: Get the GPO's square wave OTP pulse width.  Pulse width is
*              in nanoseconds.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        pw            - Pointer to the square wave pulse width
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_getSWOtpPW(
```

```
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    unsigned int      *pw);

/*
* Function:    TSYNC_GO_setSWOtpPW()
* Description: Get the GPO's square wave OTP pulse width.  Pulse width is
*              in nanoseconds from 20 nsec to 900,000,000 nsec.
*
* Parameters:
*   IN:  hnd           - Board handle
*        gpo           - GPO index
*        pw            - Square wave pulse width
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GO_setSWOtpPW(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    unsigned int      pw);

/* GPR Generic PTP Reference ==============================================*/
/*
* Function:    TSYNC_GPR_getNumInst()
* Description: Gets the number of PTP modules present in the system.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: nInstances    - The number of PTP modules present in the system
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getNumInst(
    TSYNC_BoardHandle hnd,
    unsigned int     *nInstances);

/*
* Function:    TSYNC_GPR_getClockIdentity()
* Description: Gets the Clock Identity of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Clock Identity
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClockIdentity(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplClockId  *pObj);

/*
* Function:    TSYNC_GPR_setClockIdentity()
* Description: Sets the Clock Identity of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pObj          - pointer to the Clock Identity
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setClockIdentity(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplClockId  *pObj);

/*
* Function:    TSYNC_GPR_getPriority()
* Description: Gets the Priority of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Priority structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getPriority(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplPriority *pObj);

/*
* Function:    TSYNC_GPR_setPriority()
* Description: Sets the Priority of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pObj          - pointer to the Priority structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setPriority(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplPriority *pObj);

/*
* Function:    TSYNC_GPR_getDomain()
* Description: Gets the Domain of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Domain value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getDomain(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_setDomain()
* Description: Sets the domain of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to the Domain value
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setDomain(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getClockMode()
* Description: Gets the Clock Mode of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Clock Mode value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClockMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_setClockMode()
* Description: Sets the Clock Mode of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to the Clock Mode value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setClockMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getClockSteps()
* Description: Gets the Clock Steps of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Clock Steps value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClockSteps(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_setClockSteps()
* Description: Gets the Clock Steps of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to the Clock Steps value
```

```
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setClockSteps(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getClockPorts()
* Description: Gets the number of clock Ports in the module.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Clock Ports value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClockPorts(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getClockQual()
* Description: Gets the Clock Quality structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Clock Quality structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClockQual(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_GplClockQual *pObj);

/*
* Function:    TSYNC_GPR_getStatistics()
* Description: Gets the Statistics structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Statistics structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getStatistics(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplStatistics *pObj);

/*
* Function:    TSYNC_GPR_getParentData()
* Description: Gets the Parent Data structure.
*
* Parameters:
*   IN:  hnd           - Board handle
```

```
*   OUT: pObj          - pointer to the Parent Data structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getParentData(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplParentData *pObj);

/*
* Function:    TSYNC_GPR_getTimeProp()
* Description: Gets the Time Properties structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Time Properties structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getTimeProp(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplTimeProp    *pObj);


/*
* Function:    TSYNC_GPR_getUserDesc()
* Description: Gets the User Description structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the User Description structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getUserDesc(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplUserDesc    *pObj);

/*
* Function:    TSYNC_GPR_setUserDesc()
* Description: Sets the User Description structure.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the User Description structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setUserDesc(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplUserDesc    *pObj);

/*
* Function:    TSYNC_GPR_getUnctMasterAdd()
* Description: Gets the current Unicast Master.
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Master Add structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getUnctMasterAdd(
    TSYNC_BoardHandle          hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctMasterAdd    *pObj);

/*
* Function:    TSYNC_GPR_setUnctMasterAdd()
* Description: Sets the current Unicast Master.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Master Add structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setUnctMasterAdd(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctMasterAdd    *pObj);


/*
* Function:    TSYNC_GPR_setUnctMasterDel()
* Description: Deletes a Unicast Master.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Master's Port Identity
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setUnctMasterDel(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_GplPortId      *pObj);

/*
* Function:    TSYNC_GPR_getUnctSlaveProp()
* Description: Gets properties when acting as a Unicast Slave.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Slave Prop structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getUnctSlaveProp(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctSlaveProp    *pObj);


/*
```

```
* Function:    TSYNC_GPR_getUnctMasterProp()
* Description: Gets properties when acting as a Unicast Master.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Master Prop structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getUnctMasterProp(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctMasterProp   *pObj);


/*
* Function:    TSYNC_GPR_getUnctMasterCfg()
* Description: Gets configuration when acting as a Unicast Master.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Master Cfg structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getUnctMasterCfg(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctMasterCfg    *pObj);


/*
* Function:    TSYNC_GPR_setUnctMasterCfg()
* Description: Sets configuration when acting as a Unicast Master.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Unicast Master Cfg structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setUnctMasterCfg(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplUnctMasterCfg    *pObj);


/*
* Function:    TSYNC_GPR_getPortState()
* Description: Gets the Port State.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Port State value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getPortState(
    TSYNC_BoardHandle         hnd,
```

```
    unsigned int             nInstance,
    unsigned int             *pVal);


/*
* Function:    TSYNC_GPR_getMsgRates()
* Description: Gets message rates.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Message Rates structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getMsgRates(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMsgRates    *pObj);


/*
* Function:    TSYNC_GPR_setMsgRates()
* Description: Sets message rates.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Message Rates structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setMsgRates(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMsgRates        *pObj);

/*
* Function:    TSYNC_GPR_getMsgTo()
* Description: Gets message Timeouts.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Message Timeouts structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getMsgTo(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMsgTo            *pObj);


/*
* Function:    TSYNC_GPR_setMsgTo()
* Description: Sets message Timeouts.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Message Timeouts structure
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setMsgTo(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMsgTo            *pObj);


/*
* Function:    TSYNC_GPR_getDelayMech()
* Description: Gets the Delay Mechanism.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Delay Mechanism value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getDelayMech(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_setDelayMech()
* Description: Sets the Delay Mechanism.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Port State value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setDelayMech(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_getBcastMech()
* Description: Gets the Broadcast Mechanism.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Broadcast Mechanism value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getBcastMech(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplBcastMech       *pObj);


/*
* Function:    TSYNC_GPR_setBcastMech()
* Description: Sets the Broadcast Mechanism
*
```

```
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Port State value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setBcastMech(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplBcastMech       *pObj);


/*
* Function:    TSYNC_GPR_getStaticIPV4()
* Description: Gets Static IP Configurations.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the IPV4 structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getStaticIPV4(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplIpV4             *pObj);

/*
* Function:    TSYNC_GPR_setStaticIPV4()
* Description: Sets Static IP Configurations.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the IPV4 structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setStaticIPV4(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplIpV4             *pObj);

/*
* Function:    TSYNC_GPR_getCurrentIPV4()
* Description: Gets current IP Configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the IPV4 structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getCurrentIPV4(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplIpV4             *pObj);

/*
* Function:    TSYNC_GPR_getDHCPEn()
```

```
* Description: Gets the DHCP Enable value.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the DHCP Enable value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getDHCPEn(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_setDHCPEn()
* Description: Sets the DHCP Enable value
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the DHCP Enable value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setDHCPEn(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_getMacAddr()
* Description: Gets MAC Address.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the MAC Address structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getMacAddr(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMacAddr          *pObj);

/*
* Function:    TSYNC_GPR_setMacAddr()
* Description: Sets MAC Address
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the MAC Address structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setMacAddr(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplMacAddr          *pObj);
```

```
/*
* Function:    TSYNC_GPR_getTTL()
* Description: Gets the TTL value.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the TTL value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getTTL(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_setTTL()
* Description: Sets the TTL value
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the TTL value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setTTL(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);

/*
* Function:    TSYNC_GPR_getEthTrans()
* Description: Gets the Ethernet Transport value.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Ethernet Transport value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getEthTrans(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_setEthTrans()
* Description: Sets the DHCP Enable value
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the Ethernet Transport value
*
* Returns: (TSYNC_SUCCESS) Success
*/
```

```
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setEthTrans(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_getSyncEth()
* Description: Gets the Sync-E Configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Sync-E structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getSyncEth(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplSyncEth          *pObj);

/*
* Function:    TSYNC_GPR_setSyncEth()
* Description: Sets the Sync-E Configuration
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Sync-E structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setSyncEth(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplSyncEth          *pObj);

/*
* Function:    TSYNC_GPR_getVLAN()
* Description: Gets the VLAN Configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the VLAN structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getVLAN(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplVlan             *pObj);

/*
* Function:    TSYNC_GPR_setVLAN()
* Description: Sets the VLAN configuration
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the VLAN structure
*
```

```
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setVLAN(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplVlan             *pObj);

/*
* Function:    TSYNC_GPR_getClassCfg()
* Description: Gets the Clock Class configuration.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the ClassCfg Value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getClassCfg(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_setClassCfg()
* Description: Sets the Clock Class configuration
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to the ClassCfg structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setClassCfg(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);


/*
* Function:    TSYNC_GPR_getModuleInfo()
* Description: Gets the Module Info.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Module Info structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getModuleInfo(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    TSYNC_GplModuleInfo       *pObj);


/*
* Function:    TSYNC_GPR_getControl()
* Description: Gets the Control information of the given module
*
* Parameters:
```

```
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Control structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getControl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplControl *pObj);

/*
* Function:    TSYNC_GPR_setControl()
* Description: Sets the Control information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pObj          - pointer to the Control structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setControl(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplControl *pObj);

/*
* Function:    TSYNC_GPR_getPortSpeed()
* Description: Gets the Port Speed information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to Port Speed
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getPortSpeed(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_setPortSpeed()
* Description: Sets the Port Speed information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to Port Speed value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setPortSpeed(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getSlaveStats()
* Description: Gets the Slave Stats information of the given module
*              Must use GPR_getSlaveSummary and GPR_setSlaveStats first.
```

```
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Slave Stats structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getSlaveStats(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_GplSlaveStats  *pObj);


/*
* Function:    TSYNC_GPR_setSlaveStats()
* Description: Sets the Slave number to get Stats for.
*              Must use GP_getSummary first.
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to Slave Number
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setSlaveStats(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    unsigned int      *pVal);

/*
* Function:    TSYNC_GPR_getSlaveSummary()
* Description: Gets the Slave Summary information of the given module
*              Also latches data for setSlaveStats.
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pObj          - pointer to the Slave Stats structure
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getSlaveSummary(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_GplSlaveSummary *pObj);


/*
* Function:    TSYNC_GPR_getDebug()
* Description: Gets the debug information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to Debug
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getDebug(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
```

```
    unsigned int             *pVal);


/*
* Function:    TSYNC_GPR_setDebug()
* Description: Sets the Debug information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to Debug value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setDebug(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_getProfile()
* Description: Gets the profile information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   OUT: pVal          - pointer to Profile
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_getProfile(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);


/*
* Function:    TSYNC_GPR_setProfile()
* Description: Sets the Profile information of the given module
*
* Parameters:
*   IN:  hnd           - Board handle
*   IN:  pVal          - pointer to Profile value
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_GPR_setProfile(
    TSYNC_BoardHandle         hnd,
    unsigned int              nInstance,
    unsigned int              *pVal);

/* Hardware Interface ==============================
 *
 * ======================*/

/*
* Function:    TSYNC_HW_getTime()
* Description: Get the current system time from the hardware.
*
* Parameters:
*   IN:  hnd               - Board handle
```

```
*   OUT: pObj              - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTime(
    TSYNC_BoardHandle  handle,
    TSYNC_HWTimeObj    *pObj);

/*
* Function:    TSYNC_HW_getTimeSec()
* Description: Get the current system time from the hardware in
*              seconds format.
*
* Parameters:
*   IN:  hnd               - Board handle
*   OUT: pObj              - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTimeSec(
    TSYNC_BoardHandle       handle,
    TSYNC_HWTimeSecondsObj *pObj);

/*
* Function:    TSYNC_HW_getTsEnable()
* Description: Get the current enable/disable state of timestamps.
*
* Parameters:
*   IN:  hnd               - Board handle
*   OUT: bEnable           - The enable result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTsEnable(
    TSYNC_BoardHandle  handle,
    int               *bEnable);

/*
* Function:    TSYNC_HW_setTsEnable()
* Description: Set the current enable/disable state of timestamps.
*
* Parameters:
*   IN:  hnd               - Board handle
*        bEnable           - The enable information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setTsEnable(
    TSYNC_BoardHandle handle,
    int               bEnable);

/*
* Function:    TSYNC_HW_setTsReq()
* Description: Manually generate a hardware timestamp.
*
* Parameters:
*   IN:  hnd               - Board handle
*
* Returns: (TSYNC_SUCCESS) Success
```

```
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setTsReq(
    TSYNC_BoardHandle handle);

/*
* Function:    TSYNC_HW_setTsClear()
* Description: Clear all collected timestamps from the specified source.
*
* Parameters:
*   IN:  hnd               - Board handle
*        source            - Timestamp source
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setTsClear(
    TSYNC_BoardHandle handle,
    TMSTMP_SRC        source);

/*
* Function:    TSYNC_HW_getTsCount()
* Description: Get the number of collected timestamps for the specified
*              source.
*
* Parameters:
*   IN:  hnd               - Board handle
*        source            - The timestamp source information
*   OUT: nCount            - The count result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTsCount(
    TSYNC_BoardHandle  handle,
    TMSTMP_SRC         source,
    unsigned int      *nCount);

/*
* Function:    TSYNC_HW_getTsData()
* Description: Get all collected timestamps for the specified source.
*
* Parameters:
*   IN:  hnd               - Board handle
*        source            - The timestamp source information
*   OUT: pObj              - Pointer to the timestamp data result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTsData(
    TSYNC_BoardHandle    handle,
    TMSTMP_SRC           source,
    TSYNC_HWTimeDataObj *pObj);

/*
* Function:    TSYNC_HW_getMatchTimeHi()
* Description: Get the match time value when the specified general purpose
*              output will transition to an active high state.
*
* Parameters:
*   IN:  hnd               - Board handle
*        index             - GPO index information
```

```
*   OUT: pObj              - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getMatchTimeHi(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);

/*
* Function:    TSYNC_HW_setMatchTimeHi()
* Description: Set the match time value when the specified general purpose
*              output will transition to an active high state.
*
* Parameters:
*   IN:  hnd               - Board handle
*        index             - GPO index information
*        pObj              - Pointer to the time information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setMatchTimeHi(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);

/*
* Function:    TSYNC_HW_getMatchTimeLo()
* Description: Get the match time value when the specified general purpose
*              output will transition to an active low state.
*
* Parameters:
*   IN:  hnd               - Board handle
*        index             - GPO index information
*   OUT: pObj              - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getMatchTimeLo(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);

/*
* Function:    TSYNC_HW_setMatchTimeLo()
* Description: Set the match time value when the specified general purpose
*              output will transition to an active low state.
*
* Parameters:
*   IN:  hnd               - Board handle
*        index             - GPO index information
*        pObj              - Pointer to the time information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setMatchTimeLo(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);
```

```
/*
* Function:    TSYNC_HW_getFpgaInfo()
* Description: Get the the FPGA ID and version information.
*
* Parameters:
*   IN:  hnd               - Board handle
*   OUT: id                - The ID result
*        rev               - The version result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getFpgaInfo(
    TSYNC_BoardHandle  handle,
    unsigned short    *id,
    unsigned short    *rev);

/*
* Function:    TSYNC_HW_getIntMask()
* Description: Get the hardware interrupt masking enabled state
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - The interrupt type information
*        index             - The interrupt index information
*   OUT: bEnable           - The enable result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getIntMask(
    TSYNC_BoardHandle  handle,
    INT_TYPE           intType,
    unsigned int       index,
    int               *bEnable);

/*
* Function:    TSYNC_HW_setIntMask()
* Description: Set the hardware interrupt masking enabled state
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - The interrupt type information
*        index             - The interrupt index information
*        bEnable           - The enable information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_setIntMask(
    TSYNC_BoardHandle handle,
    INT_TYPE          intType,
    unsigned int      index,
    int               bEnable);

/*
* Function:    TSYNC_HW_getTsSingle()
* Description: Get a single collected timestamp for a given source.
*
* Parameters:
*   IN:  hnd               - Board handle
*        source            - The timestamp source information
```

```
*   OUT: pObj              - Pointer to the time result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getTsSingle(
    TSYNC_BoardHandle  handle,
    TMSTMP_SRC         source,
    TSYNC_HWTimeObj   *pObj);

/*
* Function:    TSYNC_HW_getIntCnt()
* Description: Get the interrupt counter
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - The interrupt type information
*        index             - The interrupt index information
*   OUT: nIntCount         - The interrupt counter result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getIntCnt(
    TSYNC_BoardHandle  handle,
    INT_TYPE           intType,
    unsigned int       index,
    unsigned int      *nIntCount);

/*
* Function:    TSYNC_HW_getIntTs()
* Description: Get the interrupt timestamp
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - The interrupt type information
*        index             - The interrupt index information
*   OUT: pObj              - Pointer to the interrupt timestamp result
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_getIntTs(
    TSYNC_BoardHandle     handle,
    INT_TYPE              intType,
    unsigned int          index,
    TSYNC_TimeSecondsObj *pObj);


/*
* Function:    TSYNC_HW_clrIntCnt()
* Description: Clear the interrupt counter
*
* Parameters:
*   IN:  hnd               - Board handle
*        intType           - The interrupt type information
*        index             - The interrupt index information
*
* Returns: (TSYNC_SUCCESS) Success
*/
DLL_EXPORT
TSYNC_ERROR TSYNC_HW_clrIntCnt(
    TSYNC_BoardHandle  handle,
```

```
        INT_TYPE           intType,
        unsigned int       index);

   /*
    * Function:    TSYNC_HW_getTemperature()
    * Description: Read the current board temperature in counts.
    *
    *  To convert to degrees C
    *    freq = 1/(pTemp*.00000002) // convert counts to freq
    *    temperature = (freq/4)-273.15 // convert freq to Temp
    *
    * Parameters:
    *   IN:  handle           - Board handle
    *
    *   OUT: pTemp            - Pointer to contents of temperature
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_HW_getTemperature(
        TSYNC_BoardHandle handle,
        unsigned short *pTemp);


/* Include prototypes for non-KTS functionality */
#include "tsync_nonkts.h"

#ifdef __cplusplus
}
#endif

#endif  /* _defined_TSYNC_H */
```

4.1.2 tsync_nonkts.h

```
#ifndef _defined_TSYNC_NONKTS_H
#define _defined_TSYNC_NONKTS_H

/*****************************************************************************
**  Module:     tsync_nonkts.h
**  Date:       09/17/09
**  Purpose:    This is the TSYNC/TSAT PCI Card interface file for haedware
**              features that are not part of the KTS.  This file should *only*
**              be included from tsync.h.
**
** (C) Copyright 2009 Spectracom Corporation  All rights reserved.
**
*****************************************************************************/

/*** Start of non-kts entry area [DO NOT MOVE OR ALTER THIS LINE] ***/

    /* Hardware Interface ================================================*/


/*
    * Function:    TSYNC_HW_getTemperature()
    * Description: Read the current board temperature in counts.
    *
    *  To convert to degrees C
    *    freq = 1/(pTemp*.00000002) // convert counts to freq
    *    temperature = (freq/4)-273.15 // convert freq to Temp
```

```
    *
    * Parameters:
    *   IN:  handle           - Board handle
    *
    *   OUT: pTemp           - Pointer to contents of temperature
    *
    * Returns: (TSYNC_SUCCESS) Success
    */
    DLL_EXPORT
    TSYNC_ERROR TSYNC_HW_getTemperature(
        TSYNC_BoardHandle handle,
        unsigned short *pTemp);


/*** End of non-kts entry area [DO NOT MOVE OR ALTER THIS LINE] ***/

#endif  /* _defined_TSYNC_NONKTS_H */
```

4.1.3 Tsync_hw.h

```
#ifndef __TSYNC_HW__
#define __TSYNC_HW__ 1

#include "tsync_nonkts_hw.h"

#define TSYNC_TIMESTAMP_DATA_NUM        ( 512 )

typedef enum
{
    INT_1PPS = 0, // 1PPS Received
    INT_SVC_REQ = 1, // Timing System Service Request
    INT_LCL_UC_FIFO_EMPTY = 2, // Local / uC Bus FIFO Empty
    INT_LCL_UC_FIFO_OVER = 3, // Local / uC Bus FIFO Overflow
    INT_UC_LCL_FIFO_DATA = 4, // uC / Local Bus FIFO Data Ready
    INT_UC_LCL_FIFO_OVER = 5, // uC / Lcoal Bus FIFO Overflow
    INT_GPIO_IN = 6, // GPIO Input Event
    INT_TMSTMP = 7, // Timestamp Data Ready
    INT_GPIO_OUT = 8 // GPIO Output Event
} INT_TYPE;

typedef enum
{
    DEST_ID_FW = 0x1,           // Access to the firmware using
                                // HIDD defined transactions
                                // utilizing the HIP defined
                                // protocol
    DEST_ID_HW = 0x2,           // Access to hardware in directly
                                // addressable memory space
    DEST_ID_HW_NONKTS = 0x3     // Access to non-KTS hardware in directly
                                // addressable memory space
} DEST_ID;

typedef uint16_t FW_ITEM;

typedef enum
{
    HW_SYS_TIME = 0x0100,
    HW_SEC_TIME = 0x0101,
    HW_TMSTMP_EN = 0x0200,
    HW_TMSTMP_REQ = 0x0201,
    HW_TMSTMP_CLR = 0x0202,
```

```
    HW_TMSTMP_CNT = 0x0203,
    HW_TMSTMP_DATA = 0x0204,
    HW_TMSTMP_SINGLE = 0x0205,
    HW_GPO_MTCH_HI = 0x0300,
    HW_GPO_MTCH_LO = 0x0301,
    HW_FPGA_ID = 0x0400,
    HW_INT_MASK = 0x0401,
    HW_INT_CNT = 0x0402,
    HW_INT_CNT_CLR = 0x0403,
    HW_INT_TS = 0x0404,
    HW_KTS_TEMPERATURE  = 0x1100,

} HW_ITEM;

typedef union
{
    FW_ITEM fid;            // cai/iid pair as described in
                            // each HIDD defined transaction
    HW_ITEM hid;            // HW transaction item
    HW_NONKTS_ITEM hnkid;   // Non-KTS HW transaction item
} ITEM_ID;

typedef enum
{
    TMSTMP_SRC_HOST = 0x0,
    TMSTMP_SRC_GPI_0 = 0x1,
    TMSTMP_SRC_GPI_1 = 0x2,
    TMSTMP_SRC_GPI_2 = 0x3,
    TMSTMP_SRC_GPI_3 = 0x4,
    TMSTMP_SRC_HOST_AND_GPI_0 = 0x5,
    TMSTMP_SRC_COUNT
} TMSTMP_SRC;

typedef enum
{
    ID_PIN_ALL = -1,

    ID_PIN_MIN = 0,
    ID_PIN_0   = 0,
    ID_PIN_1   = 1,
    ID_PIN_2   = 2,
    ID_PIN_3   = 3,
    ID_PIN_4   = 4,
    ID_PIN_5   = 5,
    ID_PIN_6   = 6,
    ID_PIN_7   = 7,

    // Add more GPIO Inputs here

    ID_PIN_NUM = 4 // [FUTURE]      // Number of GPI pins

} ID_PIN;

typedef enum
{
    OD_PIN_ALL = -1,

    OD_PIN_MIN = 0,

    OD_PIN_0   = 0,
    OD_PIN_1   = 1,
    OD_PIN_2   = 2,
    OD_PIN_3   = 3,
```

```
    OD_PIN_4   = 4,
    OD_PIN_5   = 5,
    OD_PIN_6   = 6,
    OD_PIN_7   = 7,

    OD_PIN_NUM = 4                    // Number of GPO ([FUTURE] 8)

} OD_PIN;

#endif
```

4.2 TSync-PCIe API — Routine Descriptions

4.2.1 TSYNC_open

```
TSYNC_ERROR TSYNC_open(
    TSYNC_BoardHandle *hnd,
    char              *deviceName);
```

Description:
Open the TSYNC device.
Input Parameters:
*hnd: Handle
*deviceName: Name of the device.
Returns:
(TSYNC_SUCCESS) Success

4.2.2 TSYNC_close

```
TSYNC_ERROR TSYNC_close(
    TSYNC_BoardHandle hnd);
```

Description:
Close the TSYNC device.
Input Parameters:
*hw: Handle
Returns:
(TSYNC_SUCCESS) Success

4.2.3 TSYNC_get

```
TSYNC_ERROR TSYNC_get(
    TSYNC_BoardHandle  handle,
    DEST_ID            dest,
    ITEM_ID            iid,
    void              *inPayload,
    uint32_t           inLength,
    void              *outPayload,
    uint32_t           maxOutLength,
    uint32_t          *actualOutLength);
```

Description:
Generic get accessor.
Input Parameters:
hnd: Board handle
dest: destination of the call
iid: item id

inPayload: transaction specific payload
inLength: number of bytes in inPayload
maxOutLength: number bytes allowed in outPayload

Output Parameters:

outPayload: passed back transaction specific data
actualOutLength: the actual number of bytes passed back in outPayload

Returns:

(TSYNC_SUCCESS) Success

4.2.4 TSYNC_set

```
TSYNC_ERROR TSYNC_set(
    TSYNC_BoardHandle  handle,
    DEST_ID            dest,
    ITEM_ID            iid,
    void              *inPayload,
    uint32_t           inLength,
    void              *outPayload,
    uint32_t           maxOutLength,
    uint32_t          *actualOutLength);
```

Description:

Generic set accessor.

Input Parameters:

hnd: Board handle
dest: destination of the call
iid: item id
inPayload: transaction specific payload
inLength: number of bytes in inPayload
maxOutLength: number bytes allowed in outPayload

Output Parameters:

outPayload: passed back transaction specific data
actualOutLength: the actual number of bytes passed back in outPayload

Returns:

(TSYNC_SUCCESS) Success

4.2.5 TSYNC_waitFor

```
TSYNC_ERROR TSYNC_waitFor(
    TSYNC_BoardHandle handle,
    INT_TYPE          intType,
    uint32_t          index);
```

Description:

Blocking call to wait for specified interrupt.

Input Parameters:

hnd: Board handle
intType: the interrupt type to wait for (GPI/GPO are indexed)
index: the index of the interrupt (0 for non-indexed interrupts)

Returns:

(TSYNC_SUCCESS) Success

4.2.6 API Error Message Returns

There are two error messages that may be returned from the driver when issuing API calls. The two error messages are as follows:

A) **"Usage: Not a recognized call"**: This response indicates an invalid command or a syntax error occurred in the command that was sent.

B) **"Error: Invalid parameter"**: This response indicates the API call was valid, but a value in the entered call was not valid.

4.2.7 API Calls Supported by TSync

The TSync driver contains API calls for other Spectracom products.

Table 4.1 documents the sections of the API that are supported in the TSync product. All other API calls are unsupported.

API	Area of the board (Hardware, supervisory software, etc)	Description	Refer to Section
CS	Clock Service	Provides an abstract interface to the timing subsystem.	4.2.8
EC	LED Component	Drive the LED indicators.	4.2.8.25
FP	Frequency Output	Configures the output frequency.	0
FS	Flash Manager Service	Provides access to the images stored in all flash memory devices.	4.2.10
GI	General Purpose Input Component	Configure and monitor the general purpose input pins.	4.2.11
GO	General Purpose Output Component	Configure and monitor the general purpose output pins.	4.2.12
GR	GPS Reference Component	Execute the GPS receiver's protocol and determine 1PPS and time validity.	0
HA	Host Agent	Obtain the capabilities of the TSync.	0
HR	Host Reference	Uses information from the host PC to determine 1PPS and time validity.	4.2.15
HW	Hardware	Provide access to the direct HW accessible control/status of the timing subsystem.	0
IN	Initializer Service	Perform initial configuration and setup of each software module.	0
IP	IRIG Output Component	Generate and output IRIG streams.	4.2.18
IR	IRIG Reference Component	Control and process decoded IRIG input streams to determine 1PPS and time validity.	4.2.19
LS	Log Service	Provides a queue for errors and maintains system alarms.	4.2.20
PP	PPS Output Component	Control a 1Hz output.	4.2.21
PR	PPS Reference Component	Monitor the 1PPS input reference.	4.2.22
PTR	PTP Reference Component	Control and process decoded PTP network packets (either as an input reference or a time output).	4.2.23
RS	Reference Monitor Service	Determine the best available input reference and maintain the reference priority table.	0
SS	Supervisor Service	Maintain the time source, 1PPS source, Sync and Holdover states of the system.	4.2.25
US	Upgrade Service	Upgrade the firmware and FPGA images in the external flash memories.	0

XO	Oscillator Component	Analyze frequency measurements and make corrective adjustments to the timing system oscillator.	4.2.27
XS	Oscillator Monitor Service	Measure and provide the accuracy and stability of the timing system oscillator.	0

Table 4-1: Available API Calls

NOTE: All of the "typedef Enumerators" (variables used in the API calls to define parameters) can be found in the Header files located in sections 4.1.1 and 4.1.3 of this document.

4.2.8 Clock Service (CS) Calls

CS calls provide an abstract interface to the timing subsystem.

4.2.8.1 TSYNC_CS_getTime

```
TSYNC_ERROR TSYNC_CS_GetTime(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);
```

Description:
Get the DOY time from the firmware. TSYNC_getHWSystemTime is recommended for faster time reads.
Input Parameters:
*hw: Handle
Output Parameters:
*Timep: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

NOTE: Instance value of 1 returns BCD time, year, DOY, time information. Instance value of 2 returns the time in seconds and nanoseconds.

4.2.8.2 TSYNC_CS_setTime

```
TSYNC_ERROR TSYNC_CS_setTime(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);
```

Description:
Set the DOY time.
Input Parameters:
*hw: Handle
*Timep: Pointer to the time information

Returns:
(TSYNC_SUCCESS) Success

Note: Entered as: `CS_setTime 0 1 <year> <DOY> <Hour> <Minutes> <Seconds>`

Time Scales:
Note: Where <scale>
0= UTC
1= TAI
2= GPS
3= Local

4.2.8.3 TSYNC_CS_getTimeScale

```
TSYNC_CS_getTimeScale(
    TSYNC_BoardHandle   hnd,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Get the board's current time scale.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time scale result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.4 TSYNC_CS_setTimeScale

```
TSYNC_CS_setTimeScale(
    TSYNC_BoardHandle   hnd,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Set the board's time scale.
Input Parameters:
hnd: Board handle
pObj: Pointer to the time scale information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.5 TSYNC_CS_getTimeScaleOff

```
TSYNC_CS_getTimeScaleOff(
    TSYNC_BoardHandle          hnd,
    TSYNC_TimeScaleOffsetObj *pObj);
```

Description:
Get the specified time scale's offset from UTC. Offset is in seconds.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time scale offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.6 TSYNC_CS_setTimeScaleOff

```
TSYNC_CS_setTimeScaleOff(
    TSYNC_BoardHandle          hnd,
    TSYNC_TimeScaleOffsetObj *pObj);
```

Description:
Set the specified time scale's offset from UTC. Offset is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the time scale offset information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.7 TSYNC_CS_subsecAdj

```
TSYNC_ERROR TSYNC_CS_subsecAdj(
    TSYNC_BoardHandle       hnd,
    TSYNC_TimeSubsecAdjObj *pObj);
```

Description:
Make a one-time adjustment to the 1PPS on-time point. Adjustment is in nanoseconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the subsecond adjustment information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.8 TSYNC_CS_getTimeDiscont

This API call has been deprecated.

4.2.8.9 TSYNC_CS_setTimeDiscont

This API call has been deprecated.

4.2.8.10 TSYNC_CS_getLeapSec

```
TSYNC_ERROR TSYNC_CS_getLeapSec(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeLeapSecondObj *pObj);
```

Description:
Get the current leap second information. Offset is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the leap seconds result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.11 TSYNC_CS_setLeapSec

```
TSYNC_ERROR TSYNC_CS_setLeapSec(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeLeapSecondObj *pObj);
```

Description:
Set a new leap second. Offset is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the leap seconds information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.12 TSYNC_CS_getNextSec

```
TSYNC_ERROR TSYNC_CS_getNextSec(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeObj     *Timep);
```

Description:
Get the DOY time for the next second from the firmware.
Input Parameters:
hnd: Board handle
Output Parameters:
Timep: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.13 TSYNC_CS_getTimeZoneOff

```
TSYNC_ERROR TSYNC_CS_getTimeZoneOff(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeZoneOffsetObj *pObj);
```

Description:
Get the current time zone offset from UTC. Offset is in seconds.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time zone offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.14 TSYNC_CS_setTimeZoneOff

```
TSYNC_ERROR TSYNC_CS_setTimeZoneOff(
    TSYNC_BoardHandle        hnd,
    TSYNC_TimeZoneOffsetObj *pObj);
```

Description:
Set the current time zone offset from UTC. Offset is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the time zone offset information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.15 TSYNC_CS_getDstRule

```
TSYNC_ERROR TSYNC_CS_getDstRule(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDSTRuleObj *pObj);
```

Description:
Get the current DST rule. DST Offset is in seconds.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time zone offset rule result
Returns:
(TSYNC_SUCCESS) Success

Where 1=In DST or 0=Standard Time

4.2.8.16 TSYNC_CS_setDstRule

```
TSYNC_ERROR TSYNC_CS_setDstRule(
    TSYNC_BoardHandle     hnd,
    TSYNC_TimeDSTRuleObj *pObj);
```

Description:
Set the DST rule. DST offset is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the time zone offset rule information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.17 TSYNC_CS_getYear

```
TSYNC_ERROR TSYNC_CS_getYear(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeYearObj *pObj);
```

Description:
Get the current year.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the year result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.18 TSYNC_CS_setYear

```
TSYNC_ERROR TSYNC_CS_setYear(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeYearObj *pObj);
```

Description:
Set the year.
Input Parameters:
hnd: Board handle
pObj: Pointer to the year information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.19 TSYNC_CS_getDstState

```
TSYNC_ERROR TSYNC_CS_getDstState(
    TSYNC_BoardHandle      hnd,
    TSYNC_TimeDSTStateObj *pObj);
```

Description:
Get the current DST state.
Input Parameters:
hnd: Board handle
pObj: Pointer to the DST state information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.20 TSYNC_CS_setDstState

```
TSYNC_ERROR TSYNC_CS_setDstState(
    TSYNC_BoardHandle      hnd,
    TSYNC_TimeDSTStateObj *pObj);
```

Description:
Set the DST state.
Input Parameters:
hnd: Board handle
pObj: Pointer to the DST state information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.21 TSYNC_CS_getTimeSec

```
TSYNC_ERROR TSYNC_CS_getTimeSec(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nSeconds,
    unsigned int      *nNanos);
```

Description:
Get the time in seconds and nanoseconds format (Note: API call "TSYNC_getHWSecondsTime" is recommended for faster time reads).
Input Parameters:
hnd: Board handle
Output Parameters:
nSeconds: The seconds result
nNanos: The nanoseconds result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.22 TSYNC_CS_setTimeSec

```
TSYNC_ERROR TSYNC_CS_setTimeSec(
    TSYNC_BoardHandle hnd,
    unsigned int      nSeconds,
    unsigned int      nNanos);
```

Description:
Set the time in seconds and nanoseconds format.
Input Parameters:
hnd: Board handle
nSeconds: The seconds information
nNanos: The nanoseconds information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.23 TSYNC_CS_getTimeBcd

```
TSYNC_ERROR TSYNC_CS_getTimeBcd(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeBCDObj  *pObj);
```

Description:
Get the time in BCD format.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.8.24 TSYNC_CS_setTimeBcd

```
TSYNC_ERROR TSYNC_CS_setTimeBcd(
    TSYNC_BoardHandle  hnd,
    TSYNC_TimeBCDObj  *pObj);
```

Description:
Set the time in BCD format.
Input Parameters:
hnd: Board handle
pObj: Pointer to the time information
Returns:
(TSYNC_SUCCESS) Success

4.2.8.25 LED Components (EC) Calls

EC calls drive the LED indicators.

Note: Where Display State:
0= LED off solid
1= LED on solid
2= LED Blinks on/off (at a 2 Hz rate)
3= LED blinks a code (2 Hz rate with a 2 second pause)

Display Mode:
0= Display Sync Status
1= Display Holdover Status
2= Display Alarm Status
3= Blink on 1PPS
3= Manual Control

TSync Defaults:
Green LED: Sync Starus
Yellow LED: Holdover Status
Red LED: Alarm Status

4.2.8.26 TSYNC_EC_getMode

```
TSYNC_ERROR TSYNC_EC_getMode(
    TSYNC_BoardHandle  hnd,
    LE_INDEX           index,
    EC_MODE            *mode);
```

Description:
Get the LED usage mode state.
Input Parameters:
hnd: Board handle
index: The LED index
Output Parameters:
mode: The usage mode
Returns:
(TSYNC_SUCCESS) Success

4.2.8.27 TSYNC_EC_setMode

```
TSYNC_ERROR TSYNC_EC_setMode(
    TSYNC_BoardHandle hnd,
    LE_INDEX          index,
    EC_MODE           mode);
```

Description:
Set the LED usage mode state.
Input Parameters:
hnd: Board handle
index: The LED index
mode: The usage mode
Returns:
(TSYNC_SUCCESS) Success

4.2.8.28 TSYNC_EC_getState

```
TSYNC_ERROR TSYNC_EC_getState(
    TSYNC_BoardHandle  hnd,
    LE_INDEX           index,
    EC_STATE          *state);
```

Description:
Get the LED display state.
Input Parameters:
hnd: Board handle
index: The LED index
Output Parameters:
state: The display state
Returns:
(TSYNC_SUCCESS) Success

4.2.8.29 TSYNC_EC_setState

```
TSYNC_ERROR TSYNC_EC_setState(
    TSYNC_BoardHandle hnd,
    LE_INDEX          index,
    EC_STATE          state);
```

Description:
Set the LED display state. Settable only in manual LED mode.
Input Parameters:
hnd: Board handle
index: The LED index
state: The display state
Returns:
(TSYNC_SUCCESS) Success

4.2.9 Frequency Output (FP) Calls

FP calls configure the 10 MHz output.

4.2.9.1 TSYNC_FP_getSigCtrl

```
TSYNC_ERROR TSYNC_FP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    SIG_CTL           *sig);
```

Description:
Get the Fixed Frequency output's signature control state.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
sig: The signature control result
Returns:
(TSYNC_SUCCESS) Success

4.2.9.2 TSYNC_FP_setSigCtrl

```
TSYNC_ERROR TSYNC_FP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    SIG_CTL           sig);
```

Description:
Set the Fixed Frequency output's signature control state.
Input Parameters:
hnd: Board handle
nInstance: The instance number
sig: The signature control information
Returns:
(TSYNC_SUCCESS) Success

4.2.9.3 TSYNC_FP_getFreq

```
TSYNC_ERROR TSYNC_FP_getFreq(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    unsigned int      *freq);
```

Description:

Get the Fixed Frequency output's frequency. Frequency is in Hertz.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

freq: The frequency result

Returns:

(TSYNC_SUCCESS) Success

4.2.9.4 TSYNC_FP_getNumInst

```
TSYNC_ERROR TSYNC_FP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:

Get number of fixed-freq output instances present in the system.

Input Parameters:

hnd: Board handle

Output Parameters:

nInstances: The number of instances result

Returns:

(TSYNC_SUCCESS) Success

4.2.10 Flash Manager (FS) Calls

FS calls provide access to the images stored in all flash memory devices.

4.2.10.1 TSYNC_FS_getCrc

```
TSYNC_ERROR TSYNC_FS_getCrc(
    TSYNC_BoardHandle  hnd,
    TSYNC_FSImageObj  *pObj,
    TSYNC_FSCRCObj    *pObj2);
```

Description:

Get the CRC for a particular flash image.

Input Parameters:

hnd: Board handle
pObj: Pointer to the image information

Output Parameters:

pObj2: Pointer to the crc result

Returns:

(TSYNC_SUCCESS) Success

4.2.10.2 TSYNC_FS_calcCrc

```
TSYNC_ERROR TSYNC_FS_calcCrc(
    TSYNC_BoardHandle  hnd,
    TSYNC_FSImageObj  *pObj,
    TSYNC_FSCRCObj    *pObj2);
```

Description:
Calculate the CRC for a particular flash image.
Input Parameters:
hnd: Board handle
pObj: Pointer to the image information
Output Parameters:
pObj2: Pointer to the crc result
Returns:
(TSYNC_SUCCESS) Success

4.2.10.3 TSYNC_FS_getHeader

```
TSYNC_ERROR TSYNC_FS_getHeader(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageObj       *pObj,
    TSYNC_FSImageHeaderObj *pObj2);
```

Description:
Get the image header for a particular flash image.
Input Parameters:
hnd: Board handle
pObj: Pointer to the image information
Output Parameters:
pObj2: Pointer to the image header result
Returns:
(TSYNC_SUCCESS) Success

4.2.10.4 TSYNC_FS_getVersion

```
TSYNC_ERROR TSYNC_FS_getVersion(
    TSYNC_BoardHandle   hnd,
    TSYNC_FSImageObj   *pObj,
    TSYNC_FSVersionObj *pObj2);
```

Description:
Get the image version for a particular flash image.
Input Parameters:
hnd: Board handle
pObj: Pointer to the image information

Output Parameters:
pObj2: Pointer to the image version result
Returns:
(TSYNC_SUCCESS) Success

4.2.11 General Purpose Input (GI) Calls

GI calls configure and monitor the general purpose input (GPI) pins.

Note: Where <edge>
0= Falling edge
1= Rising edge
2= both (uncommon)

Where <valid>
0= Not valid
1= Valid

Where <pin> can be a value of “0” through “3” (“Pin” correlates to the GPI Pin number on the board).

Where <index> should always be a value of “0” with only one board installed.

4.2.11.1 TSYNC_GI_getValue

```
TSYNC_ERROR TSYNC_GI_getValue(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    int               *bEnabled);
```

Description:
Get the specified GPI’s current input value.
Input Parameters:
hnd: Board handle
index: The input index
Output Parameters:
bEnabled: The value result
Returns:
(TSYNC_SUCCESS) Success

4.2.11.2 TSYNC_GI_getEdge

```
TSYNC_ERROR TSYNC_GI_getEdge(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    EDGE              *edge);
```

Description:
Get the GPI's trigger edge used when detecting input changes.
Input Parameters:
hnd: Board handle
index: The input index
Output Parameters:
edge: The edge result
Returns:
(TSYNC_SUCCESS) Success

4.2.11.3 TSYNC_GI_setEdge

```
TSYNC_ERROR TSYNC_GI_setEdge(
    TSYNC_BoardHandle hnd,
    ID_PIN            index,
    EDGE              edge);
```

Description:
Set the GPI's trigger edge used when detecting input changes.
Input Parameters:
hnd: Board handle
index: The input index
edge: The edge information
Returns:
(TSYNC_SUCCESS) Success

4.2.11.4 TSYNC_GI_getTsEnable

```
TSYNC_ERROR TSYNC_GI_getTsEnable(
    TSYNC_BoardHandle  hnd,
    ID_PIN             index,
    int               *bEnable);
```

Description:
Get the GPI's timestamp enable state when time stamping input changes.
Input Parameters:
hnd: Board handle
index: The input index
Output Parameters:
bEnable: The enabled result
Returns:
(TSYNC_SUCCESS) Success

4.2.11.5 TSYNC_GI_setTsEnable

```
TSYNC_ERROR TSYNC_GI_setTsEnable(
    TSYNC_BoardHandle hnd,
```

```
    ID_PIN              index,
    int                 bEnable);
```

Description:
Set the GPI's timestamp enable state used when time stamping input changes.
Input Parameters:
hnd: Board handle
index: The input index
bEnable: The enabled information
Returns:
(TSYNC_SUCCESS) Success

4.2.11.6 TSYNC_GI_getNumInst

```
TSYNC_ERROR TSYNC_GI_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int       *nInstances);
```

Description:
Get number of GPIO Inputs present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.12 General Purpose Output (GO) Calls

GO calls configure and monitor the general purpose output (GPO) pins.

Note: Where <edge>
0= Falling edge
1= Rising edge
2= both (uncommon)

Where <valid>
0= Not valid
1= Valid

Where <pin> can be a value of "0" through "3" ("Pin" correlates to the GPI Pin number on the board).

Where <index> should always be a value of "0" with only one board installed.

4.2.12.1 TSYNC_GO_setSigCtrl

```
TSYNC_ERROR TSYNC_GO_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    SIG_CTL           *sig);
```

Description:
Get the GPO's signature control state.
Input Parameters:
hnd: Board handle
gpo: GPO index
Output Parameters:
sig: The signature control result
Returns:
(TSYNC_SUCCESS) Success

4.2.12.2 TSYNC_GO_setSigCtrl

```
TSYNC_ERROR TSYNC_GO_setSigCtrl(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    SIG_CTL           sig);
```

Description:
Set the GPO's signature control state.
Input Parameters:
hnd: Board handle
gpo: GPO index
sig: The signature control information
Returns:
(TSYNC_SUCCESS) Success

4.2.12.3 TSYNC_GO_getEnable

```
TSYNC_ERROR TSYNC_GO_getEnable(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bEnable);
```

Description:
Get the GPO's enable state.
Input Parameters:
hnd: Board handle
gpo: GPO index
Output Parameters:
bEnable: The output enable result

Returns:
(TSYNC_SUCCESS) Success

4.2.12.4 TSYNC_GO_setEnable

```
TSYNC_ERROR TSYNC_GO_setEnable(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               bEnable);
```

Description:
Set the GPO's enable state.
Input Parameters:
hnd: Board handle
gpo: GPO index
bEnable: The output enable information
Returns:
(TSYNC_SUCCESS) Success

4.2.12.5 TSYNC_GO_getValue

```
TSYNC_ERROR TSYNC_GO_getValue(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bValue);
```

Description:
Get the GPO's current output value.
Input Parameters:
hnd: Board handle
gpo: GPO index
Output Parameters:
bValue: The value result
Returns:
(TSYNC_SUCCESS) Success

4.2.12.6 TSYNC_GO_getMode

```
TSYNC_ERROR TSYNC_GO_getMode(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    OD_MODE           *mode);
```

Description:
Get the GPO's mode state.
Input Parameters:

hnd: Board handle
gpo: GPO index

Output Parameters:

mode: The mode result

Returns:

(TSYNC_SUCCESS) Success

4.2.12.7 TSYNC_GO_setMode

```
TSYNC_ERROR TSYNC_GO_setMode(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    OD_MODE           mode);
```

Description:

Set the GPO's mode state.

Input Parameters:

hnd: Board handle
gpo: GPO index
mode: The mode information

Returns:

(TSYNC_SUCCESS) Success

4.2.12.8 TSYNC_GO_getDvmValue

```
TSYNC_ERROR TSYNC_GO_getDvmValue(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    int               *bValue);
```

Description:

Get the GPO's Direct Value Mode (DVM) value state.

Input Parameters:

hnd: Board handle
gpo: GPO index

Output Parameters:

bValue: The DVM result

Returns:

(TSYNC_SUCCESS) Success

4.2.12.9 TSYNC_GO_setDvmValue

```
TSYNC_ERROR TSYNC_GO_setDvmValue(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    int               bValue);
```

Description:
Set the GPO's Direct Value Mode (DVM) value state.
Input Parameters:
hnd: Board handle
gpo: GPO index
bValue: The DVM information
Returns:
(TSYNC_SUCCESS) Success

4.2.12.10 TSYNC_GO_getMatchEnable

```
TSYNC_ERROR TSYNC_GO_getMatchEnable(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    LEVEL              lvl,
    int               *bEnable);
```

Description:
Get the GPO's match enable state and level.
Input Parameters:
hnd: Board handle
gpo: GPO index
lvl: Low or High Match Time
Output Parameters:
bEnable: Match enable
Returns:
(TSYNC_SUCCESS) Success

4.2.12.11 TSYNC_GO_setMatchEnable

```
TSYNC_ERROR TSYNC_GO_setMatchEnable(
    TSYNC_BoardHandle  hnd,
    OD_PIN             gpo,
    LEVEL              lvl,
    int               *bEnable);
```

Description:
Set the GPO's match enable state and level.
Input Parameters:
hnd: Board handle

gpo: GPO index
lvl: Low orHigh Match Time
bEnable: Match enable

Returns:

(TSYNC_SUCCESS) Success

4.2.12.12 TSYNC_GO_getSquareWave

```
TSYNC_ERROR TSYNC_GO_getSquareWave(
    TSYNC_BoardHandle   hnd,
    OD_PIN              gpo,
    TSYNC_GPOSquareObj *pObj);
```

Description:

Get the GPO's square wave output configuration structure. Offset, period, and duty cycle are in nanoseconds.

Input Parameters:

hnd: Board handle
gpo: GPO index

Output Parameters:

pObj: Pointer to the configuration result

Returns:

(TSYNC_SUCCESS) Success

4.2.12.13 TSYNC_GO_setSquareWave

```
TSYNC_ERROR TSYNC_GO_setSquareWave(
    TSYNC_BoardHandle   hnd,
    OD_PIN              gpo,
    TSYNC_GPOSquareObj *pObj);
```

Description:

Get the GPO's square wave output configuration structure. Offset, period, and duty cycle are in nanoseconds. Offset is from -500 msec to +500 msec. Period is from 100 nsec to 1 sec. Pulse width is from 10 nsec to 999,999,990 nsec.

Input Parameters:

hnd: Board handle
gpo: GPO index
pObj: Pointer to the configuration information

Returns:

(TSYNC_SUCCESS) Success

4.2.12.14 TSYNC_GO_getSWOtpPW

```
TSYNC_ERROR TSYNC_GO_getSWOtpPW(
    TSYNC_BoardHandle  hnd,
```

```
    OD_PIN              gpo,
    unsigned int       *pw);
```

Description:
Get the GPO's square wave OTP pulse width. Pulse width is in nanoseconds.
Input Parameters:
hnd: Board handle
gpo: GPO index
pw: Pointer to the square wave pulse width

Returns:
(TSYNC_SUCCESS) Success

4.2.12.15 TSYNC_GO_setSWOtpPW

```
TSYNC_ERROR TSYNC_GO_setSWOtpPW(
    TSYNC_BoardHandle hnd,
    OD_PIN            gpo,
    OD_MODE           mode);
```

Description:
Set the GPO's square wave OTP pulse width. Pulse width is in nanoseconds from 20 nsec to 900,000,000 nsec.
Input Parameters:
hnd: Board handle
gpo: GPO index
mode: Square wave pulse width
Returns:
(TSYNC_SUCCESS) Success

4.2.12.16 TSYNC_GO_getNumInst

```
TSYNC_ERROR TSYNC_GO_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of GPIO Outputs present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.13 GPS Reference Component (GR) Calls

GR calls execute the GPS receiver's protocol and determine 1PPS and time validity.

Note: Where <setmode>

0= Single satellite mode
1= Standard mode
2= Continuous mode
3= Averaging mode
4= Timing mode
5= Standby mode
6= Self mode

Where <dyn>

0= Land
1= Sea
2= Air
3=Stationary

Where <index> should always be a value of "0" with only one board installed.

4.2.13.1 TSYNC_GR_getOffset

```
TSYNC_ERROR TSYNC_GR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);
```

Description:
Get the GPS reference's 1PPS input offset. Offset is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nOffset: The offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.2 TSYNC_GR_setOffset

```
TSYNC_ERROR TSYNC_GR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);
```

Description:

Set the GPS reference's 1PPS input offset. Offset is in nanoseconds from -500 msec to +500 msec.

Input Parameters:

hnd: Board handle
nInstance: The instance number
nOffset: The offset information

Returns:

(TSYNC_SUCCESS) Success

4.2.13.3 TSYNC_GR_getValidity

```
TSYNC_ERROR TSYNC_GR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);
```

Description:

Get the GPS validity structure.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

bTimeValid: The time reference result
bPpsValid: The pps reference result

Returns:

(TSYNC_SUCCESS) Success

4.2.13.4 TSYNC_GR_getPosition

```
TSYNC_ERROR TSYNC_GR_getPosition(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_LLAObj      *pObj);
```

Description:

Get the GPS position. Latitude and longitude are in radians. Altitude is in meters.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

pObj: Pointer to the position result

Returns:

(TSYNC_SUCCESS) Success

4.2.13.5 TSYNC_GR_setPosition

```
TSYNC_ERROR TSYNC_GR_setPosition(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_LLAObj      *pObj);
```

Description:
Set the GPS position. Latitude and longitude are in radians. Altitude is in meters.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the position information
Returns:
(TSYNC_SUCCESS) Success

4.2.13.6 TSYNC_GR_getMode

```
TSYNC_ERROR TSYNC_GR_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    GL_MODE           *mode
    GL_DYN            *dyn);
```

Description:
Get the GPS receiver mode.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
mode: The receiver mode result
dyn : The receiver dynamics result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.7 TSYNC_GR_setMode

```
TSYNC_ERROR TSYNC_GR_setMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    GL_MODE           mode
    GL_DYN            *dyn);
```

Description:
Set the GPS receiver mode.
Input Parameters:
hnd: Board handle
nInstance: The instance number
mode: The receiver mode information
dyn : The receiver dynamics information
Returns:
(TSYNC_SUCCESS) Success

4.2.13.8 TSYNC_GR_getDynamics

This API call is deprecated (use `TSYNC_GR_getMode` instead)

4.2.13.9 TSYNC_GR_setDynamics

This API call is deprecated (use `TSYNC_GR_setMode` instead)

4.2.13.10 TSYNC_GR_getFixData

```
TSYNC_ERROR TSYNC_GR_getFixData(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_FixDataObj  *pObj);
```

Description:
Get the GPS position fix data.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the position fix data result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.11 TSYNC_GR_getSatData

```
TSYNC_ERROR TSYNC_GR_getSatData(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_SatDataObj  *pObj);
```

Description:
Get the GPS satellite data.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the satellite data result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.12 TSYNC_GR_getSurveyProg

```
TSYNC_ERROR TSYNC_GR_getSurveyProg(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int      *nProgress);
```

Description:
Get the GPS survey progress.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nProgress: The survey progress result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.13 TSYNC_GR_getMfrMdl

```
TSYNC_ERROR TSYNC_GR_getMfrMdl(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_ManModObj   *pObj);
```

Description:
Get the GPS receiver manufacturer and model.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:

pObj: Pointer to the manufacturer and model result

Returns:

(TSYNC_SUCCESS) Success

4.2.13.14 TSYNC_GR_getRcvInfo

```
TSYNC_ERROR TSYNC_GR_getRcvInfo(
    TSYNC_BoardHandle      hnd,
    unsigned int           nInstance,
    TSYNC_ReceiverInfoObj *pObj);
```

Description:

Get the GPS receiver info.

Input Parameters:

hnd: Board handle

nInstance: The instance number

Output Parameters:

pObj: Pointer to the receiver info result

Returns:

(TSYNC_SUCCESS) Success

4.2.13.15 TSYNC_GR_getCustom

```
TSYNC_ERROR TSYNC_GR_getCustom(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_CustomMessageObj *pObj);
```

Description:

Get the last unhandled GPS message.

Input Parameters:

hnd: Board handle

nInstance: The instance number

Output Parameters:

pObj: Pointer to the custom message result

Returns:

(TSYNC_SUCCESS) Success

4.2.13.16 TSYNC_GR_setCustom

```
TSYNC_ERROR TSYNC_GR_setCustom(
    TSYNC_BoardHandle       hnd,
    unsigned int            nInstance,
    TSYNC_CustomMessageObj *pObj);
```

Description:
Send a custom message to the GPS.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the custom message result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.17 TSYNC_GR_getNumInst

```
TSYNC_ERROR TSYNC_GR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of GPS references instances present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.13.18 TSYNC_GR_delPos

```
TSYNC_ERROR TSYNC_GR_delPos(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance);
```

Description:
Clear any position information that is stored in persistent inside the GPS receiver.
Input Parameters:
hnd: Board handle
Returns:
(TSYNC_SUCCESS) Success

4.2.13.19 TSYNC_GR_getRefId

```
TSYNC_ERROR TSYNC_GR_getRefId(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_RefIdObj   *pObj);
```

Description:
Get reference identifier for a GPS reference instance.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: pointer to the Reference ID
Returns:
(TSYNC_SUCCESS) Success

4.2.13.20 TSYNC_GR_Reset

```
TSYNC_ERROR TSYNC_GR_delPos(
    TSYNC_BoardHandle hnd,
    GL_RESET          reset);
```

Description:
Reset the GPS receiver.
Input Parameters:
hnd: Board handle
nInstance: The instance number
reset: The Reset Type
Returns:
(TSYNC_SUCCESS) Success

4.2.13.21 TSYNC_GR_getAntenna

```
TSYNC_ERROR TSYNC_GR_getAntenna(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance
    GL_ANT_STATUS    *status);
```

Description:
Get the GPS receiver antenna status.
Input Parameters:
hnd: Board handle
nInstance: The instance number
status: pointer to the Antenna Status
Returns:
(TSYNC_SUCCESS) Success

4.2.13.22 TSYNC_GR_getConstSel

```
TSYNC_ERROR TSYNC_GR_getConstSel(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int       *constellation);
```

Description:
Get the GPS *Constellation selection.*
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
*constellation: Pointer to the Constellation selection information
Returns:
(TSYNC_SUCCESS) Success

4.2.13.23 TSYNC_GR_setConstSel

```
TSYNC_ERROR TSYNC_GR_setConstSel(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    unsigned int       constellation);
```

Description:
Send the Constellation selection.
Input Parameters:
hnd: Board handle
nInstance: The instance number
`constellation`: Pointer to the Constellation selection information
Returns:
(TSYNC_SUCCESS) Success

4.2.14 Host Agent (HA) Calls

HA calls are used to obtain the capabilities of TSync boards.

4.2.14.1 TSYNC_HA_GetCaps

```
TSYNC_ERROR TSYNC_HA_getCaps(
    TSYNC_BoardHandle        hnd,
    unsigned int             pageNum,
    TSYNC_CapabilityPageObj *pObj);
```

Description:
Gets a single page of capability results.
Input Parameters:
hnd: Board handle
pageNum: Capability result page
Output Parameters:
**pCap: Pointer to the page of capability results
Returns:
(TSYNC_SUCCESS) Success

4.2.15 Host Reference (HR) Calls

HR calls provide the ability to get and set the validity of the host input reference.

4.2.15.1 TSYNC_HR_getValidity

```
TSYNC_ERROR TSYNC_HR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);
```

Description:
Get the reference validity from the Host.
Input Parameters:
hnd: Board handle
nInstance: the Instance Number
Output Parameters:
bTimeValid: The time reference result
bPpsValid: The pps reference result
Returns:
(TSYNC_SUCCESS) Success

4.2.15.2 TSYNC_HR_setValidity

```
TSYNC_ERROR TSYNC_HR_setValidity(
    TSYNC_BoardHandle hnd,
    unsigned int       nInstance,
    int               bTimeValid);
```

Description:
Set the reference validity of the Host.
Input Parameters:
hnd: Board handle
nInstance: the Instance Number
bTimeValid: The time reference information
Returns:
(TSYNC_SUCCESS) Success

4.2.15.3 TSYNC_HR_setTime

```
TSYNC_ERROR TSYNC_HR_setTime(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    TSYNC_TimeObj    *pObj);
```

Description:
Set the time from the Host.
Input Parameters:
hnd: Board handle
nInstance: the Instance Number
pObj: the Time to set
Returns:
(TSYNC_SUCCESS) Success

4.2.15.4 TSYNC_HR_getLocal

```
TSYNC_ERROR TSYNC_HR_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Get the Host reference's local time zone and DST rule.
Timezone and DST offsets are in seconds.
Input Parameters:
hnd: Board handle
nInstance: the Instance Number
Output Parameters:

Returns:
(TSYNC_SUCCESS) Success

4.2.15.5 TSYNC_HR_setLocal

```
TSYNC_ERROR TSYNC_HR_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Set the Host reference's local time zone and DST rule.
Timezone and DST offsets are in seconds.

Input Parameters:
hnd: Board handle
nInstance: the Instance Number
pObj: the Local Clock information
Returns:
(TSYNC_SUCCESS) Success

4.2.15.6 TSYNC_HR_getTimeScale

```
TSYNC_ERROR TSYNC_HR_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj
```

Description:
Get the Host reference's time scale.
Input Parameters:
hnd: Board handle
nInstance: the Instance Number
Output Parameters:
pObj: The Time Scale information
Returns:
(TSYNC_SUCCESS) Success

4.2.15.7 TSYNC_HR_setTimeScale

```
TSYNC_ERROR TSYNC_HR_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj
```

Description:

Get the Host reference's time scale.

Input Parameters:

hnd: Board handle
nInstance: the Instance Number
pObj: The Time Scale information

Returns:

(TSYNC_SUCCESS) Success

4.2.15.8 TSYNC_HR_getRefID

```
TSYNC_ERROR TSYNC_HR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    TSYNC_RefIdObj    *pObj);
```

Description:

Get reference identifier for a Host reference instance.

Input Parameters:

hnd: Board handle
nInstance: the Instance Number

Output Parameters:

pObj: The Reference Identifier result

Returns:

(TSYNC_SUCCESS) Success

4.2.15.9 TSYNC_HR_getNumInst

```
TSYNC_ERROR TSYNC_HR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:

Get number of Host reference instances present in the system.

Input Parameters:

hnd: Board handle

Output Parameters:

nInstances: pointer to the number of instances

Returns:

(TSYNC_SUCCESS) Success

4.2.16 Hardware (HW) Calls

HW calls provide access to the direct hardware accessible control/status of the timing subsystem (time stamping and time reads, controlling interrupts, match time, etc).

Note: Where <en >
0=Not enabled
1=Enabled

4.2.16.1 TSYNC_HW_getTime

```
TSYNC_ERROR TSYNC_HW_getTime(
    TSYNC_BoardHandle  handle,
    TSYNC_HWTimeObj   *pObj);
```

Description:
Get the current system time from the hardware.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.2 TSYNC_HW_getTimeSec

```
TSYNC_ERROR TSYNC_HW_getTimeSec(
    TSYNC_BoardHandle      handle,
    TSYNC_HWTimeSecondObj *pObj);
```

Description:
Get the current system time from the hardware in seconds format.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.3 TSYNC_HW_getTsEnable

```
TSYNC_ERROR TSYNC_HW_getTsEnable(
    TSYNC_BoardHandle  handle,
    int               *bEnable);
```

Description:

Get the current enable/disable state of timestamps.

Input Parameters:

hnd: Board handle

Output Parameters:

bEnable: The enable result

Returns:

(TSYNC_SUCCESS) Success

4.2.16.4 TSYNC_HW_setTsEnable

```
TSYNC_ERROR TSYNC_HW_setTsEnable(
    TSYNC_BoardHandle handle,
    int               bEnable);
```

Description:

Set the current enable/disable state of timestamps.

Input Parameters:

hnd: Board handle

bEnable: The enable information

Returns:

(TSYNC_SUCCESS) Success

4.2.16.5 TSYNC_HW_setTsReq

```
TSYNC_ERROR TSYNC_HW_setTsReq(
    TSYNC_BoardHandle handle);
```

Description:

Manually generate a hardware timestamp.

Input Parameters:

hnd: Board handle

Returns:

(TSYNC_SUCCESS) Success

4.2.16.6 TSYNC_HW_setTsClear

```
TSYNC_ERROR TSYNC_HW_setTsClear(
    TSYNC_BoardHandle handle,
    TMSTMP_SRC        source);
```

Description:

Clear all collected timestamps from the specified source.

Input Parameters:

hnd: Board handle

source: Timestamp source

Returns:

(TSYNC_SUCCESS) Success

4.2.16.7 TSYNC_HW_getTsCount

```
TSYNC_ERROR TSYNC_HW_getTsCount(
    TSYNC_BoardHandle  handle,
    TMSTMP_SRC         source,
    unsigned int      *nCount);
```

Description:
Get the number of collected timestamps for the specified source.
Input Parameters:
hnd: Board handle
source: The timestamp source information
Output Parameters:
nCount: The count result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.8 TSYNC_HW_getTsData

```
TSYNC_ERROR TSYNC_HW_getTsData(
    TSYNC_BoardHandle    handle,
    TMSTMP_SRC           source,
    TSYNC_HWTimeDataObj *pObj);
```

Description:
Get all collected timestamps for the specified source.
Input Parameters:
hnd: Board handle
source: The timestamp source information
Output Parameters:
pObj: Pointer to the timestamp data result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.9 TSYNC_HW_getMatchTimeHi

```
TSYNC_ERROR TSYNC_HW_getMatchTimeHi(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);
```

Description:
Get the match time value when the specified general purpose output will transition to an active high state.
Input Parameters:
hnd: Board handle
index: GPO index information
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.10 TSYNC_HW_setMatchTimeHi

```
TSYNC_ERROR TSYNC_HW_setMatchTimeHi(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);
```

Description:
Set the match time value when the specified general purpose output will transition to an active high state.
Input Parameters:
hnd: Board handle
index: GPO index information
pObj: Pointer to the time information
Returns:
(TSYNC_SUCCESS) Success

4.2.16.11 TSYNC_HW_getMatchTimeLo

```
TSYNC_ERROR TSYNC_HW_getMatchTimeLo(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);
```

Description:
Get the match time value when the specified general purpose output will transition to an active low state.
Input Parameters:
hnd: Board handle
index: GPO index information
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.12 TSYNC_HW_setMatchTimeLo

```
TSYNC_ERROR TSYNC_HW_setMatchTimeLo(
    TSYNC_BoardHandle  handle,
    OD_PIN             index,
    TSYNC_TimeObj     *pObj);
```

Description:
Set the match time value when the specified general purpose output will transition to an active low state.
Input Parameters:
hnd: Board handle
index: GPO index information
pObj: Pointer to the time information
Returns:
(TSYNC_SUCCESS) Success

4.2.16.13 TSYNC_HW_getFpgaInfo

```
TSYNC_ERROR TSYNC_HW_getFpgaInfo(
    TSYNC_BoardHandle  handle,
    unsigned short    *id,
    unsigned short    *rev);
```

Description:
Get the the FPGA ID and version information.
Input Parameters:
hnd: Board handle
Output Parameters:

id: The ID result
rev: The version result

Returns:
(TSYNC_SUCCESS) Success

4.2.16.14 TSYNC_HW_getIntMask

```
TSYNC_ERROR TSYNC_HW_getIntMask(
    TSYNC_BoardHandle  handle,
    INT_TYPE           intType,
    unsigned int       index,
    int               *bEnable);
```

Description:
Get the hardware interrupt masking enabled state

Input Parameters:
hnd: Board handle
intType: the interrupt type information (GPI/GPO are indexed)
index: the index of the interrupt (0 for non-indexed interrupts)

Output Parameters:
bEnable: The enable result

Returns:
(TSYNC_SUCCESS) Success

4.2.16.15 TSYNC_HW_setIntMask

```
TSYNC_ERROR TSYNC_HW_setIntMask(
    TSYNC_BoardHandle handle,
    INT_TYPE          intType,
    unsigned int      index,
    int               bEnable);
```

Description:
Set the hardware interrupt masking enabled state

Input Parameters:
hnd: Board handle
intType: the interrupt type information (GPI/GPO are indexed)
index: the index of the interrupt (0 for non-indexed interrupts)
bEnable: The enable information

Returns:
(TSYNC_SUCCESS) Success

4.2.16.16 TSYNC_HW_getTsSingle

```
TSYNC_ERROR TSYNC_HW_getTsSingle(
    TSYNC_BoardHandle  handle,
```

```
    TMSTMP_SRC          source,
    TSYNC_HWTimeObj    *pObj);
```

Description:
Get a single collected timestamp for a given source
Input Parameters:
hnd: Board handle
source: The timestamp source information
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.17 TSYNC_HW_getIntCnt

```
TSYNC_ERROR TSYNC_HW_getIntCnt(
    TSYNC_BoardHandle  handle,
    INT_TYPE           intType,
    unsigned int       index,
    unsigned int       *nIntCount);
```

Description:
Get a interrupt count for a given interrupt source
Input Parameters:
hnd: Board handle
intType: the interrupt type information (GPI/GPO are indexed)
index: the index of the interrupt (0 for non-indexed interrupts)
Output Parameters:
`nIntCount`: Pointer to the interrupt count result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.18 TSYNC_HW_clrIntCnt

```
TSYNC_ERROR TSYNC_HW_clrIntCnt(
    TSYNC_BoardHandle  handle,
    INT_TYPE           intType,
    unsigned int       index);
```

Description:
Reset a interrupt count for a given interrupt source
Input Parameters:
hnd: Board handle
intType: the interrupt type information (GPI/GPO are indexed)
index: the index of the interrupt (0 for non-indexed interrupts)
Output Parameters:
None

Returns:
(TSYNC_SUCCESS) Success

4.2.16.19 TSYNC_HW_getIntTs

```
TSYNC_ERROR TSYNC_HW_getIntTs(
    TSYNC_BoardHandle     handle,
    INT_TYPE              intType,
    unsigned int          index,
    TSYNC_TimeSecondsObj  *pObj);
```

Description:
Get the interrupt timestamp
Input Parameters:
hnd: Board handle
intType: The interrupt type information
index: The interrupt index information
Output Parameters:
pObj: Pointer to the interrupt timestamp result
Returns:
(TSYNC_SUCCESS) Success

4.2.16.20 TSYNC_HW_getTemperature

```
TSYNC_ERROR TSYNC_HW_getTemperature(
    TSYNC_BoardHandle  handle,
    unsigned short     *pTemp);
```

Description:
Read the current board temperature in counts. To convert to degrees C:

freq = 1/(pTemp*.00000002) // converts counts to freq
temperature = (freq/4)-273.15 // converts freq to Temperature in deg C

Input Parameters:
hnd: Board handle
Output Parameters:
pTemp: Pointer to contents of temperature
Returns:
(TSYNC_SUCCESS) Success

4.2.17 Initializer Service (IN) Calls

IN calls perform initial configuration and setup of each software module.

4.2.17.1 TSYNC_IN_getStatus

```
TSYNC_ERROR TSYNC_IN_getStatus(
    TSYNC_BoardHandle       hnd,
    unsigned int            pageNum,
    TSYNC_InitStatusResult *pObj);
```

Description:
Get the board's initialization results.
Input Parameters:
hnd: Board handle
pageNum: Table entry index
Output Parameters:
pObj: Pointer to the initialization status result
Returns:
(TSYNC_SUCCESS) Success

4.2.18 IRIG Output Component (IP) Calls

IP calls configure the output IRIG data streams, including the IRIG format, contents of the control field section (if included in the IRIG signal), Signature Control (which determines when the IRIG modulation will be present), Offsets (to account for cable delays and other latencies), amplitude adjustment, etc.

Note: Where <ce>
0= BCD TOY, Control Functions, Binary Seconds (SBS)
1= BCD TOY, Control Functions
2= BCD TOY
3= BCD TOY, Binary Seconds (SBS)
4= BCD TOY/Year, Control Functions, Binary Seconds (SBS)
5= BCD TOY/Year, Binary Seconds (SBS)
6= BCD TOY/Year
7= BCD TOY/Year, Binary Seconds (SBS)
8= Unknown- no fields

Where <cf>
0= All bits of the Control Field section are ignored.
1= Control Field conforms to RCC 200-04
2= Control Field conforms to IEEEE C37.118-2005
3= Control Field conforms to Spectracom format (i.e. NetClock)
4= Control Field conforms to NASA formats

Where <mode>
0= IRIG DCLS only
1= IRIG AM only
2= IRIG Manchester coding
3= Unknown
4= Port supports both AM and DCLS

Where <mod>
0= No carrier
1= 100 Hz
2= 1 kHz
3= 10 kHz
4= 100 kHz
5= 1 MHz
6= Unknown frequency

4.2.18.1 TSYNC_IP_getSigCtrl

```
TSYNC_ERROR TSYNC_IP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    SIG_CTL           *sig);
```

Description:
Get the IRIG output's signature control state.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
sig: The signature control result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.2 TSYNC_IP_setSigCtrl

```
TSYNC_ERROR TSYNC_IP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    SIG_CTL           sig);
```

Description:
Set the IRIG output's signature control state.
Input Parameters:
hnd: Board handle
nInstance: The instance number
sig: The signature control information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.3 TSYNC_IP_getOffset

```
TSYNC_ERROR TSYNC_IP_getOffset(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    int               *nOffset);
```

Description:
Get the IRIG output's offset. Offset is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nOffset: The offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.4 TSYNC_IP_setOffset

```
TSYNC_ERROR TSYNC_IP_setOffset(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    int               nOffset);
```

Description:
Set the IRIG output's offset. Offset is in nanoseconds from -500 msec to +500 msec.
Input Parameters:
hnd: Board handle
nInstance: The instance number
nOffset: The offset information

Returns:
(TSYNC_SUCCESS) Success

4.2.18.5 TSYNC_IP_getLocal

```
TSYNC_ERROR TSYNC_IP_getLocal(
    TSYNC_BoardHandle    hnd,
    IO_PORT              nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Get the IRIG reference's local time zone and DST rule. Timezone and DST offsets are in seconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Local Clock result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.6 TSYNC_IP_setLocal

```
TSYNC_ERROR TSYNC_IP_setLocal(
    TSYNC_BoardHandle    hnd,
    IO_PORT              nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Set the IRIG reference's local time zone and DST rule. Timezone and DST offsets are in seconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Local Clock information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.7 TSYNC_IP_getFormat

```
TSYNC_ERROR TSYNC_IP_getFormat(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    IL_FMT            *format);
```

Description:
Get the IRIG format.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
format: The format result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.8 TSYNC_IP_setFormat

```
TSYNC_ERROR TSYNC_IP_setFormat(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    IL_FMT            format);
```

Description:
Set the IRIG format.
Input Parameters:
hnd: Board handle
nInstance: The instance number
format: The format information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.9 TSYNC_IP_getAmplitude

```
TSYNC_ERROR TSYNC_IP_getAmplitude(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    unsigned int      *amp);
```

Description:
Get the IRIG amplitude. Amplitude is in range of 3 - 255.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
amp: The amplitude result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.10 TSYNC_IP_setAmplitude

```
TSYNC_ERROR TSYNC_IP_setAmplitude(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    unsigned int      amp);
```

Description:
Set the IRIG amplitude. Amplitude is in range of 3 - 255.
Input Parameters:
hnd: Board handle
nInstance: The instance number
amp: The amplitude information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.11 TSYNC_IP_getMod

```
TSYNC_ERROR TSYNC_IP_getMod(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    IL_MOD            *mod);
```

Description:
Get the IRIG modulation.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
mod: The modulation result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.12 TSYNC_IP_getFreq

```
TSYNC_ERROR TSYNC_IP_getFreq(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    IL_FRQ            *freq);
```

Description:
Get the IRIG frequency.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
freq: The frequency result

Returns:
(TSYNC_SUCCESS) Success

4.2.18.13 TSYNC_IP_setFreq

```
TSYNC_ERROR TSYNC_IP_setFreq(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    IL_FRQ            freq);
```

Description:
Set the IRIG frequency.
Input Parameters:
hnd: Board handle
nInstance: The instance number
freq: The frequency information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.14 TSYNC_IP_getCodedExpr

```
TSYNC_ERROR TSYNC_IP_getCodedExpr(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    IL_CE             *ce);
```

Description:
Get the IRIG Coded Expression.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
ce: The Coded Expression result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.15 TSYNC_IP_setCodedExpr

```
TSYNC_ERROR TSYNC_IP_setCodedExpr(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    IL_CE             ce);
```

Description:
Set the IRIG Coded Expression.

Input Parameters:
hnd: Board handle
nInstance: The instance number
ce: The Coded Expression information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.16 TSYNC_IP_getCtrlField

```
TSYNC_ERROR TSYNC_IP_getCtrlField(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    IL_CF             *cf);
```

Description:
Get the IRIG Control Field.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
cf: The Control Field result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.17 TSYNC_IP_setCtrlField

```
TSYNC_ERROR TSYNC_IP_setCtrlField(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    IL_CF             cf);
```

Description:
Set the IRIG Control Field.
Input Parameters:
hnd: Board handle
nInstance: The instance number
cf: The Control Field information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.18 TSYNC_IP_getMessage

```
TSYNC_ERROR TSYNC_IP_getMessage(
    TSYNC_BoardHandle     hnd,
    IO_PORT               nInstance,
    TSYNC_IRIGMessageObj *pObj);
```

Description:
Get the latest IRIG output message.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the message information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.19 TSYNC_IP_setMessage

```
TSYNC_ERROR TSYNC_IP_setMessage(
    TSYNC_BoardHandle     hnd,
    IO_PORT               nInstance,
    TSYNC_IRIGMessageObj *pObj);
```

Description:
Set the latest IRIG output message.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the message information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.20 TSYNC_IP_getCfData

```
TSYNC_ERROR TSYNC_IP_getCfData(
    TSYNC_BoardHandle    hnd,
    IO_PORT              nInstance,
    TSYNC_IRIGCfDataObj *pObj);
```

Description:
Get the latest IRIG control field data.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the control field information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.21 TSYNC_IP_setCfData

```
TSYNC_ERROR TSYNC_IP_setCfData(
    TSYNC_BoardHandle    hnd,
    IO_PORT              nInstance,
    TSYNC_IRIGCfDataObj *pObj);
```

Description:
Set the IRIG control field data manually.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the control field information
Returns:
(TSYNC_SUCCESS) Success

4.2.18.22 TSYNC_IP_getNumInst

```
TSYNC_ERROR TSYNC_IP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of IRIG output instances present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.23 TSYNC_IP_getTimeScale

```
TSYNC_ERROR TSYNC_IP_getTimeScale(
    TSYNC_BoardHandle   hnd,
    IO_PORT             nInstance,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Get the IRIG output's time scale.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the time scale result
Returns:
(TSYNC_SUCCESS) Success

4.2.18.24 TSYNC_IP_setTimeScale

```
TSYNC_ERROR TSYNC_IP_setTimeScale(
    TSYNC_BoardHandle   hnd,
    IO_PORT             nInstance,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Set the IRIG output's time scale.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the time scale information
Returns:
(TSYNC_SUCCESS) Success

4.2.19 IRIG Reference Component (IR) Calls

IR calls control and process decoded IRIG input streams to determine 1PPS and time validity. These settings include configuring the IRIG format, contents of the control field section (if included in the IRIG signal), Signature Control (which determines when the IRIG modulation will be present), Offsets (to account for cable delays and other latencies), etc.

Note: Where <frq>

0= BCD TOY, Control Functions, Binary Seconds (SBS)
1= BCD TOY, Control Functions
2= BCD TOY
3= BCD TOY, Binary Seconds (SBS)
4= BCD TOY/Year, Control Functions, Binary Seconds (SBS)
5= BCD TOY/Year, Binary Seconds (SBS)
6= BCD TOY/Year
7= BCD TOY/Year, Binary Seconds (SBS)
8= Unknown- no fields

Where <cf>

0= All bits of the Control Field section are ignored.
1= Control Field conforms to RCC 200-04
2= Control Field conforms to IEEEE C37.118-2005
3= Control Field conforms to Spectracom format (i.e. NetClock)
4= Control Field conforms to NASA formats

Where <mod>

0= No carrier
1= 100 Hz
2= 1 kHz
3= 10 kHz
4= 100 kHz

5= 1 MHz
6= Unknown frequency

4.2.19.1 TSYNC_IR_getOffset

```
TSYNC_ERROR TSYNC_IR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);
```

Description:
Get the IRIG reference's 1PPS input offset. Offset is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nOffset: The offset information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.2 TSYNC_IR_setOffset

```
TSYNC_ERROR TSYNC_IR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);
```

Description:
Set the IRIG reference's 1PPS input offset. Offset is in nanoseconds from -500 msec to +500 msec.
Input Parameters:
hnd: Board handle
nInstance: The instance number
nOffset: The offset information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.3 TSYNC_IR_getValidity

```
TSYNC_ERROR TSYNC_IR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);
```

Description:

Get the IRIG validity structure.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

bTimeValid: The time reference result
bPpsValid: The pps reference result

Returns:

(TSYNC_SUCCESS) Success

4.2.19.4 TSYNC_IR_getMode

```
TSYNC_ERROR TSYNC_IR_getMode(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_MODE           *mode);
```

Description:

Get the IRIG mode.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

mode: The receiver mode result

Returns:

(TSYNC_SUCCESS) Success

4.2.19.5 TSYNC_IR_setMode

```
TSYNC_ERROR TSYNC_IR_setMode(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_MODE           mode);
```

Description:

Set the IRIG mode.

Input Parameters:

hnd: Board handle
nInstance: The instance number
mode: The receiver mode information

Returns:

(TSYNC_SUCCESS) Success

4.2.19.6 TSYNC_IR_getFormat

```
TSYNC_ERROR TSYNC_IR_getFormat(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FMT            *format);
```

Description:
Get the IRIG format.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
format: The format result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.7 TSYNC_IR_setFormat

```
TSYNC_ERROR TSYNC_IR_setFormat(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FMT            format);
```

Description:
Set the IRIG format. Settable only when in manual mode.
Input Parameters:
hnd: Board handle
nInstance: The instance number
format: The format information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.8 TSYNC_IR_getMod

```
TSYNC_ERROR TSYNC_IR_getMod(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_MOD            *mod);
```

Description:
Get the IRIG modulation.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
mod: The modulation result

Returns:
(TSYNC_SUCCESS) Success

4.2.19.9 TSYNC_IR_getFreq

```
TSYNC_ERROR TSYNC_IR_getFreq(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_FRQ            *freq);
```

Description:
Get the IRIG carrier frequency.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
freq: The frequency result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.10 TSYNC_IR_setFreq

```
TSYNC_ERROR TSYNC_IR_setFreq(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_FRQ            freq);
```

Description:
Set the IRIG carrier frequency. Settable only when in manual mode.
Input Parameters:
hnd: Board handle
nInstance: The instance number
freq: The frequency information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.11 TSYNC_IR_getCodedExpr

```
TSYNC_ERROR TSYNC_IR_getCodedExpr(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CE             *ce);
```

Description:
Get the IRIG Coded Expression.

Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
ce: The Coded Expression result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.12 TSYNC_IR_setCodedExpr

```
TSYNC_ERROR TSYNC_IR_setCodedExpr(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    IL_CE             ce);
```

Description:
Set the IRIG Coded Expression.
Input Parameters:
hnd: Board handle
nInstance: The instance number
ce: The Coded Expression information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.13 TSYNC_IR_getCtrlField

```
TSYNC_ERROR TSYNC_IR_getCtrlField(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    IL_CF             *cf);
```

Description:
Get the IRIG Control Field.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
cf: The Control Field result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.14 TSYNC_IR_setCtrlField

```
TSYNC_ERROR TSYNC_IR_setCtrlField(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
```

```
    IL_CF              cf);
```

Description:
Set the IRIG Control Field.
Input Parameters:
hnd: Board handle
nInstance: The instance number
cf: The Control Field information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.15 TSYNC_IR_getMessage

```
TSYNC_ERROR TSYNC_IR_getMessage(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);
```

Description:
Get the latest IRIG input message.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the message result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.16 TSYNC_IR_setMessage

```
TSYNC_ERROR TSYNC_IR_setMessage(
    TSYNC_BoardHandle     hnd,
    unsigned int          nInstance,
    TSYNC_IRIGMessageObj *pObj);
```

Description:
Set the latest IRIG input message.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the message information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.17 TSYNC_IR_getNumInst

```
TSYNC_ERROR TSYNC_IR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of IRIG references instances present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.18 TSYNC_IR_getCfData

```
TSYNC_ERROR TSYNC_IR_getCfData(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_IRIGCfDataObj *pObj);
```

Description:
Get the latest IRIG Control Field data received.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the control field information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.19 TSYNC_IR_getLocal

```
TSYNC_ERROR TSYNC_IR_getLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Get the IRIG regerence's local time zone and DST rule. Timezone and DST offsets are in seconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Local Clock result

Returns:
(TSYNC_SUCCESS) Success

4.2.19.20 TSYNC_IR_setLocal

```
TSYNC_ERROR TSYNC_IR_setLocal(
    TSYNC_BoardHandle    hnd,
    unsigned int         nInstance,
    TSYNC_LocalClockObj *pObj);
```

Description:
Set the IRIG regerence's local time zone and DST rule. Timezone and DST offsets are in seconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Local Clock information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.21 TSYNC_IR_getTimeScale

```
TSYNC_ERROR TSYNC_IR_getTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Get the IRIG regerence's time scale.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the time scale result
Returns:
(TSYNC_SUCCESS) Success

4.2.19.22 TSYNC_IR_setTimeScale

```
TSYNC_ERROR TSYNC_IR_setTimeScale(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_TimeScaleObj *pObj);
```

Description:
Set the IRIG regerence's time scale

Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the time scale information
Returns:
(TSYNC_SUCCESS) Success

4.2.19.23 TSYNC_IR_getRefId

```
TSYNC_ERROR TSYNC_IR_getRefId(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_RefIdObj     *pObj);
```

Description:
Get reference identifier for an IRIG reference instance.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Reference ID
Returns:
(TSYNC_SUCCESS) Success

4.2.20 Log Service (LS) Calls

LS calls provide a queue for errors and maintain system alarms.

4.2.20.1 TSYNC_LS_getErrorLog

```
TSYNC_ERROR TSYNC_LS_getErrorLog(
    TSYNC_BoardHandle  hnd,
    TSYNC_ErrorLogObj *pObj);
```

Description:
Get the error log.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the error log result
Returns:
(TSYNC_SUCCESS) Success

Alarms:
Note: Where <alarm type>
0= Not in Sync (or Holdover mode)
1= In Holdover mode
2= Frequency error

3= software alarm
4= software alarm
5= 1PPS not in sync
6= Reference change

4.2.20.2 TSYNC_LS_getAlarm

```
TSYNC_ERROR TSYNC_LS_getAlarm(
    TSYNC_BoardHandle  hnd,
    TSYNC_AlarmObj    *pObj,
    TSYNC_FlagObj     *pObj2);
```

Description:
Get the alarm state for the specified alarm.
Input Parameters:
hnd: Board handle
pObj: Pointer to the alarm index information
Output Parameters:
pObj2: Pointer to the alarm flag result
Returns:
(TSYNC_SUCCESS) Success

4.2.20.3 TSYNC_LS_setAlarm

```
TSYNC_ERROR TSYNC_LS_setAlarm(
    TSYNC_BoardHandle  hnd,
    TSYNC_AlarmObj    *pObj,
    TSYNC_FlagObj     *pObj2);
```

Description:
Clear the specified alarm. (Set 0 to clear)
Input Parameters:
hnd: Board handle
pObj: Pointer to the alarm index information
pObj2: Pointer to the alarm flag result
Returns:
(TSYNC_SUCCESS) Success

4.2.20.4 TSYNC_LS_getVersion

```
TSYNC_ERROR TSYNC_LS_getVersion(
    TSYNC_BoardHandle         hnd,
    TSYNC_FirmwareVersionObj *pObj);
```

Description:
Get the firmware version string.
Input Parameters:

hnd: Board handle

Output Parameters:

pObj: Pointer to the firmware version result

Returns:

(TSYNC_SUCCESS) Success

4.2.21 PPS Output Component (PP) Calls

PP calls control a 1Hz output.

Note: Where <edge>
0=Falling edge
1=Rising edge
2=both (uncommon)

4.2.21.1 TSYNC_PP_getSigCtrl

```
TSYNC_ERROR TSYNC_PP_getSigCtrl(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    SIG_CTL           *sig);
```

Description:

Get the PPS output's signature control state.

Input Parameters:

hnd: Board handle
nInstance: The instance number

Output Parameters:

sig: The signature control result

Returns:

(TSYNC_SUCCESS) Success

4.2.21.2 TSYNC_PP_setSigCtrl

```
TSYNC_ERROR TSYNC_PP_setSigCtrl(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    SIG_CTL           sig);
```

Description:

Set the PPS output's signature control state.

Input Parameters:

hnd: Board handle
nInstance: The instance number
sig: The signature control information

Returns:

(TSYNC_SUCCESS) Success

4.2.21.3 TSYNC_PP_getFreq

```
TSYNC_ERROR TSYNC_PP_getFreq(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    unsigned int      *freq);
```

Description:
Get the PPS output's frequency. Frequency is 1 Hertz
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
freq: The frequency result
Returns:
(TSYNC_SUCCESS) Success

4.2.21.4 TSYNC_PP_getOffset

```
TSYNC_ERROR TSYNC_PP_getOffset(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    int               *nOffset);
```

Description:
Get the PPS output's offset. Offset is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nOffset: The offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.21.5 TSYNC_PP_setOffset

```
TSYNC_ERROR TSYNC_PP_setOffset(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    int               nOffset);
```

Description:
Set the PPS output's offset. Offset is in nanoseconds from -500 msec to +500 msec.
Input Parameters:
hnd: Board handle

nInstance: The instance number
nOffset: The offset information

Returns:
(TSYNC_SUCCESS) Success

4.2.21.6 TSYNC_PP_getEdge

```
TSYNC_ERROR TSYNC_PP_getEdge(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    EDGE              *edge);
```

Description:
Get the PPS output's edge.

Input Parameters:
hnd: Board handle
nInstance: The instance number

Output Parameters:
edge: The edge result

Returns:
(TSYNC_SUCCESS) Success

4.2.21.7 TSYNC_PP_setEdge

```
TSYNC_ERROR TSYNC_PP_setEdge(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    EDGE              edge);
```

Description:
Set the PPS output's edge.

Input Parameters:
hnd: Board handle
nInstance: The instance number
edge: The edge information

Returns:
(TSYNC_SUCCESS) Success

4.2.21.8 TSYNC_PP_getPulseWidth

```
TSYNC_ERROR TSYNC_PP_getPulseWidth(
    TSYNC_BoardHandle  hnd,
    IO_PORT            nInstance,
    unsigned int      *pw);
```

Description:
Get the PPS output's pulse width. Pulse width is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pw: The pulse width result
Returns:
(TSYNC_SUCCESS) Success

4.2.21.9 TSYNC_PP_setPulseWidth

```
TSYNC_ERROR TSYNC_PP_setPulseWidth(
    TSYNC_BoardHandle hnd,
    IO_PORT           nInstance,
    unsigned int      pw);
```

Description:
Set the PPS output's pulse width. Pulse width is in nanoseconds from 10 nsec to 999,999,990 nsec.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pw: The pulse width information
Returns:
(TSYNC_SUCCESS) Success

4.2.21.10 TSYNC_PP_getNumInst

```
TSYNC_ERROR TSYNC_PP_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of PPS output instances present in the system.
Input Parameters:
hnd: Board handle
Output Parameters:
nInstances: The number of instances result
Returns:
(TSYNC_SUCCESS) Success

4.2.22 PPS Reference Component (PR) Calls

PR calls monitor the 1PPS input reference.

Note: Where <edge>

0= Falling edge
1= Rising edge
2= both (uncommon)

Where <valid>
0= Not valid
1= Valid

Where <pin> can be a value of “0” through “3”

Where <index> should always be a value of “0” with only one board installed.

4.2.22.1 TSYNC_PR_getOffset

```
TSYNC_ERROR TSYNC_PR_getOffset(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *nOffset);
```

Description:
Get the 1PPS reference’s input offset. Offset is in nanoseconds.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
nOffset: The offset result
Returns:
(TSYNC_SUCCESS) Success

4.2.22.2 TSYNC_PR_setOffset

```
TSYNC_ERROR TSYNC_PR_setOffset(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    int               nOffset);
```

Description:
Set the 1PPS reference’s input offset. Offset is in nanoseconds from -500 msec to +500 msec.
Input Parameters:
hnd: Board handle
nInstance: The instance number
nOffset: The offset
Returns:
(TSYNC_SUCCESS) Success

4.2.22.3 TSYNC_PR_getEdge

```
TSYNC_ERROR TSYNC_PR_getEdge(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    EDGE              *edge);
```

Description:
Get the 1PPS reference's active edge setting.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
edge: The edge result
Returns:
(TSYNC_SUCCESS) Success

4.2.22.4 TSYNC_PR_setEdge

```
TSYNC_ERROR TSYNC_PR_setEdge(
    TSYNC_BoardHandle hnd,
    unsigned int      nInstance,
    EDGE              edge);
```

Description:
Set the 1PPS reference's active edge setting.
Input Parameters:
hnd: Board handle
nInstance: The instance number
edge: The edge information
Returns:
(TSYNC_SUCCESS) Success

4.2.22.5 TSYNC_PR_getValidity

```
TSYNC_ERROR TSYNC_PR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);
```

Description:
Get the 1PPS validity structure.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:

bTimeValid: The time reference result
bPpsValid: The pps reference result

Returns:
(TSYNC_SUCCESS) Success

4.2.22.6 TSYNC_PR_getNumInst

```
TSYNC_ERROR TSYNC_PR_getNumInst(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nInstances);
```

Description:
Get number of PPS reference instances present in the system.

Input Parameters:
hnd: Board handle

Output Parameters:
nInstances: The number of instances result

Returns:
(TSYNC_SUCCESS) Success

4.2.22.7 TSYNC_PR_getRefId

```
TSYNC_ERROR TSYNC_PR_getRefId(
    TSYNC_BoardHandle   hnd,
    unsigned int        nInstance,
    TSYNC_RefIdObj     *pObj);
```

Description:
Get reference identifier for a PPS reference instance.

Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Reference ID

Returns:
(TSYNC_SUCCESS) Success

4.2.23 PTP Reference Component (PTR) Calls

PTR calls control and process decoded PTP network packets (either as an input reference or a time output).

4.2.23.1 TSYNC_PTR_getModuleInfo

```
TSYNC_ERROR TSYNC_PTR_getModuleInfo(
   TSYNC_BoardHandle       hnd,
```

```
   unsigned int             nInstance,
   TSYNC_PTPModuleInfoObj *pObj);
```

Description:
Gets the PTP module's version and build date.

Input Parameters:
hnd: Board handle
nInstance: The instance number

Output Parameters:
pObj: Pointer to the Module Information object

Returns:
(TSYNC_SUCCESS) Success

4.2.23.2 TSYNC_PTR_getEthernetItf

```
TSYNC_ERROR TSYNC_PTR_getEthernetItf(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPEthernetItfObj *pObj);
```

Description:
Gets Ethernet settings for the PTP Module.

Input Parameters:
hnd: Board handle
nInstance: The instance number

Output Parameters:
pObj: Pointer to the Ethernet Settings object

Returns:
(TSYNC_SUCCESS) Success

4.2.23.3 TSYNC_PTR_setEthernetItf

```
TSYNC_ERROR TSYNC_PTR_setEthernetItf(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPEthernetItfObj *pObj);
```

Description:
Sets Ethernet settings for the PTP Module.

Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Ethernet Settings object

Returns:
(TSYNC_SUCCESS) Success

NOTE: When enabling DHCP, the rest of the PTPEthernetItfObj structure still needs to be populated. This data will be retailed by the TSync-cPCI, but it will not be used as long as DHCP is enabled.

After changing Ethernet Settings, please reset the module.

4.2.23.4 TSYNC_PTR_getUnitSettings

```
TSYNC_ERROR TSYNC_PTR_getUnitSettings(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPUnitSettingsObj *pObj);
```

Description:
Gets general PTP settings for the PTP Module.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Unit Settings object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.5 TSYNC_PTR_setUnitSettings

```
TSYNC_ERROR TSYNC_PTR_setUnitSettings(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPUnitSettingsObj *pObj);
```

Description:
Sets general PTP settings for the PTP Module.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Unit Settings object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.6 TSYNC_PTR_getPortState

```
TSYNC_ERROR TSYNC_PTR_getPortState(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPPortStateObj  *pObj);
```

Description:
Gets the current state of the PTP port.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Port Settings object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.7 TSYNC_PTR_getPortSettings

```
TSYNC_ERROR TSYNC_PTR_getPortSettings(
   TSYNC_BoardHandle         hnd,
   unsigned int              nInstance,
   TSYNC_PTPPortSettingsObj *pObj);
```

Description:
Gets configuration information for the PTP port.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Port Settings object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.8 TSYNC_PTR_setPortSettings

```
TSYNC_ERROR TSYNC_PTR_setPortSettings(
   TSYNC_BoardHandle         hnd,
   unsigned int              nInstance,
   TSYNC_PTPPortSettingsObj *pObj);
```

Description:
Sets configuration information for the PTP port.
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the Port Settings object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.9 TSYNC_PTR_getClkQuality

```
TSYNC_ERROR TSYNC_PTR_getClkQuality(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPClkQualityObj *pObj);
```

Description:
Gets the module's reported clock quality information.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Clock Quality object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.10 TSYNC_PTR_getTimeProperties

```
TSYNC_ERROR TSYNC_PTR_getTimeProperties(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPClkQualityObj *pObj);
```

Description:
Gets the module's reported time properties information.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Time Properties object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.11 TSYNC_PTR_getParentProperties

```
TSYNC_ERROR TSYNC_PTR_getParentProperties(
   TSYNC_BoardHandle       hnd,
   unsigned int            nInstance,
   TSYNC_PTPParentPropObj *pObj);
```

Description:
Gets the module's parent properties dataset.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Parent Properties object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.12 TSYNC_PTR_getGrandmasterProperties

```
TSYNC_ERROR TSYNC_PTR_getGrandmasterProperties(
   TSYNC_BoardHandle           hnd,
   unsigned int                nInstance,
   TSYNC_PTPGrandmasterPropObj *pObj);
```

Description:
Gets the module's Grandmaster Properties dataset.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the Grandmaster Properties object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.13 TSYNC_PTR_saveSettingsToROM

```
TSYNC_ERROR TSYNC_PTR_saveSettingsToROM(
   TSYNC_BoardHandle            hnd,
   unsigned int                 nInstance
```

Description:
Saves any settings that have been changed in the PTP module to the module's ROM.

Input Parameters:
hnd: Board handle
nInstance: The instance number

Returns:
(TSYNC_SUCCESS) Success

4.2.23.14 TSYNC_PTR_resetModule

```
TSYNC_ERROR TSYNC_PTR_resetModule(
   TSYNC_BoardHandle            hnd,
   unsigned int                 nInstance,
   PTL_RESET                    resetType);
```

Description:
Resets the PTP Module.

Input Parameters:
hnd: Board handle
nInstance: The instance number

Output Parameters:
resetType: the type of reset requested

Returns:
(TSYNC_SUCCESS) Success

0 = Cold reset
1 = Warm reset
2 = Restore Factory Defaults and Reset

4.2.23.15 TSYNC_PTR_getNumInst

```
TSYNC_ERROR TSYNC_PTR_getNumInst(
   TSYNC_BoardHandle            hnd,
   unsigned int                *nInstances);
```

Description:
Gets the number pf PTP instances in the system.

Input Parameters:
hnd: Board handle

Output Parameters:
nInstances: pointer to number of PTP instances

Returns:
(TSYNC_SUCCESS) Success

4.2.23.16 TSYNC_PTR_reinitModule

```
TSYNC_ERROR TSYNC_PTR_resetModule(
   TSYNC_BoardHandle            hnd,
   unsigned int                 nInstance);
```

Description:
Reinitializes the PTP stack in the PTP module.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Returns:
(TSYNC_SUCCESS) Success

4.2.23.17 TSYNC_PTR_getValidity

```
TSYNC_ERROR TSYNC_PTR_getValidity(
    TSYNC_BoardHandle  hnd,
    unsigned int       nInstance,
    int               *bTimeValid,
    int               *bPpsValid);
```

Description:
Get the PTP validity structure (when used as a PTP Slave).
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
bTimeValid: The time reference result
bPpsValid: The pps reference result
Returns:
(TSYNC_SUCCESS) Success

4.2.23.18 TSYNC_PTR_getMode

```
TSYNC_ERROR TSYNC_PTR_getMode(
   TSYNC_BoardHandle         hnd,
   unsigned int              nInstance,
   int                      *bMode);
```

Description:
Gets the PTP module's current operational mode
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
bMode: pointer to Mode object
Returns:
(TSYNC_SUCCESS) Success

0 = PTP Slave Mode
1 = PTP Master Mode

4.2.23.19 TSYNC_PTR_setMode

```
TSYNC_ERROR TSYNC_PTR_setMode(
   TSYNC_BoardHandle          hnd,
   unsigned int               nInstance,
   int                       *bMode);
```

Description:
Sets the PTP module's current operational mode
Input Parameters:
hnd: Board handle
nInstance: The instance number
bMode: pointer to Mode object
Returns:
(TSYNC_SUCCESS) Success

0 = PTP Slave Mode
1 = PTP Master Mode

After changing the operational mode, please reset the module.

4.2.23.20 TSYNC_PTR_getMacAddr

```
TSYNC_ERROR TSYNC_PTR_getMacAddr(
   TSYNC_BoardHandle          hnd,
   unsigned int               nInstance,
   TSYNC_PTPMacAddrObj       *pObj);
```

Description:
Gets the PTP module's current MAC Address
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
bMode: pointer to MAC Address object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.21 TSYNC_PTR_getModuleStatus

```
TSYNC_ERROR TSYNC_PTR_getModuleStatus(
   TSYNC_BoardHandle          hnd,
   unsigned int               nInstance,
```

```
   int                          *bModuleStatus);
```

Description:
Gets the PTP module's status information.
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
bModuleStatus: ppinter to current Module Status
Returns:
(TSYNC_SUCCESS) Success

Module Status reports the cause of the last reset operation.

4.2.23.22 TSYNC_PTR_getUserDesc

```
TSYNC_ERROR TSYNC_PTR_getUserDesc(
   TSYNC_BoardHandle          hnd,
   unsigned int               nInstance,
   TSYNC_PTPUserDescObj      *pObj);
```

Description:
Gets the PTP User Description strings
Input Parameters:
hnd: Board handle
nInstance: The instance number
Output Parameters:
pObj: Pointer to the User Description object
Returns:
(TSYNC_SUCCESS) Success

4.2.23.23 TSYNC_PTR_setUserDesc

```
TSYNC_ERROR TSYNC_PTR_setUserDesc(
   TSYNC_BoardHandle          hnd,
   unsigned int               nInstance,
   TSYNC_PTPUserDescObj      *pObj);
```

Description:
Gets the PTP User Description strings
Input Parameters:
hnd: Board handle
nInstance: The instance number
pObj: Pointer to the User Description object
Returns:
(TSYNC_SUCCESS) Success

4.2.24 Reference Monitor Service (RS) Calls

RS calls determine the best available Time and 1PPS input references and maintain the reference priority table.

Note: Where <Table type>
0= Factory table
1= User table
2= Working table

Where <enable>
0= Input References not enabled
1= Input References enabled

Where <index> can be a value of “0” through “7” (Where the index is the row of the table).

Where <priority> can be a value of “1” through “8” (With 1 being the highest priority reference and 8 being the lowest priority reference).

4.2.24.1 TSYNC_RS_getTable

```
TSYNC_ERROR TSYNC_RS_getTable(
    TSYNC_BoardHandle        hnd,
    TSYNC_TableTypeObj      *pObj,
    TSYNC_ReferenceTableObj *pObj2);
```

Description:
Get specified reference priority table.
Input Parameters:
hnd: Board handle
pObj: Pointer to the table type information
Output Parameters:
pObj2: Pointer to the reference result
Returns:
(TSYNC_SUCCESS) Success

4.2.24.2 TSYNC_RS_getBestRef

```
TSYNC_ERROR TSYNC_RS_getBestRef(
    TSYNC_BoardHandle    hnd,
    TSYNC_TableEntryObj *pObj);
```

Description:
Get current best working priority table entry.
Input Parameters:
hnd: Board handle

Output Parameters:
pObj: Pointer to the table entry result
Returns:
(TSYNC_SUCCESS) Success

4.2.24.3 TSYNC_RS_getEntry

```
TSYNC_ERROR TSYNC_RS_getEntry(
    TSYNC_BoardHandle    hnd,
    int                  index,
    TSYNC_TableEntryObj *pObj);
```

Description:
Get working priority table entry by index.
Input Parameters:
hnd: Board handle
index: Table entry index
Output Parameters:
pObj: Pointer to the table entry result
Returns:
(TSYNC_SUCCESS) Success

4.2.24.4 TSYNC_RS_addEntry

```
TSYNC_ERROR TSYNC_RS_addEntry(
    TSYNC_BoardHandle    hnd,
    TSYNC_TableEntryObj *pObj);
```

Description:
Add an entry to the working priority table.
Input Parameters:
hnd: Board handle
pObj: Pointer to the table entry information
Returns:
(TSYNC_SUCCESS) Success

4.2.24.5 TSYNC_RS_setFactDef

```
TSYNC_ERROR TSYNC_RS_setFactDef(
    TSYNC_BoardHandle hnd);
```

Description:
Reset the working priority table to the factory priority table.
Input Parameters:
hnd: Board handle
Returns:

(TSYNC_SUCCESS) Success

4.2.24.6 TSYNC_RS_setUserDef

```
TSYNC_ERROR TSYNC_RS_setUserDef(
    TSYNC_BoardHandle hnd);
```

Description:
Reset the reference table to the user default settings, if saved user priority table exists.
Input Parameters:
hnd: Board handle
Returns:
(TSYNC_SUCCESS) Success

4.2.24.7 TSYNC_RS_saveUserDef

```
TSYNC_ERROR TSYNC_RS_saveUserDef(
    TSYNC_BoardHandle hnd);
```

Description:
Save the working priority table to the user priority table.
Input Parameters:
hnd: Board handle
Returns:
(TSYNC_SUCCESS) Success

4.2.24.8 TSYNC_RS_deleteEntry

```
TSYNC_ERROR TSYNC_RS_deleteEntry(
    TSYNC_BoardHandle hnd,
    unsigned int      index);
```

Description:
Delete a working priority table entry by index.
Input Parameters:
hnd: Board handle
index: Index of table entry to delete
Returns:
(TSYNC_SUCCESS) Success

4.2.24.9 TSYNC_RS_getPriority

```
TSYNC_ERROR TSYNC_RS_getPriority(
    TSYNC_BoardHandle  hnd,
```

```
    unsigned int        index,
    unsigned int       *priority);
```

Description:
Get specified working priority table entry's priority.
Input Parameters:
hnd: Board handle
index: Table entry index
Output Parameters:
priority: Pointer to the priority result
Returns:
(TSYNC_SUCCESS) Success

4.2.24.10 TSYNC_RS_setPriority

```
TSYNC_ERROR TSYNC_RS_setPriority(
    TSYNC_BoardHandle hnd,
    unsigned int      index,
    unsigned int      priority);
```

Description:
Set specified working priority table entry's priority.
Input Parameters:
hnd: Board handle
index: Table entry index
priority: Pointer to the priority information
Returns:
(TSYNC_SUCCESS) Success

4.2.24.11 TSYNC_RS_getEnable

```
TSYNC_ERROR TSYNC_RS_getEnable(
    TSYNC_BoardHandle  hnd,
    unsigned int        index,
    unsigned int       *enabled);
```

Description:
Get specified working priority table entry's enable state.
Input Parameters:
hnd: Board handle
index: Table entry index
Output Parameters:
enabled: Pointer to the enabled result
Returns:
(TSYNC_SUCCESS) Success

4.2.24.12 TSYNC_RS_setEnable

```
TSYNC_ERROR TSYNC_RS_setEnable(
    TSYNC_BoardHandle hnd,
    unsigned int      index,
    unsigned int      enabled);
```

Description:
Set specified working priority table entry's enable state.
Input Parameters:
hnd: Board handle
index: Table entry index
enabled: Pointer to the enabled information
Returns:
(TSYNC_SUCCESS) Success

4.2.24.13 TSYNC_RS_getStateTable

```
TSYNC_ERROR TSYNC_RS_getStateTable(
    TSYNC_BoardHandle             hnd,
    TSYNC_ReferenceStateTableObj *pObj);
```

Description:
Get the reference validity state table.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the reference state table result
Returns:
(TSYNC_SUCCESS) Success

4.2.25 Supervisor Service (SS) Calls

SS calls maintain the time source, 1PPS source, Sync and Holdover states of the system.

TFOM (Time Figure of Merit) is a measure of how accurate the system's PPS is relative to the reference 1PPS. Lower numbers are better. (The TSync is not capable of a TFOM value lower than 3.)

4.2.25.1 TSYNC_SS_reset

```
TSYNC_ERROR TSYNC_SS_reset(
    TSYNC_BoardHandle  hnd,
    TSYNC_ResetObj    *pObj);
```

Description:
Reset the TSYNC board.
Input Parameters:
hnd: Board handle
pObj: Pointer to the reset type information
Returns:
(TSYNC_SUCCESS) Success
Note: Only use “0” for the <reset type>.

4.2.25.2 TSYNC_SS_getRef

```
TSYNC_ERROR TSYNC_SS_getRef(
    TSYNC_BoardHandle   hnd,
    TSYNC_ReferenceObj *pObj);
```

Description:
Get currently selected time and 1PPS reference.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the reference result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.3 TSYNC_SS_getMaxTfom

```
TSYNC_ERROR TSYNC_SS_getMaxTfom(
    TSYNC_BoardHandle  hnd,
    TFOM              *tfom);
```

Description:
Get the maximum TFOM threshold to stay in sync.
Input Parameters:
hnd: Board handle
Output Parameters:
tfom: Pointer to the maximum TFOM result
Returns:
(TSYNC_SUCCESS) Success

Returns the user-configurable TFOM threshold.

4.2.25.4 TSYNC_SS_setMaxTfom

```
TSYNC_ERROR TSYNC_SS_setMaxTfom(
    TSYNC_BoardHandle hnd,
    TFOM              tfom);
```

Description:
Set the maximum TFOM threshold to stay in sync.
Input Parameters:
hnd: Board handle
tfom: Pointer to the maximum TFOM information
Returns:
(TSYNC_SUCCESS) Success

Sets the user-configurable TFOM threshold. When the system TFOM is above this threshold, the system will leave the “sync” state end enter the “holdover” state. The default threshold is 15. Since 15 is the highest TFOM value, this means that the threshold will never be triggered.

4.2.25.5 TSYNC_SS_getTfom

```
TSYNC_ERROR TSYNC_SS_getTfom(
    TSYNC_BoardHandle  hnd,
    TFOM              *tfom);
```

Description:
Get the current system TFOM.
Input Parameters:
hnd: Board handle
Output Parameters:
tfom: Pointer to the current system TFOM result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.6 TSYNC_SS_getSync

```
TSYNC_ERROR TSYNC_SS_getSync(
    TSYNC_BoardHandle  hnd,
    int               *bSync);
```

Description:
Get the current sync state.
Input Parameters:
hnd: Board handle
Output Parameters:
bSync: Pointer to the current sync state result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.7 TSYNC_SS_getHoldover

```
TSYNC_ERROR TSYNC_SS_getHoldover(
    TSYNC_BoardHandle  hnd,
    int                *bHoldover);
```

Description:
Get the current holdover state.
Input Parameters:
hnd: Board handle
Output Parameters:
bHoldover: Pointer to the current holdover state result
Returns:
(TSYNC_SUCCESS) Success

Where 0 indicates not in Holdover (False) and 1 indictes in Holdover (True)

4.2.25.8 TSYNC_SS_getHoldoverTO

```
TSYNC_ERROR TSYNC_SS_getHoldoverTO(
    TSYNC_BoardHandle  hnd,
    unsigned int       *nHoldoverTimeout);
```

Description:
Get the current holdover timeout.
Input Parameters:
hnd: Board handle
Output Parameters:
nHoldoverTimeout: Pointer to the current timeout state result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.9 TSYNC_SS_setHoldoverTO

```
TSYNC_ERROR TSYNC_SS_setHoldoverTO(
    TSYNC_BoardHandle hnd,
    unsigned int      nHoldoverTimeout);
```

Description:
Set the current holdover timeout.
Input Parameters:
hnd: Board handle
nHoldoverTimeout: Pointer to the current timeout state information
Returns:
(TSYNC_SUCCESS) Success

Time reads:
Where <time type>
0= DOY time (Year, Day of Year, Hour, Min,Sec, nsec

1= BCD time (BCD Year, Day of Year, Hour, Min, Sec,ms, us)
2= Seconds (Total number of seconds inepock, nsec)

4.2.25.10 TSYNC_SS_getTimestamp

```
TSYNC_ERROR TSYNC_SS_getTimestamp(
    TSYNC_BoardHandle  hnd,
    SS_TS_SRC          src,
    TSYNC_TimeObj     *pObj);
```

Description:
Get the specified state change timestamp.
Input Parameters:
hnd: Board handle
src: The timestamp source
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

Where <time type>:
0=DOY (Day of Year)
1- BCD (Binary Coded Decimal)
2- Sec (Straight number of seconds since midnight)

4.2.25.11 TSYNC_SS_getTimestampBcd

```
TSYNC_ERROR TSYNC_SS_getTimestampBcd(
    TSYNC_BoardHandle  hnd,
    SS_TS_SRC          src,
    TSYNC_TimeBCDObj  *pObj);
```

Description:
Get the specified state change timestamp in BCD format.
Input Parameters:
hnd: Board handle
src: The timestamp source
Output Parameters:
pObj: Pointer to the time result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.12 TSYNC_SS_getTimestampSec

```
TSYNC_ERROR TSYNC_SS_getTimestampSec(
    TSYNC_BoardHandle  hnd,
```

```
    SS_TS_SRC          src,
    unsigned int      *nSeconds,
    unsigned int      *nNanos);
```

Description:
Get the specified state change timestamp in seconds format.
Input Parameters:
hnd: Board handle
src: The timestamp source
Output Parameters:
nSeconds: Pointer to the seconds time result
nNanos: Pointer to the nanoseconds time result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.13 TSYNC_SS_getUptime

```
TSYNC_ERROR TSYNC_SS_getUptime(
    TSYNC_BoardHandle  hnd,
    unsigned int      *nUptime);
```

Description:
Get the board's total uptime in minutes.
Input Parameters:
hnd: Board handle
Output Parameters:
nUptime: Pointer to the uptime result
Returns:
(TSYNC_SUCCESS) Success

4.2.25.14 TSYNC_SS_getFreeRun

```
TSYNC_ERROR TSYNC_SS_getFreeRun(
    TSYNC_BoardHandle  hnd,
    int               *bFreerun);
```

Description:
Get the current freerun state.
Input Parameters:
hnd: Board handle
Output Parameters:
bFreerun: The freerun result
Returns:
(TSYNC_SUCCESS) Success

4.2.26 Upgrade Service (US) Calls

US calls are used to upgrade the firmware and FPGA images in the external flash memories.

4.2.26.1 TSYNC_US_GetState

```
TSYNC_ERROR TSYNC_US_getState(
    TSYNC_BoardHandle  hnd,
    TSYNC_StateObj    *obj);
```

Description:
Retreive the update status.
Input Parameters:
*hw: Handle
*obj: Pointer to the state result
Returns:
(TSYNC_SUCCESS) Success

4.2.26.2 TSYNC_US_Start

```
TSYNC_ERROR TSYNC_US_start(
    TSYNC_BoardHandle       hnd,
    TSYNC_FSImageHeaderObj *obj);
```

Description:
Begin an image update sequence.
Input Parameters:
*hw: Handle
Output Parameters:
*obj: Pointer to the update header information
Returns:
(TSYNC_SUCCESS) Success

4.2.26.3 TSYNC_US_Data

```
TSYNC_ERROR TSYNC_US_data(
    TSYNC_BoardHandle    hnd,
    TSYNC_UpdateDataObj *obj);
```

Description:
Send a single data block in the update sequence.
Input Parameters:
*hw: Handle
Output Parameters:
*obj: Pointer to the update data information
Returns:

(TSYNC_SUCCESS) Success

4.2.26.4 TSYNC_US_End

```
TSYNC_ERROR TSYNC_US_end(
    TSYNC_BoardHandle   hnd,
    TSYNC_UpdateEndObj *obj);
```

Description:
Finish the update sequence.
Input Parameters:
*hw: Handle
Output Parameters:
*obj: Pointer to the update end information
Returns:
(TSYNC_SUCCESS) Success

4.2.26.5 TSYNC_US_Cancel

```
TSYNC_ERROR TSYNC_US_cancel(
    TSYNC_BoardHandle hnd,
    FS_IMG            imageType);
```

Description:
Cancel the update sequence.
Input Parameters:
*hw: Handle
Output Parameters:
imageType: the type of update sequence being cancelled
Returns:
(TSYNC_SUCCESS) Success

4.2.27 Oscillator Component (XO) Calls

XO calls analyze frequency measurements and make corrective adjustments to the timing system oscillator (disciplining).

4.2.27.1 TSYNC_XO_getDiscState

```
TSYNC_ERROR TSYNC_XO_getDiscState(
    TSYNC_BoardHandle  hnd,
    int               *disc);
```

Description:
Get the external oscillator's disciplining state.

Input Parameters:
hnd: Board handle
Output Parameters:
disc: The disciplining state result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.2 TSYNC_XO_getMode

```
TSYNC_ERROR TSYNC_XO_getMode(
    TSYNC_BoardHandle  hnd,
    XO_MODE           *mode);
```

Description:
Get the external oscillator's mode used when or testing.
Input Parameters:
hnd: Board handle
Output Parameters:
mode: The mode result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.3 TSYNC_XO_setMode

```
TSYNC_ERROR TSYNC_XO_setMode(
    TSYNC_BoardHandle hnd,
    XO_MODE           mode);
```

Description:
Set the external oscillator's mode used when disciplining or testing.
Input Parameters:
hnd: Board handle
mode: The mode information
Returns:
(TSYNC_SUCCESS) Success

4.2.27.4 TSYNC_XO_getDac

```
TSYNC_ERROR TSYNC_XO_getDac(
    TSYNC_BoardHandle  hnd,
    unsigned short    *dac);
```

Description:
Get the external oscillator's DAC setting for testing.
Input Parameters:
hnd: Board handle

Output Parameters:
dac: The DAC result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.5 TSYNC_XO_setDac

```
TSYNC_ERROR TSYNC_XO_setDac(
    TSYNC_BoardHandle hnd,
    unsigned short    dac);
```

Description:
Set the external oscillator's DAC setting for testing.
Input Parameters:
hnd: Board handle
dac: The DAC information
Returns:
(TSYNC_SUCCESS) Success

4.2.27.6 TSYNC_XO_getAlarm

```
TSYNC_ERROR TSYNC_XO_getAlarm(
    TSYNC_BoardHandle  hnd,
    unsigned int      *alarm);
```

Description:
Get the external oscillator's alarm state.
Input Parameters:
hnd: Board handle
Output Parameters:
alarm: The alarm result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.7 TSYNC_XO_getSerNum

```
TSYNC_ERROR TSYNC_XO_getSerNum(
    TSYNC_BoardHandle  hnd,
    unsigned int      *sernum);
```

Description:
Get the external oscillator's serial number.
Input Parameters:
hnd: Board handle
Output Parameters:
sernum: The serial number result
Returns:

(TSYNC_SUCCESS) Success

4.2.27.8 TSYNC_XO_getMfrMdl

```
TSYNC_ERROR TSYNC_XO_getMfrMdl(
    TSYNC_BoardHandle  hnd,
    TSYNC_ManModObj   *pObj);
```

Description:
Get the external oscillator's manufacturer and model.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the man/mod result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.9 TSYNC_XO_getMessage

```
TSYNC_ERROR TSYNC_XO_getMessage(
    TSYNC_BoardHandle       hnd,
    TSYNC_CustomMessageObj *pObj);
```

Description:
Get a custom message response from the external oscillator.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the message result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.10 TSYNC_XO_setMessage

```
TSYNC_ERROR TSYNC_XO_setMessage(
    TSYNC_BoardHandle       hnd,
    TSYNC_CustomMessageObj *pObj);
```

Description:
Send a Custom Message to the external oscillator.
Input Parameters:
hnd: Board handle
pObj: Pointer to the message information
Returns:
(TSYNC_SUCCESS) Success

4.2.27.11 TSYNC_XO_getCmd

```
TSYNC_ERROR TSYNC_XO_getCmd(
    TSYNC_BoardHandle  hnd,
    TSYNC_OscDiscObj  *pObj);
```

Description:
Get a disciplining dataset from the external oscillator.

Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the dataset result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.12 TSYNC_XO_setCmd

```
TSYNC_ERROR TSYNC_XO_setCmd(
    TSYNC_BoardHandle  hnd,
    TSYNC_OscDiscObj  *pObj);
```

Description:
Send a disciplining command and dataset to the external oscillator.

Input Parameters:
hnd: Board handle
pObj: Pointer to the command information
Returns:
(TSYNC_SUCCESS) Success

4.2.27.13 TSYNC_XO_getPhaseErr

```
TSYNC_ERROR TSYNC_XO_getPhaseErr(
    TSYNC_BoardHandle  hnd,
    int               *err);
```

Description:
Gets the estimated phase error of the oscillator.
Input Parameters:
hnd: Board handle
Output Parameters:
err: pointer to phase error result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.14 TSYNC_XO_getFreqErr

```
TSYNC_ERROR TSYNC_XO_getFreqErr(
    TSYNC_BoardHandle  hnd,
    float              *err);
```

Description:
Gets the frequency error of the oscillator.
Input Parameters:
hnd: Board handle
Output Parameters:
err: pointer to Frequency error result
Returns:
(TSYNC_SUCCESS) Success

4.2.27.15 TSYNC_XO_getOscType

```
TSYNC_ERROR TSYNC_XO_getOscType(
    TSYNC_BoardHandle  hnd,
    OSC                *oscType);
```

Description:
Get the system oscillator type.
Input Parameters:
hnd: Board handle
Output Parameters:
oscType: pointer to the Oscillator Type
Returns:
(TSYNC_SUCCESS) Success

4.2.27.16 TSYNC_XO_getCalVal

```
TSYNC_ERROR TSYNC_XO_getCalVal(
    TSYNC_BoardHandle  hnd,
    float              *cal);
```

Description:
Get the calibration value of the external oscillator.
Input Parameters:
hnd: Board handle
Output Parameters:
cal: calibration value
Returns:
(TSYNC_SUCCESS) Success

4.2.28 Oscillator Monitor Service (XS) Calls

XS calls measure and provide the accuracy and stability of the timing system oscillator.

4.2.28.1 TSYNC_XS_Register

```
TSYNC_ERROR TSYNC_XS_register(
    TSYNC_BoardHandle  hnd,
    TSYNC_MeterHandle *pObj);
```

Description:
Reserve a meter.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the meter handle result
Returns:
(TSYNC_SUCCESS) Success

4.2.28.2 TSYNC_XS_Unregister

```
TSYNC_ERROR TSYNC_XS_unregister(
    TSYNC_BoardHandle  hnd,
    TSYNC_MeterHandle *pObj);
```

Description:
Free the specified meter.
Input Parameters:
hnd: Board handle
pObj: Pointer to the meter handle information
Returns:
(TSYNC_SUCCESS) Success

4.2.28.3 TSYNC_XS_GetWindowSize

```
TSYNC_ERROR TSYNC_XS_getWindowSize(
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterWinSizeObj *pObj);
```

Description:
Get the specified meter's window size. Size is in seconds.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the window size result
Returns:

(TSYNC_SUCCESS) Success

4.2.28.4 TSYNC_XS_SetWindowSize

```
TSYNC_ERROR TSYNC_XS_setWindowSize(
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterWinSizeObj *pObj);
```

Description:
Set the specified meter's window size. Size is in seconds.
Input Parameters:
hnd: Board handle
pObj: Pointer to the window size information
Returns:
(TSYNC_SUCCESS) Success

4.2.28.5 TSYNC_XS_GetMeterData

```
TSYNC_ERROR TSYNC_XS_getMeterData(
    TSYNC_BoardHandle   hnd,
    TSYNC_MeterDataObj *pObj);
```

Description:
Get the specified meter's error data. Error data is in nanoseconds.
Input Parameters:
hnd: Board handle
Output Parameters:
pObj: Pointer to the meter data result
Returns:
(TSYNC_SUCCESS) Success

4.2.28.6 TSYNC_XS_MeterCmd

```
TSYNC_ERROR TSYNC_XS_meterCmd(
    TSYNC_BoardHandle      hnd,
    TSYNC_MeterCommandObj *pObj);
```

Description:
Send a command to a meter.
Input Parameters:
hnd: Board handle
pObj: Pointer to the meter command information
Returns:
(TSYNC_SUCCESS) Success

5 Example Routines

The available API calls can be used individually or grouped together to perform various functions, such as to generate and process interrupts at specified moments, generate time stamps each time an event occurs or at a specified time (Match Time), etc. This section provides information on commonly used functionalility of the TSync boards.

5.1 Interrupt Generation

The TSync board can generate an interrupt when specified events occur. The TSync driver can differentiate between the different events that can cause interrupts. The TSync driver handles the interrupt and provides information on the type of event that caused the interrupt to the caller. Applications can wait on a certain type of interrupt using the "`TSYNC_waitFor`" call. For instance, one application can wait for an event to occur on the GPO0 pin, and another application can wait for a timestamp event to occur.

The events that generate interrupts are listed and described in Section 5.9 of the TSync-cPCI user manual.

The "intType" value of the "`waitfor`" call configures which interrupt to wait for before proceeding. Refer to the table below for the intType values and their corresponding interrupt types.

intType value	Interrupt type
0	`1PPS Received`
1	`Timing System Service Request`
2	`Local / uC Bus FIFO Empty`
3	`Local / uC Bus FIFO Overflow`
4	`uC / Local Bus FIFO Data Ready`
5	`uC / Lcoal Bus FIFO Overflow`
6	`GPIO Input Event`
7	`Timestamp Data Ready`
8	`GPIO Output Event`

Interrupt usage is set up by unmasking the interrupt using the `TSYNC_HW_setIntMask` API call (Refer to Section 4.2.16.15). The user's application software would then wait on an interrupt utilizing the `TSYNC_waitfor` API call (Refer to Section 4.2.5).

Assuming the user would use the General Purpose Outputs (GPO) in square wave mode to generate periodic interrupts, you would need to set the output mode using the `TSYNC_GO_setMode` API (Refer to Section 4.2.12.7). Then you would need to set the square wave configuration using the `TSYNC_GO_setSquareWave` API (Section 4.2.12.13). Finally, you would enable the GPO using the `TSYNC_GO_setEnable` API (Section 4.2.12.4).

After perfoming these calls, you would receive an interrupt based on the period that the GPO is set to.

TSYNC_hwSetIntMask

The index of TSync's `hwSetIntMask` call is used for interrupt types that have more than one source, like GPIO. The index is then used to indicate which instance of that interrupt type to mask/unmask. If you wanted to unmask the GPI input 1, for example, then your interrupt type would be GPI and your index would be 1 to indicate general purpose input 1.

Interrupt FAQs

Q. Is there a way to determine how many interrupts we have missed from one call to 'wait _for_interrupt' to the next?

A. Not through the driver itself. You may be able to determine it through the OS. In Linux, the OS keeps a running total of the interrupts generated by any interrupt line in `/proc/interrupts`. Just be aware that it is collecting all interrupts on that line, so if it is shared with other devices, those will be shown as well. Also, all interrupts from the TSync will be aggregated in the tally.

5.2 External Event Input (Time Stamping)

The GPI pins allow an external signal to be received as a trigger to generate timestamps. When the selected active edge is received, a timestamp is generated and stored in a buffer, out of which it can be read when needed.

For event input to be enabled, time stamping must be enabled using the `TSYNC_HW_SetTSEnable` command. This command only needs to be run once.

NOTE: On a Windows system, the Windows driver must be used to enable a true event. While the Windows control utility included with the Windows driver can handle some of the functionality of the external event *input*, the control utility by itself can only show simulated input and cannot handle a true applied input to the board.

In addition, the GPIO (Input and Output) pin must be enabled. GPIO pins are not enabled by default, so the signal will not be noticed until the specific pins to be used are enabled when the input signal is attached.

NOTE: Which breakout cable is being used with the board determines which GPIO pins can be used. The available enhanced breakout cable allows for connectivity to all of the GPIO pins. The standard breakout cable allows for connection to GPI Input 0 only.

Once the input is enabled, there are commands available to define whether the board should trigger on the rising edge or the trailing edge, as desired.

The following are example program commands to enable GPIO Input 0 for timestamps on the rising edge:

1. `HW_SetTsEnable 0 1` (Enables time stamping system)
2. `GI_SetTsEnable 0 0 1` (Enables GPIO Input 0 time stamping)
3. `GI_SetEdge 0 0 1` (Enables GPIO Input 0 rising active edge)

The following are the Information/Reading timestamp commands:

- `HW_GetTsCount 0 1` (Gets number of timestamps collected from GPIO input 0)
  NOTE: Index 0 is for Host generated time stamps.
- `HW_GetTsData 0 1` (Retrieves all collected time stamps from GPIO input 0)
  NOTE: Index 0 is for Host generated time stamps.
- `HW_GetTsSingle 0 1` (Retrieves one collected time stamp from GPIO input 0)
  NOTE: Index 0 is for Host generated time stamps.

NOTE: The `TSYNC_HW_GetTsSingle` command will only grab the "oldest" timestamp out of the FIFO buffer, taking one timestamp at a time (Refer to Section 4.2.16.16). To retrieve all of the timestamps in the buffer, rather than just one, use the `TSYNC_HW_getTSdata` command (Refer to Section 4.2.16.8).

The current state of GPIO Input 0 can be read at any time with the following command:

- `GI_GetValue 0 0` (Retrieves the current state of GIPO Input 0)

The Host Driver contains a 512-deep FIFO for each individual timestamp input. If the Application Software allows a timestamp FIFO to overflow, older timestamps will be discarded as new ones come in.

The timestamp interrupt will only trigger when a new timestamp occurs. It will not keep triggering if the data is not read out. Because of this, using the `TSYNC_HW_getTsData` command instead of the `TSYNC_HW_getTsSingle` command is recommended (`TSYNC_HW_getTsData` will grab all timestamps that may be contained in the buffer by the time they are able to be retrieved instead of just the oldest timestamp).

NOTE: You should refer to the `TMSTMP_SRC` enumeration when using driver calls or example programs that deal with timestamps. Host timestamps are read using Source 0, timestamps for GPI pins 0 through 1 use Sources 1 through 4.

5.3 Match Time

The General I/O is configurable as a Match Time Event pin (GPO1), which will activate at a preset time and become inactive at another preset time. The Match Time Event provides two user-settable times to make the General I/O pin active and inactive. The Match Time Event-configured General I/O pin has a programmable edge, allowing the selection of Low to High or High to Low.

The following are commands utilized for match time:

`GO_SetEnable:` (General enable of a general purpose output)

`GO_SetMode:` (Set the general purpose output to match time mode (mode = 1))

`GO_SetMatchEnable:` (Enable/disable the time match of a general purpose output. Each output must be enabled for matching high level times and low level times.

`HW_SetMatchTimeHi:` (Set the time at which the specified general purpose output should go high. Time can be set up to 100 days ahead and as close as 50 msec.)

`HW_SetMatchTimeLo:` (Set the time at which the specified general purpose output should go low. Time can be set up to 100 days ahead and as close as 50 msec).

The three "`GO`" commands (`GO_SetEnable`, `GO_SetMode` and `GO_SetMatchEnable`) are "one time commands" that only need to be entered after the board is powered up to enable the match time functionality. Once these commands have been run once, the board is then available to start performing match times and providing outputs (either lows or highs) when the match times occur, until the board is reset/power cycled.

Once the match time mode and the match output pin have been enabled with the three above commands, the state of the match time output pin (a high or a low) is then controlled by the `HW_SetMatchTimeHi` and `HW_SetMatchTimeLo` commands. These two commands can be run in a pair if desired to change the state and then reset it back to the original state (changing it to one state at a specified time and then changing it back to the original state at a different specified time), or you can run just one or the other to change the state and then maintain that state (occurring at one specified time).

NOTE: Be sure to configure the board's timescale to match the timescale of the times being entered. Otherwise, matches won't happen as expected. For example, if the default board timescale is UTC and the match times are entered as local time, the match times will be several hours off. Refer to **5.6 Changing the TSync Boards TimeScale to Local Time Instead of UTC** below.

Match Time FAQs

Q. Should these calls be made in any particular order ?

A. The calling order for the three `TSYNC_GO_XXX` commands is not necessarily order sensitive, but the commands are required to set up the output to function with the `TSYNC_HW_setMatchTimeXX` calls. Note that if the output is enabled with `TSYNC_GO_setEnable`, whatever the output should be, based on the currently selected output mode, will appear on the general purpose outputs.

Q. The meaning of `TSYNC_GO_setEnable` is unclear, does it enable the pulse capability or turn the pulse on?

A. This call is specific to the outputs to the pin, enabling (turning on) the general purpose output signal to the pin. All of the modes operate within the hardware and can be configured without being enabled by this call, but the outputs must be enabled by this call to be reflected on the pins.

Q. `GO_SetMatchEnable` has a level argument which seems strange given the names of the two functions after it. Since I want a high level pulse, I am assuming that I should use LEVEL_HIGH… is this correct ?

A. The two functions `TSYNC_HW_setMatchTimeHi` and `TSYNC_HW_setMatchTimeLo` are calls that go explicitly to the hardware. `TSYNC_GO_setMatchEnable` is a call to the software component and provides a single call to enable high time matching or low time matching. LEVEL_HIGH is passed to specify that matching against the high match time is to be enabled. LEVEL_LOW is passed to specify that matching against the low match time is to be enabled..

Q. What is the default state of the General Purpose Outputs on power-up, before any calls are made?

A. All general purpose outputs default to low outputs at system startup.

5.4 Using the Host PC as an External Time Reference for the TSync Board

There are two ways to use the TSync board without a GPS or IRIG input: “self” mode and “Host” mode. This is configured in the reference priority table.

When “self” is used as a reference, manually entered time is automatically considered valid data. Conversely, when “Host” is used as a reference, entered time is not automatically considered valid data, and driver calls must be used to command the system to consider entered time as valid data before a manually entered time can be used.

On older TSync boards (with software version 1.x.x), the external 1PPS input is paired with a “Host” time reference. Therefore, in order for the TSync board to synchronize to the external 1PPS, the time must be set on the board through the Host Reference and the Host Reference must be set as Valid.

NOTE: An alternative method is to add a reference table entry for “self” “epp0”, which pairs the external 1PPS with the “self” reference instead of the “Host” reference. The “self” reference as a time reference then assumes that whatever the TSync has for time is valid, whether it was set by a host or not.

To use the "`hst0`" as the time reference:

(1) enable an entry into the working priority table to allow the PC to be the time reference for the board ("`hst0`") and to establish it as the highest priority. (Refer to the `RS_addentry` command).

The full command structure for this is:

```
rs_addentry 0 1 1 "hst0" "epp0"
```

(2) Set the time using the `cs_settime` command (Refer to Section 4.2.8.2).

The command structure for this command is:

`cs_settime 0 0` (followed by the desired time and date).

Once the time has been manually set,

(3) Declare the host as having valid time using the `hr_setvalidity` command. (Refer to Section 4.2.15.2).

The command structure for this command is:

```
Hr_setvalidity 0 1
```

After this command has been issued, the TSync board will use the host as its time reference. As long as the host is synchronized to the external GPS receiver, the host's time/date will be synced to GPS, and the board will be synced to the host's time.

NOTE: If the host is not externally synced to the GPS receiver, the host PC and the board will have the same time, but they won't match the time of the GPS receiver.

Once changes have been made to the Reference Table, this table can be saved in the card and can be set to persist across reboots by using this command:

```
RS_setUserDef 0
```

Step (2), setting the time, and step (3), declaring the validity of the time input, will need to be performed manually or via the driver each time the host PC is powered-down (i.e., if it isn't left up and running all of the time).

Be advised that the default time scale for the TSync board is UTC. This differs from raw GPS by 15 seconds currently, due to periodic leap seconds added to UTC time scale. If the computer is synchronized with raw GPS time, you will likely want to change the time scale of the board to GPS as well, so that all devices will be on the same time scale (refer to `cs_setTimeScale`).

Newer TSync boards (with software version 2.0.0 or greater), come with a new entry option in the default reference table: “self” for time and “epp0” for 1PPS. With this new entry in the reference table, the board will consider itself in sync if only a 1PPS signal is connected: the card will use itself as its time reference. When operating in this mode, `CS_GetTime` can be used to set the current time. If this entry is undesirable for your application, remove the entry and save the resulting reference table.

5.5 Converting the GPO Outputs to 1PPS Output Signals

Using the GPOs as 1PPS outputs is possible in square wave mode. Simply set up the square wave to output a 1PPS with the pulse width, active edge, and offset you want.

There are some considerations to take into account when using it, however. The square wave output as currently designed is only aligned to the 1PPS once when configured. That means that during initial alignment of the board’s internal 1PPS to an input using HW smoothing, before

the SW takes over with disciplining, the square wave outputs lose their alignment. Once SW takes over (which can be determined by looking at the disciplining state or waiting for the `SS_TFOM` value to fall below 15), if a square wave is set up as a 1PPS, it will always be aligned from that point on.

The following are the API calls used to set up a 1PPS signal using a GPO. Each API call has a corresponding example program showing usage.

1. **`TSYNC_GO_setMode`**
   Used to set the specified GPO to square wave mode (OD_MODE_SQUARE_WAVE = 2)

2. **`TSYNC_GO_setSquareWave`**
   Used to set up the square wave parameters. The output square wave is aligned to the internal 1PPS once after this call is made.
   - Offset: Offset from the 1PPS
   - Period: Period of the Square Wave in ns (1000000000ns for 1PPS)
   - Pulse Width: Pulse Width of the Square Wave (in nanoseconds)
   - Active Edge: Whether square wave is rising or falling edge.

3. **`TSYNC_GO_setEnable`**
   Used to enabled / disable a given GPO

4. **`TSYNC_PP_setPulseWidth`**
   Used to set up the Pulse Width of the 1PPS output (in nanoseconds).

5.6 Changing the TSync Board's TimeScale to Local Time Instead of UTC

The default timescale for the TSync board is UTC time. The TSync board can be set to use a local timescale for the core clock instead of being configured for UTC. The local timescale needs to be set up and then the system needs to be set to utilize that timescale.

First, you need to set the local Time Zone Offset from UTC using the `TSYNC_CS_setTimeZoneOff` call. The offset should be provided in seconds offset from UTC.

If you want to provide DST information, use the `TSYNC_CS_setDstRule` library call. For example, the DST rule for the Eastern Time Zone is:

Reference:		0	(Local time reference)
In:	Week:	2	
	Day:	0	(Sunday)
	Month:	3	(March)
	Hour:	2	
Out:	Week:	1	
	Day:	0	(Sunday)
	Month:	11	(November)
	Hour:	2	
	Offset:	3600	(1 hour)

The final step is to tell the clock to utilize the local timescale with the `TSYNC_CS_setTimescale` library call. Once you change the board's time scale to local, you can enter the match times in local time as well as they will then correlate to the same time base.

5.7 Operation of the TSync-cPCI-PTP

This document makes use of the Example Programs that are included as part of the TSync driver installation.

`<device>` = which TSync card is in the system, starting at 0.
`<inst>` = which PTP instance to use. TSync/PTP has only one PTP module, so use Instance 0 (this corresponds with the ptp0 entry in the State Table).

5.7.1 Basic Commands

5.7.1.1 Resetting the PTP Interface

To reset the PTP operation:

```
PTR_ResetModule <device> <inst> <reset type>
```

Reset Type 0 = Cold Reset
Reset Type 1 = Hot Reset
Reset Type 2 = Reset to Factory Defaults

5.7.1.2 Operational Mode

To put the TSync/PTP into PTP Slave Mode:

```
PTR_SetMode <device> <inst>  0
```

To put the TSync/PTP into Master Mode:

```
PTR_SetMode <device> <inst> 1
```

After setting the Operational Mode, issuing the following commands will make that change persist across reboots and reset the module to recognize the new settings:

```
PTR_SaveSettingstoROM <device> <inst>
PTR_ResetModule <device> <inst> 0
```

5.7.1.3 Ethernet Configuration

DHCP/Static IP

To read the current IP information:

```
PTR_GetEthernetITF <device> <inst>
```

To use a Static IP:

```
PTR_SetEthernetITF <device> <inst> <0> <Static IP> <Netmask> <Gateway>
```

To use DHCP:

```
PTR_SetEthernetITF <device> <inst> <1> <Static IP> <Netmask> <Gateway>
```

NOTE: The PTP module will only attempt to find a DHCP address once, when the module is first powered-up or after a reset. If the module cannot find an IP address, it will use the Static IP address given in this `PTR_SetEthernetITF` command. To re-acquire an IP address, reset the module.

After changing the Ethernet settings (either static or DHCP), issue the following commands to persist the settings and reset the module:

```
PTR_SaveSettingstoROM <device> <inst>
PTR_ReinitModule <device> <inst> 0
```

5.7.2 Basic PTP Configuration

5.7.2.1 Configuration as Slave Device

The TSync-cPCI-PTP is configured from the factory to function as a PTP Slave. It should synchronize to a PTP Master set up with the following parameters:

- Announce Rate = once every 4 seconds or faster
- Delay Mechanism = End-to-End
- Multicast operation active (Unicast disabled)
- Two-Step operation

When the TSync-cPCI-PTP card is first connected to a network that contains an active PTP Master, it may take up to a minute for the Port State to change to the “slave” state. After that, it will take up to two minutes for the PTP connection to be accepted as a valid reference by the TSync/PTP.

If the TSync-cPCI-PTP device is not entering the “Slave” Port state (as reported by `PTR_GetPortState`), check the following configuration parameters:

- Make sure the Operation Mode is set to Slave.
- Make sure a valid IP address is currently being used.

NOTE: If DHCP is enabled and the TSync / PTP was not successful in obtaining an IP address, the Module will need to be Reset. Use the "`PTR_ResetModule`" command to reobtain an IP address:

```
PTR_ResetModule <device> <inst> 0
```

The Reset command may take up to two minutes to complete.

- If the card was previously used as a PTP Master and has been changed to a Slave, make sure that the Priority1 and Priority2 values are both set to "128."
- Check to make sure that the Master's Announce Rate is within the slave's Announce Reception Timeout interval (refer to the Port Settings section for more information).

5.7.2.2 Quick Configuration as Master Device

In order to convert a TSync card from Slave with default values to Master, the following commands must be issued:

PTR_SetMode <device> <inst> 1	*Sets mode to Master*
PTR_GetUnitSettings <device> <inst>	*Record Clock Identity*
PTR_SetUnitSettings <device> <inst> <clock ID> 0 0 0 1 1	*Provide Clock Identity here Sets priority low*
PTR_SetEthernetITF <device> <inst> <dhcp> <static IP> <netmask> <gw>	*If not already configured*
PTR_SaveSettingsToROM <device> <inst>	*Makes changes persist*
PTR_ResetModule <device> <inst> 0	*Cold Reset of Module*

In order to operate in Master Mode, the TSync card must be synchronized to a non-PTP reference. Available references include:

- IRIG (DCLS or AM)
- External PPS with time set from the Host
- Time set in the card running in "Self" mode (non-traceable)

After the module is reset, the following commands may need to be issued to fully configure the Master:

CS_SetYear <device> <year>	*Sets current year*
CS_SetTimeScaleOff <device> 1 <offset>	*Sets current TAI offset*

5.7.2.3 Advanced Configuration as Master Device

In order to configure the TSync/PTP card to be a Master, the PTP Mode must be set to Master Mode using `PTR_GetMode`. In addition the priority1 and priority2 fields in the Unit Settings command must be changed from their default values of 128 to lower numbers.

When the PTP Module is set to Master Mode, the module will immediately attempt to become the master on the network. If it does, it will start to transmit PTP packets (even if the TSync is not yet synchronized).

There are several reasons why the TSync/PTP card may not become the active master, or may not be broadcasting the correct time, even if Master Mode is enabled:

1. If using any reference other than "self" for 1PPS, the TSync/PTP will not become an active Master until the TFOM value of the system is less than 15. After the TSync/PTP has first gone into sync after power-up, it may take a minute or two for the TFOM value to fall to an acceptable level. Use the "`SS_GetTFOM <device>`" call to determine the current TFOM value.

2. PTP uses the TAI timescale to transfer time. The available non-PTP references on the TSync/PTP card all communicate time in the UTC timescale. UTC is offset from TAI by a small amount which changes every time a leap second occurs. The UTC-TAI Offset is part of the PTP Specification and must be provided to a Master, but none of the available non-PTP references provide that offset. Therefore, in order for the device to become a Master, the offset must be provided by the Host. Use the "`CS_SetTimeScaleOff <device> 1 <offset>`" command to set the offset.

3. If there are multiple masters on the network, the TSync/PTP uses the Best Master Clock algorithm specified in the PTP Specification to decide whether or not to become a master. If, after applying the BMC algorithm, the TSync/PTP determines that there is a better master, the device will transition into the "Passive" port state. A full discussion of the Best Master Clock algorithm is beyond the scope of this document, but the first thing the algorithm considers is the priority1 parameter, with lower values having higher priority. If more than one master is broadcasting on a network, the Master that is broadcasting the lowest number in the priority1 field will become the Active Master, and all other masters will become passive.

Other points of note when operating a TSync/PTP card as a Master:

The Ethernet packets generated by the TSync/PTP card have a TTL (Time-To-Live) value of 1. They will not propagate through standard routers.

If you are using a time source (such as standard IRIG) which does not provide the Current Year, the Current Year can be provided to the TSync device by using the the `CS_SetYear` Example Program. When first setting up a TSync/PTP Master, please use `CS_GetTime` to ensure that the proper year is used.

The PTP standard supports the transmission of Leap Second information. The only available reference on the TSync that provides Leap Second information is IRIG with IEEE extensions, and that is only if the IRIG source supports them.

If the IRIG source can provide IEEE extension information, the TSync must be configured to process it, by using a Coded Expression setting that includes Control Fields, and setting the Control Fields to use the IEEE extensions.

If leap second information is not provided by the source, it must be provided to the TSync through the Host system. There is no limit in the TSync regarding how far in advance it can be set, and it can be set to any arbitrary time. However, the PTPv2 standard only recognizes leap seconds at the second before midnight (as of this writing, Leap Seconds have only occurred on the second before midnight, Jan 1, or the second before midnight, July 1).

Even if a leap second is set at an earlier time in the TSync, it will not be broadcast in the PTP announce message until 12 hours prior to the event, in compliance with the PTPv2 specification. After the leap second event , the leap second value will not be cleared from the TSync until overwritten by a new Leap Second event.

Leap Second information does not persist across power cycles.

Leap Seconds are scheduled using `CS_SetLeapSec`.

To set a positive Leap Second (adding second 60) at midnight, January 1, 2009 on TSync card 0:

```
CS_SetLeapSec 0 1 0 0 0 1 2009
```

To set a negative leap second (skipping second 59), use -1 as the offset.

The current scheduled leap second can be obtained using `CS_GetLeapSec <device>`.

5.7.3 Advanced PTP Configuration

This section provides additional information about how to monitor the state of the PTP interface and how to properly configure it. Only a subset of the available commands are discussed here. Please refer to the **4.2.23 PTP Reference Component (PTR) Calls** for more information on available commands.

This section assumes the reader has some knowledge of the PTPv2 Specification. For more information, please refer to the Specification itself (IEEE 1588-2008). Another good source of PTP information is NIST's IEEE 1588 website (http://ieee1588.nist.gov/).

5.7.3.1 PTP Port State

The "`PTR_GetPortState <device> <inst>`" command provides information on the state of the PTP connection.

- **Port Number**: For this product, will always be "1."
- **Port Enabled**: For this product, will always be "Y."
- **Port State**: Reports the current state of the PTP State Machine. The TSync/PTP uses the following states:
  - *Initializing*: Cable is unplugged / power-up state.
  - *Listening*: TSync/PTP is looking for a Master.
  - *Master*: TSync/PTP Master has become the active master on the network.
  - *Passive*: TSync/PTP Master has become a passive master. (There is another Master on the network with higher priority)
  - *Uncalibrated*: TSync/PTP Slave has detected a Master on the network.
  - *Slave*: TSync/PTP Slave is actively synchronizing to a Master on the network. For more information on Port State definitions, please refer to Section 9.2.4 of the PTPv2 Spec.
- **Link Connected**: reports "Y" if the Ethernet Link is connected, "N" if not.

5.7.3.2 PTP Unit Settings

The "`PTR_GetUnitSettings <device> <inst>`" command provides access to configuration information for the module.

The "`PTR_SetUnitSettings`" command is used to set this configuration information for proper operation of the PTP Network.

- **Clock Identity**: A unique identifier for every PTP device. Consists of eight 8-bit octet fields. By default, fields 1, 2, and 3 match the first three octets of the six-octet MAC address, fields 4 and 5 are "FF" and "FE", respectively, and fields 6, 7, and 8 match the last three octets of the MAC address.
- **One-Step Mode**:
  - "N" : operates in two-step mode (default value).
  - "Y" : operates in one-step mode (this will reduce accuracy).
- **Unicast**: Set to "N" by default, operating in Multicast mode.
  - *Unicast mode is not currently supported.*
- **Domain Number**: Reports the PTP Domain (defaults to 0) (See PTPv2 Sec. 7.1)
- **Priority1**: Reports the PTP Priority1 value (defaults to 128) (See PTPv2 Sec. 7.6.2.2)
- **Priority2**: Reports the PTP Priority2 value (defaults to 128) (See PTPv2 Sec. 7.6.2.3)

5.7.3.3 Port Settings

The "`PTR_GetPortSettings <device> <inst>` "command provides access to configuration information for the broadcast rate of PTP packets.

The "`PTR_SetPortSettings`" command provides write access to configuration information for the broadcast rate of PTP packets.

- **Port Number**: For this product, will always be "1."
- **Announce Reception Timeout**: In order for a slave to synchronize to a Master, it must see at least one announce message coming in during an interval defined as the slave's Announce Interval times the slave's Announce Reception Timeout. Example: if a slave's Announce Reception Timeout is 3, and its Log Announce Interval is 1, it must see an Announce message within 3 * (2^1) = 6 seconds.
- **Log (base 2) of Announce Interval**: A slave uses this value combined with the Announce Reception Timeout to determine its announce timeout. A Master uses this value to determine the rate at which it sends out Announce messages.
- **Log (base 2) of Sync Interval**: A Master uses this value to determine the rate at which Sync messages are transmitted.
- **Log (base 2) of Delay Request interval**: A Master will broadcast this value to Slaves to determine the rate at which Delay Request messages are transmitted (when the End-to-End Delay Mechanism is chosen).
- **Log (base 2) of Peer Delay Request interval**: A Master will broadcast this value to Slaves to determine the rate at which Peer Delay Request messages are transmitted (when the Peer-to-Peer Delay Mechanism is chosen).
- **Delay Mechanism**:
  - 0x01 = End-to-End Delay Mechanism
  - 0x02 = Peer-to-Peer Delay Mechanism
  - 0xFE = Delay Mechanism disabled

  Several parameters are represented as the logarithm to base 2 of the number of seconds between packets. For example:

  Value 0 = 2^0 = 1 second between packets
  Value 1 = 2^1 = 2 seconds between packets
  Value 0 = 2^4 = 16 seconds between packets
  Value -2 = $2^{(-2)}$ = (1/4) = .25 seconds between packets

5.7.3.4 PTP Grandmaster Properties

The "`PTR_GetGMProp <device> <inst>`"command provides access to information about the current GrandMaster (if the TSync/PTP is operating in Master mode, this information will describe the TSync/PTP board itself).

- **Clock Identity**: the GrandMaster's Clock Identity (see description in Unit Settings).
- **Clock Class**: a number describing the state of the clock (see PTPv2 Table 5 of Section 7.6.2.4).
- **Clock Accuracy**: a number describing the accuracy of the oscillator in the Grandmaster (see PTPv2 Spec Section 7.6.2.5).
- Offset Scaled log variance of the Master (see PTPv2 Spec Section 7.6.3).
- Priority1 setting of the Grandmaster.
- Priority2 setting of the Grandmaster.

5.7.4 PTP Firmware Upgrade Instructions

The TSync-cPCI-PTP product has the capability to upgrade the PTP firmware via FTP through the PTP Ethernet port.

In order to properly interface with the simple FTP server in the TSync-cPCI-PTP port, a simple FTP client must be used. More complex graphical FTP clients (that use multiple FTP sockets to communicate) may not communicate properly with the FTP server. Using the basic-command line FTP clients in Windows (or in most major Linux distributions) is recommended.

This process also makes use of several of the example programs that are detailed elsewhere in this section.

Upgrade Steps:

1.) Attach a Windows or Linux PC to the PTP network, and unpack the upgrade bundle on the PC. (This PC may be the same PC that is used as a host for a TSync-cPCI-PTP). Make sure your network mask and gateway are configured properly so that the TSync-cPCI-PTP cards are accessible to the machine containing the upgrade file.

2.) Note the IP addresses for all TSync-cPCI-PTP cards on the network. (These can be obtained by running the "`PTR_GetEthernetITF <card id> 0`" Example Program on the Host for each TSync-cPCI-PTP card).

3.) Launch a command shell in whatever OS you are using on the upgrade PC.

4.) Navigate to the directory containing the PTP upgrade `.bin` file on the upgrade PC.

5.) Type "`ftp xxx.xxx.xxx.xxx`" on the command line, substituting the IP address of the first targeted TSync-cPCI-PTP card. Connect using these credentials:

Username: `spadmin`
Password: `admin123`

6.) **Important:** Set the FTP transfer mode to "binary" before proceeding. (For most FTP clients, this is done by simply typing "`binary`", "`bin`", or "`b`" after first connecting.) **Failure to transfer the image as a Binary file may corrupt the firmware image, and require the unit to be returned to Spectracom for servicing.**

7.) Transfer the binary image using the FTP command "`put xxxxxx.bin`", using the proper filename for the upgrade image. The transfer should take less than 30 seconds to complete on a lightly loaded network.

8.) Type "`quit`" to quit the FTP client.

9.) Reboot the Host Machine.

If a reboot is not desired, use the following example programs instead:

```
PTR_ResetModule <card id> 0 0
SS_Reset <card id> 0
```

Where `<card id>` is 0 for a system with one TSync-cPCI-PTP card in it.

10.) Confirm that the new PTP Firmware Version ID matches the version you are upgrading to (using the "`PTR_GetModuleInfo <card id> 0`" example program).

11.) Upgrade all other TSync-cPCI-PTP cards on the network by restarting on Step 5.

6 TPRO/TSAT Timing Board Driver API Support

The TSync-cPCI supports existing Spectracom TPRO/TSAT timing board driver API calls as described herein.

6.1 Header Files

6.1.1 Tpro.h

```
/*****************************************************************************
**
**  Module  : tpro.h
**  Date    : 04/05/06
**  Purpose : This is the TPRO-PCI interface include file.
**
**  Copyright(C) 2006 Spectracom Corporation. All Rights Reserved.
**
*****************************************************************************/

#ifndef _defined_TPRO_
#define _defined_TPRO_

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

#ifndef DLL_EXPORT
#define DLL_EXPORT /* */
#endif

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

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

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

/*
**  Oscillator frequencies - for Compact PCI Card Only
*/
#define OSC_OUT_OFF      (0)
#define OSC_OUT_1KHZ     (1)
#define OSC_OUT_1MHZ     (2)
#define OSC_OUT_5MHZ     (3)
#define OSC_OUT_10MHZ    (4)

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

```
  int            file_descriptor;
  unsigned short devid;
  unsigned short options;

} TPRO_BoardObj;

/*
** TPRO ALTITUDE OBJECT
*/
typedef struct TPRO_AltObj {

  float meters;

} TPRO_AltObj;

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

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

} TPRO_DateObj;

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

  unsigned short degrees;
  float          minutes;

} TPRO_LongObj, TPRO_LatObj;

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

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

} TPRO_MatchObj;

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

  unsigned char satsTracked; /* num sats tracked */
  unsigned char satsView;    /* num sats in view */

} TPRO_SatObj;

/*
** TPRO HEARTBEAT OBJECT
*/
```

```
typedef struct TPRO_HeartObj {

  unsigned char signalType; /* square or pulse */
  unsigned char outputType; /* jamming option */
  double        frequency;  /* heartbeat freq */

} TPRO_HeartObj;

/*
** TPRO TIME OBJECT
*/
typedef struct TPRO_TimeObj {

  double         secsDouble; /* seconds floating pt */
  unsigned char  seconds;    /* seconds whole num */
  unsigned char  minutes;
  unsigned char  hours;
  unsigned short days;
  unsigned short year;
  unsigned short flags;      /* bit 15 flagsInvalid(1); bit 2 SYNC, bit 1 TCODE; all
others 0 */

} TPRO_TimeObj;

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

  int jiffies;            /*-- # jiffies to wait ---*/
  double  seconds;
  unsigned char  minutes;
  unsigned char  hours;
  unsigned short days;

} TPRO_WaitObj;

/*
** TPRO MEM OBJECT FOR PEEK/POKE
*/
typedef struct TPRO_MemObj {

  unsigned short offset;
  unsigned short value;
  unsigned long  l_value;

} TPRO_MemObj;

#include "tpro_error_codes.h"

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

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

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

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

#endif // _defined_TPRO_
```

6.1.2 Tpro_error_codes.h

```
#ifndef __tpro_error_codes_h__
#define __tpro_error_codes_h__ 1

/* WARNING: X Macros!
 *
 * X Macros will help us keep the list of error conditions tightly
 * coupled with meaningful text messages for users.
 */

#define TPRO_ERROR_CODES                                                \
    TPRO_X( TPRO_SUCCESS                    , 0   , "success")          \
    TPRO_X( TPRO_HANDLE_ERR                 , 1   , "error bad handle") \
    TPRO_X( TPRO_OBJECT_ERR                 , 2   , "error creating obj") \
    TPRO_X( TPRO_CLOSE_HANDLE_ERR           , 3   , "err closing device") \
    TPRO_X( TPRO_DEVICE_NOT_OPEN_ERR        , 4   , "device not opened") \
    TPRO_X( TPRO_INVALID_BOARD_TYPE_ERR     , 5   , "invalid device")   \
    TPRO_X( TPRO_FREQ_ERR                   , 6   , "invalid frequency") \
    TPRO_X( TPRO_YEAR_PARM_ERR              , 7   , "invalid year")     \
    TPRO_X( TPRO_DAY_PARM_ERR               , 8   , "invalid day")      \
    TPRO_X( TPRO_HOUR_PARM_ERR              , 9   , "invalid hour")     \
    TPRO_X( TPRO_MIN_PARM_ERR               , 10  , "invalid minutes")  \
    TPRO_X( TPRO_SEC_PARM_ERR               , 11  , "invalid seconds")  \
    TPRO_X( TPRO_DELAY_PARM_ERR             , 12  , "invalid delay")    \
    TPRO_X( TPRO_TIMEOUT_ERR                , 13  , "device timed out") \
    TPRO_X( TPRO_COMM_ERR                   , 14  , "communication error") \
    TPRO_X( TPRO_DEV_BUSY                   , 15  , "device busy ") \
    TPRO_X( TPRO_MATCH_PARM_ERR             , 16  , "invalid match type") \
    TPRO_X( TPRO_NULL_POINTER               , 17  , "NULL pointer")

enum tpro_error_code {
#define TPRO_X(name, value, string) name = value,
    TPRO_ERROR_CODES
#undef TPRO_X

    TPRO_INVALID_ERROR_CODE
};

const char * tpro_strerror(enum tpro_error_code);

#endif
```

6.2 TPRO/TSAT Driver API Support — Routine Descriptions

6.2.1 TPRO_setOscillator

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

Description:
This routine will select whether the Oscillator output is 10 MHz, 5 MHz, 1 MHz, 1 kHz, or Off. The power-on default state is “OFF”.

Input Parameters:
hnd: Board handle
enbp: Pointer to frequency value

Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.2 TPRO_getLatitude

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

Description:
This routine retrieves the latitude information.

Input Parameters:
Hnd: Board handle

Output Parameters:
Latp: 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

6.2.3 TPRO_getLongitude

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

Description:
This routine retrieves the longitude information.
Input Parameters:
hnd: Board handle
Output Parameters:
Longp: Pointer to TPRO_LongObj
Returns:
(TPRO_INVALID_BOARD_TYPE_ERR) invalid board type for function
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.4 TPRO_getSatInfo

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

Description:
This routine retrieves the number of satellites tracked.
Input Parameters:
hnd: Board handle
Output Parameters:
Satp: Pointer to TPRO_SatObj
Returns:
(TPRO_INVALID_BOARD_TYPE_ERR) invalid board type for function
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.5 TPRO_getTime

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

Description:
This routine retrieves the current time. The seconds value is received as type double.
Input Parameters:
hnd: Board handle
Output Parameters:
Timep: Pointer to TPRO_TimeObj
Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.6 TPRO_resetFirmware

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

Description:
This routine resets the firmware. This function is for troubleshooting purposes only and should not be used in the main application.
Input Parameters:
hnd: Board handle
Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.7 TPRO_setHeartbeat

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

Description:
This routine controls the hearbeat output. The heartbeat output may be a square wave or pulse at various frequencies. This routine is tied to control general purpose output 0.
Input Parameters:
hnd: Board handle
Heartp: Pointer to TPRO_HeartObj
Returns:
(TPRO_FREQ_ERR) invalid frequency value
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.8 TPRO_setMatchTime

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

Description:
This routine drives the match output line high (start time) or low (stop time) when the desired time is met. This routine is tied to control general purpose output 1.
Input Parameters:
hnd: Board handle
Matchp: Pointer to 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 – 60)
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.9 TPRO_setPropDelayCorr

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

Description:

This routine sets the propagation delay correction factor.

Input Parameters:

hnd: Board handle

us: 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

6.2.10 TPRO_setTime

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

Description:

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

Input Parameters:

hnd: Board handle

Timep: Pointer to 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 – 60)
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.11 TPRO_setYear

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

Description:
This routine programs the desired year. If the board is synchronized to a GPS antenna this value will not be accepted.
Input Parameters:
hnd: Board handle
yr: 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

6.2.12 TPRO_simEvent

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

Description:
This routine simulates an external time tag event.
Input Parameters:
hnd: Board handle
Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.13 TPRO_synchControl

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

Description:
This routine commands the 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.
Input Parameters:
hnd: Board handle
enbp: Pointer to the synch enable
Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.14 TPRO_synchStatus

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

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

Input Parameters:
hnd: Board handle

Output Parameters:
enbp: Pointer to the synch status variable

Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.15 TPRO_waitEvent

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

Description:
This routine reports the time of an external time-tagged event from the on-board FIFO. Events are stored in the FIFO and this routine will read the FIFO for events. If the FIFO is empty, the routine will wait for an interrupt and report the event for the timeout specified in the TPRO_WaitObj object. This routine is tied to general purpose input 0.

Input Parameters:
hnd: Board handle

Output Parameters:
waitp: Pointer to the WaitObj

Returns:
(TPRO_COMM_ERR) error communicating with driver
(TPRO_SUCCESS) success

6.2.16 TPRO_waitHeartbeat

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

Description:

This routine will report the status of the heartbeat interrupt. Given an amount of time in jiffies, the return code will report timeout or success determined by the heartbeat interrupt status. This routine is tied to general purpose output 0.

Input Parameters:

hnd: Board handle
jiffies: Pointer ot the jiffies timeout value

Returns:

(TPRO_COMM_ERR) error communicating with driver
(TPRO_TIMEOUT_ERR) timeout waiting on heartbeat interrupt
(TPRO_SUCCESS) success

6.2.17 TPRO_waitMatch

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

Description:

This routine will report the status of the match time interrupt. Given an amount of time in jiffies, the return code will report timeout or success determined by the match time interrupt status. This routine is tied to general purpose output 1.

Input Parameters:

hnd: Board handle
jiffies: Pointer to the jiffies timeout value

Returns:

(TPRO_COMM_ERR) error communicating with driver
(TPRO_TIMEOUT_ERR) timeout waiting on heartbeat interrupt
(TPRO_SUCCESS) success

Document Revision History			
Rev	***ECN***	***Description***	***Date***
A	3316	*First draft of Spectracom documentation for this product.*	December 2013

Spectracom Corporation
1565 Jefferson Road, Suite 460
Rochester, NY 14623
www.spectracomcorp.com
Phone: US +1.585.321.5800
Fax: US +1.585.321.5219