Symmetricom®
(Formerly TrueTime, Inc.)

Table of Contents

Disclaimers

Help file built: 03/13/03

Basic Functions

The following functions are available no matter what mode the device is operating in.

TT_CloseDevice
TT_ConfigurationSettings
TT_FileTimeToSystemTimeEx
TT_GetDeviceInfo
TT_GetHardwareStatus
TT_GetMode
TT_GetOutputBNCSource
TT_GetPhaseCompensation
TT_GetRegister
TT_OpenDevice
TT_SetMode
TT_SetOutputBNCSource
TT_SetPhaseCompensation
TT_SetRegister
TT_SystemTimeExToFileTime

Event Functions

The following functions are used to process hardware generated events. They are available in all operating modes.

TT_Callback
TT_GetExternalEvent
TT_GetExternalEventTriggerEdge
TT_GetExternalEventTriggerSource
TT_GetRateGenerator
TT_GetTimeCompare
TT_RegisterCallback
TT_SetExternalEvent
TT_SetExternalEventTriggerEdge
TT_SetExternalEventTriggerSource
TT_SetRateGenerator
TT_SetTimeCompare

External 1 PPS Mode Functions

The following functions are available when the device is operating in External 1 PPS mode. Unless otherwise specified, these functions are not available in any other mode. See **TT_SetMode** for more information.

TT_GetLeapSecond
TT_PresetTime
TT_ReadDiagnosticRegister
TT_ReadTime

TT_SetLeapSecond

Generator Mode Functions

The following functions are available when the device is operating in Generator mode. Unless otherwise specified, these functions are not available in any other mode. See **TT_SetMode** for more information.

TT_PresetTime
TT_ReadDiagnosticRegister
TT_ReadTime
TT_StartGenerator
TT_StopGenerator

GPS Mode Functions

The following functions are available when the device is operating in GPS mode. Unless otherwise specified, these functions are not available in any other mode. See **TT_SetMode** for more information.

TT_PresetPosition
TT_ReadDiagnosticRegister
TT_ReadGpsInfo
TT_ReadTime

Synthesizer Functions

The following functions are used to control the synthesizer. They are available in all operating modes.

TT_GetSynthesizer
TT_GetSynthesizerOnTimeEdge
TT_GetSynthesizerRunStatus
TT_SetSynthesizer
TT_SetSynthesizerOnTimeEdge
TT_SetSynthesizerRunStatus

Time Code Mode Functions

The following functions are available when the device is operating in Time Code mode. Unless otherwise specified, these functions are not available in any other mode. See **TT_SetMode** for more information.

TT_PresetTime
TT_ReadDiagnosticRegister
TT_ReadTime
TT_ReadTimecodeInfo

About Windows Time

The Win32 API supports several time formats, and numerous functions to get, set, convert, and compare time. The TrueTime SDK supports **FILETIME**, one of these Win32 formats. This enables total integration of TrueTime functionality with Windows system operation.

The following Win32 functions are availabe for use with time.

CompareFileTime
DosDateTimeToFileTime
FileTimeToDosDateTime
FileTimeToLocalFileTime
FileTimeToSystemTime
GetFileTime
GetLocalTime
GetSystemTime
GetSystemTimeAdjustment
GetSystemTimeAsFileTime
GetTickCount
GetTimeZoneInformation
LocalFileTimeToFileTime
SetFileTime
SetLocalTime
SetSystemTime
SetSystemTimeAdjustment
SetTimeZoneInformation
SystemTimeToFileTime
SystemTimeToTzSpecificLocalTime

For more information, see Microsoft Win32 SDK documentation.

Overview of the TrueTime SDK

The TrueTime SDK is designed to make it easy to integrate precision time information into your application.

Using TrueTimeSDK in a C or C++ Application

The TrueTime SDK provides a header file, TrueTimeSDK.h, which defines all structures, enumerations, and functions necessary to use the TrueTime device. Applications must be linked to TrueTimeSDK.lib. On Windows 95/98, TrueTimeSDK.dll and TrueTime.VxD must be distributed with the application, and must be accessible in the current path. On Windows NT, TrueTimeSDK.dll and TrueTime.SYS must be distributed with the application, and TrueTime.SYS must be installed as a kernel-mode driver. Include the TrueTimeSDK.h header file in the header file of your main program. Add the TrueTimeSDK.lib into MSVC++ in the list of linked library files. To do this from the file bar, select Projects->Settings and then select the link TAB.

When distributing your application you must distribute the truetime dll and the approptiate driver with your application.

Windows NT/2000: The truetimePCI.sys file must be installed as a driver in c:\winnt\system32\drivers. The truetime.dll can remain local in the same directory as your application.

Windows 95/98: The truetime.vxd and truetime.dll can remain local in the same directory as your application or be installed in the c:\windows\system directory.

The following functions are available.

TT_Callback
TT_CloseDevice
TT_ConfigurationSettings
TT_FileTimeToSystemTimeEx
TT_GetDeviceInfo
TT_GetExternalEvent
TT_GetExternalEventTriggerEdge
TT_GetExternalEventTriggerSource
TT_GetHardwareStatus
TT_GetLeapSecond
TT_GetMode
TT_GetOutputBNCSource
TT_GetPhaseCompensation
TT_GetRateGenerator
TT_GetRegister
TT_GetSynthesizer
TT_GetSynthesizerOnTimeEdge
TT_GetSynthesizerRunStatus
TT_GetTimeCompare
TT_OpenDevice
TT_PresetPosition
TT_PresetTime
TT_ReadDiagnosticRegister
TT_ReadGpsInfo
TT_ReadTime
TT_ReadTimecodeInfo
TT_RegisterCallback
TT_SetExternalEvent
TT_SetExternalEventTriggerEdge
TT_SetExternalEventTriggerSource
TT_SetLeapSecond
TT_SetMode
TT_SetOutputBNCSource
TT_SetPhaseCompensation
TT_SetRateGenerator
TT_SetRegister
TT_SetSynthesizer
TT_SetSynthesizerOnTimeEdge
TT_SetSynthesizerRunStatus
TT_SetTimeCompare
TT_StartGenerator
TT_StopGenerator
TT_SystemTimeExToFileTime

Module TrueTimeCmn.H

Filename: TrueTimeCmn.h

Description

Defines the common structures, enums and definitions used by Applications/SDK/DDK for the TrueTime 560-590x devices.

Environment:

MSVC++ 5.0/WinNT Kernel Mode Driver/Win95 VxD

Revision History:

Module TrueTimeSDK.H

Filename: TrueTimeSDK.h

Description

Defines the public API for the TrueTime 560-590x devices.

Environment:

MSVC++ 5.0/Kernel Mode

Revision History:

TT_Callback Function

Defined in: TrueTimeSDK.h

TT_API void TT_Callback
(
    TT_EVENT *eEventType* **,**
    FILETIME* *pFileTime* **,**
    DWORD *dwMissingInterrupts* **,**
    PVOID *pContext*
)

Description

This function is a placeholder for a user-defined callback routine to be executed upon external, periodic, time compare or synthesizer events.

Return Value

The function does not return a value.

Parameters

eEventType
    Value specifying the type of event callback (see **TT_EVENT**).

pFileTime
    Address of **FILETIME** structure containing the event time. This field is not defined when *eEventType* is **TT_EVENT_PERIODIC**.

dwMissingInterrupts
    number of missing interrupts.

pContext
    Address of user-defined context.

Example

The following example illustrates the use of this function.

```
TT_STATUS MyCallback(TT_EVENT eEventType, FILETIME* pFileTime, DWORD,
PVOID pContext)
{
    // based on the type of event
    switch(eEventType)
    {
        case TT_EVENT_EXTERNAL:
            // ...
            break;

        case TT_EVENT_PERIODIC:
            // ...
            break;

        case TT_EVENT_TIME_COMPARE:
            // ...
            break;

        case TT_EVENT_SYNTHESIZER:
            // ...
            break;
    }
}
```

See Also **TT_RegisterCallback**

TT_CloseDevice Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_CloseDevice
(
HANDLE *hDevice*
)

Description Detaches the calling application from the specified TrueTime device. This function should be called by every application using the TrueTime SDK, prior to termination. It performs resource deallocation and other cleanup operations.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
The handle to the TrueTime device to close.

Example The following example illustrates the use of this function.

```
HANDLE hDevice

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // close the desired device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_OpenDevice**

TT_ConfigurationSettings Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_ConfigurationSettings
(
HANDLE *hDevice* **,**
TT_CONFIGURATION_SETTINGS *eFunction*
)

Description

This function allows the saving and restoring of board configuration settings. This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eFunction
The functions available for board configuration **TT_CONFIGURATION_SETTINGS**.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_CONFIGURATION_SETTINGS eFunction;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Save the DAC value
    if (TT_ConfigurationSettings(hDevice,eFunction) == TT_SUCCESS)
    {
        // the board configuration has been saved/changed
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_ReadDiagnosticRegister

TT_FileTimeToSystemTimeEx Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_FileTimeToSystemTimeEx
(
  FILETIME* *lpFileTime* **,**
  SYSTEMTIME_EX* *lpSystemTime*
)

Description

This function converts a 64-bit file time to system time format. It only works with **FILETIME** values that are less than 0x8000000000000000.

Return Value

If the function fails, the return value is zero. To get extended error information, call GetLastError.

Parameters

lpFileTime
Pointer to a **FILETIME** structure containing the file time to convert to system date and time format.

lpSystemTime
Pointer to a **SYSTEMTIME_EX** structure to receive the converted file time.

Example

The following example illustrates the use of this function.

```
FILETIME        FileTime;
SYSTEMTIME_EX   SystemTime;
```

```
// Convert the FileTime to SystemTimeEx format
if (TT_FileTimeToSystemTimeEx(&FileTime, &SystemTime) == TT_SUCCESS)
{
    // FileTime is successfully converted to the SystemTimeEx
    ...
}
```

TT_GetDeviceInfo Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetDeviceInfo
(
  HANDLE *hDevice* **,**
  TT_MODEL* *pModel* **,**
  DWORD* *pdwBus* **,**
  DWORD* *pdwSlot*
)

Description

This function returns static information about the board associated with the handle provided. This function is available in all modes and requires **FILE_GENERIC_READ** access.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

pModel
  Address of buffer to receive **TT_MODEL**

pdwBus
  Address of **DWORD** to receive the bus number of the card.

pdwSlot
  Address of **DWORD** to receive the slot number of the card.

Example

The following example illustrates the use of this function.

```
HANDLE hDevice
TT_MODEL    Model;
DWORD       dwBus;
DWORD       dwSlot;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get device info
    if (TT_GetDeviceInfo(hDevice, &Model, &dwBus, &dwSlot) ==
TT_SUCCESS)
    {
        // the requested information is available
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_GetExternalEvent Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetExternalEvent
(
  HANDLE *hDevice* **,**
  BOOL* *bEnableInterrupt*
)

Description

This function returns the current status of the external input event. If a callback has been registered (**TT_RegisterCallback**), it will be called at the external event.

This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

bEnableInterrupt
  Address of **BOOL** to receive current enable status.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
BOOL    bEnableInterrupt;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get external event
    if (TT_GetExternalEvent(hDevice, &bEnableInterrupt) == TT_SUCCESS)
    {
        // external event is available
        ...
    }
}
```

See Also

TT_RegisterCallback TT_SetExternalEvent

TT_GetExternalEventTriggerEdge Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetExternalEventTriggerEdge
(
  HANDLE *hDevice* **,**
  TT_EXTERNAL_EVENT_TRIGGER_EDGE*
peExternalEventTriggerEdge
)

Description

This function gets the polarity of the trigger to the external event This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peExternalEventTriggerEdge
The external event trigger polarity (see **TT_EXTERNAL_EVENT_TRIGGER_EDGE**).

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_EXTERNAL_EVENT_TRIGGER_EDGE eExternalEventTriggerEdge;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the polarity of the trigger to the synthesizer
    if (TT_GetExternalEventSource(hDevice, &eExternalEventTriggerEdge)
== TT_SUCCESS)
    {
        if (eExternalEventTriggerEdge == TT_FALLING)
            // the external event is being triggered on the falling edge
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_SetExternalEventTriggerEdge

TT_GetExternalEventTriggerSource Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetExternalEventTriggerSource
(
HANDLE *hDevice* **,**
TT_EXTERNAL_EVENT_TRIGGER_SOURCE* *peExternalEventTriggerEdge*
)

Description

This function gets the source of the trigger for the external event. This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peExternalEventTriggerEdge
The external event trigger source **TT_EXTERNAL_EVENT_TRIGGER_SOURCE**.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_EXTERNAL_EVENT_TRIGGER_SOURCE eExternalEventTriggerSource;

// open device 0 for Read/Write access
```

```
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the trigger source
    if (TT_GetExternalEventSource(hDevice, &eExternalEventTriggerSource)
== TT_SUCCESS)
    {
        if (eExternalEventTriggerSource == TT_SYNTHESIZER(
            // the external event is being triggered by the synthesizer
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_SetExternalEventTriggerSource

TT_GetHardwareStatus Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetHardwareStatus
(
HANDLE *hDevice* **,**
TT_HARDWARE_STATUS* *pHardwareStatus*
)

Description

This function returns the contents of the status register This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pHardwareStatus
A structure **TT_HARDWARE_STATUS** that will contain the status of the device.

Example

The following example illustrates the use of this function.

```
HANDLE                  hDevice;
TT_HARDWARE_STATUS      HardwareStatus

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the board status
    if (TT_GetStatus(hDevice,&HardwareStatus) == TT_SUCCESS)
    {
        // Check the antenna
        if (HardwareStatus.AntennaShorted)
            // The antenna is shorted

    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_GetLeapSecond Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetLeapSecond
(
  HANDLE *hDevice* **,**
  BOOL* *pbEnable*
)

Description This function returns the current status of the hardware to add a leap second at the end of the current day. This function requires **FILE_GENERIC_READ** access and is available in 1 PPS mode.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

pbEnable
  Address of **BOOL** to receive current enable/disable status.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;
BOOL    bEnable;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get the leap second flag
    if (TT_GetLeapSecond(hDevice, &bEnable) == TT_SUCCESS)
    {
        // the leap second flag is now available
        ...
    }
}
```

See Also **TT_SetLeapSecond**

TT_GetMode Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetMode
(
  HANDLE *hDevice* **,**
  TT_OPERATION_MODE* *peOperationMode* **,**
  TT_SYNCH_SOURCE* *peSynchSource* **,**
  TT_TIMECODE* *peTimeCode*
)

Description This function gets the current operating mode, which is maintained by the hardware in non-volatile memory. Since the hardware is pre-configured by the factory, it is not necessary for any application to call this function. However, it is available for situations where the factory settings must be known. This function requires **FILE_GENERIC_READ** access.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peOperationMode
Address of **TT_OPERATION_MODE** to receive the current operation mode.

peSynchSource
Address of **TT_SYNCH_SOURCE** to receive the current synchronization source when *peOperationMode* is **TT_MODE_SYNCHRONIZED**).

peTimeCode
Address of **TT_TIMECODE** to receive the current timecode standard when *peSynchSource* is **TT_SYNCH_TIMECODE**.

Example

The following example illustrates the use of this function.

```
HANDLE              hDevice;
TT_OPERATION_MODE   OperationMode;
TT_SYNCH_SOURCE     SynchSource;
TT_TIMECODE         TimeCode;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the Current Operating Mode for the desired device
    if (TT_GetMode(hDevice, &OperationMode, &SynchSource, &TimeCode) ==
TT_SUCCESS)
    {
        // Get Mode for the device is successful
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_SetMode

TT_GetOutputBNCSource Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetOutputBNCSource
(
HANDLE *hDevice* **,**
TT_OUTPUT_BNC_SOURCE* *peOutputBncSource*
)

Description

This function gets the source that will be available on the output BNC This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peOutputBncSource
The varible to receive the output BNC source setting as specified in **TT_OUTPUT_BNC_SOURCE**.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_OUTPUT_BNC_SOURCE eOutputBncSource;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the source that is connected to the output BNC
    if (TT_GetOutputBNCSource(hDevice,&eOutputBncSource) == TT_SUCCESS)
    {
        if (eOutputBncSource == TT_OUTPUT_SYNTHESIZER)
            // the synthesizer is being output at the output BNC
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_SetOutputBNCSource**

TT_GetPhaseCompensation Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetPhaseCompensation
(
HANDLE *hDevice* **,**
SHORT* *plOffset*
)

Description Gets the current phase compensation for the device. This function requires **FILE_GENERIC_READ** access, and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
A handle to the TrueTime device, returned by **TT_OpenDevice**. LONG* plOffset // @parm Address of **LONG** to receive the current compensation from -1000 to +1000 microseconds.

plOffset
Address of **LONG** to receive the current compensation from -1000 to +1000 microseconds.

Example The following example illustrates the use of this function.

```
HANDLE hDevice;
LONG    lOffset;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get phase compensation
    if (TT_GetPhaseCompensation(hDevice, &lOffset) == TT_SUCCESS)
    {
        // the phase compensation is available in lOffset
        ...
    }
}
```

See Also **TT_SetPhaseCompensation**

TT_GetRateGenerator Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetRateGenerator
(
HANDLE *hDevice* **,**
TT_GENERATOR_RATE* *peRate* **,**
BOOL* *pbEnableInterrupt*
)

Description

This function obtains the current generator event settings.

This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peRate
Address of **TT_GENERATOR_RATE** to receive the rate at which pulses are generated. A value of **TT_RATE_DISABLE** indicates that pulse generation is disabled.

pbEnableInterrupt
The address to receive the status of the rate generator interrupt.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_GENERATOR_RATE   eRate;
BOOL bEnableInterrupt;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get rate generator events
    if (TT_GetRateGenerator(hDevice, &eRate, &bEnableInterrupt) ==
TT_SUCCESS)
    {
        // the status rate generator events is available
        ...
    }
}
```

See Also **TT_RegisterCallback TT_SetRateGenerator**

TT_GetRegister Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetRegister
(
  HANDLE *hDevice* **,**
  ULONG *ulRegisterOffset* **,**
  UCHAR* *pucRegisterValue*
)

Description

This function returns the contents of the register at the address specified. This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  The handle to the TrueTime device.

ulRegisterOffset
  The address of the register to read.

pucRegisterValue
  The address of the byte to receive the register contents.

Example

The following example illustrates the use of this function.

```
HANDLE      hDevice
ULONG       RegisterOffset;
UCHAR*      RegisterValue;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get device info
    if (TT_GetRegister(hDevice, RegisterOffset, &RegisterValue) ==
TT_SUCCESS)
    {
        // the contents of the register is available
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_SetRegister

TT_GetSynthesizer Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetSynthesizer
(
  HANDLE *hDevice* **,**
  DWORD* *pdwFrequency* **,**

BOOL* *bEnableInterrupt*
)

Description

This function gets the current synthesizer settings.

This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pdwFrequency
Address of **DWORD** to receive the rate at which pulses are generated.

bEnableInterrupt
The address to receive the status of the synthesizer interrupt.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
DWORD   dwFrequency;
BOOL    bEnableInterrupt;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice ==
TT_SUCCESS)
{
    // get synthesizer settings
    if (TT_GetSynthesizer(hDevice, &dwFrequency, &bEnableInterrupt) ==
TT_SUCCESS)
    {
        // the synthesizer settings are now available
        ...
    }
}
```

See Also

TT_RegisterCallback TT_SetSynthesizer TT_GetSynthesizerRunStatus

TT_GetSynthesizerOnTimeEdge Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetSynthesizerOnTimeEdge
(
HANDLE *hDevice* **,**
TT_SYNTHESIZER_ON_TIME_EDGE* *peSynthesizerOnTimeEdge*
)

Description

This function gets the on time edge polarity of the synthesizer This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peSynthesizerOnTimeEdge
The syntesizer on-time edge polarity
TT_SYNTHESIZER_ON_TIME_EDGE.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_SYNTHESIZER_ON_TIME_EDGE eSynthesizerOnTimeEdge;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the on time edge of the synthesizer to falling
    if (TT_GetSynthesizerOnTimeEdge(hDevice, &eSynthesizerOnTimeEdge) ==
TT_SUCCESS)
    {
        if (eSynthesizerOnTimeEdge == TT_SYNTHESIZER_FALLING)
            // the synthesizer is set to be on time on the falling edge
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_SetSynthesizerOnTimeEdge

TT_GetSynthesizerRunStatus Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetSynthesizerRunStatus
(
  HANDLE *hDevice* **,**
  BOOL* *bRun*
)

Description

This function gets Run/Stop status of the synthesizer

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

bRun
  A flag to Start or Stop the synthesizer.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
BOOL    bRun

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Get the run status of the synthesizer
    if (TT_GetSynthesizerRunStatus(hDevice,&bRun) == TT_SUCCESS)
    {
        // Run status of synthesizer is now available
        ...
    }
}
```

See Also **TT_SetSynthesizerRunStatus TT_GetSynthesizer TT_SetSynthesizer**

TT_GetTimeCompare Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_GetTimeCompare
(
  HANDLE *hDevice* **,**
  FILETIME* *pFileTime* **,**
  TT_TIME_CONVERT *eConvertFlag* **,**
  TT_TIME_COMPARE* *peCompareFlag* **,**
  BOOL* *pbEnableInterrupt*
)

Description

This function returns the current settings for the time compare event.

This function requires **FILE_GENERIC_READ** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

pFileTime
  Address of **FILETIME** structure returning the event time to match.

eConvertFlag
  The **TT_TIME_CONVERT** conversion to apply to the time returned in *pFileTime*.

peCompareFlag
  Address of **TT_TIME_COMPARE** to receive flag indicating the significant digits of *pFileTime* to compare.

pbEnableInterrupt
  Address of **BOOL** to receive current Time Compare interrupt status.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
FILETIME FileTime;
TT_TIME_COMPARE eCompareFlag;
BOOL bEnableInterrupt;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set time compare events
    if (TT_SetTimeCompare(hDevice, &FileTime, TT_CONVERT_NONE,
&eCompareFlag, &bEnableInterrupt) == TT_SUCCESS)
    {
        // time compare events status is available
        ...
    }
}
```

See Also **TT_RegisterCallback TT_SetTimeCompare**

TT_OpenDevice Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_OpenDevice
(
  DWORD *dwDeviceID* **,**
  DWORD *dwDesiredAccess* **,**
  HANDLE* *phDevice*
)

Description Attaches the calling application to the specified TrueTime device. This function must be called by every application using the TrueTime SDK. The parameter *dwDeviceID* is the zero-based ID of the desired board. Board ID's start with the value zero, and increment for each available board. An application can determine the number of boards available by incrementing *dwDeviceID* until the error **TT_STATUS_INVALID_ID** is returned. Multiple device support is only available on Windows NT.

This function performs resource allocation and other initialization operations. Only one application can open a device with **GENERIC_WRITE** access.

Return Value The function returns **TT_STATUS**.

Parameters *dwDeviceID*
  The zero-based ID of the desired device.

dwDesiredAccess
  Desired Access - **GENERIC_READ**, **GENERIC_WRITE**, or both

phDevice
  A poiner to receive the handle to the desired TrueTime device.

Example The following example illustrates the use of this function.

```
HANDLE hDevice

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Open is successful with both Read and Write access
    ...
}
```

See Also **TT_CloseDevice**

TT_PresetPosition Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_PresetPosition
(
  HANDLE *hDevice* **,**

TT_POSITION* *pPosition*
)

Description

This function is used to preset the GPS position. Presetting an initial position will speed up acquisition time by a minute or two.

This function requires **FILE_GENERIC_WRITE** access and is available in GPS mode.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pPosition
Address of a **TT_POSITION** structure to set the current position.

Example

The following example illustrates the use of this function.

```
HANDLE          hDevice;
TT_POSITION     Position;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set position near San Jose, CA
    Position.fLatitude = 35.0;
    Position.fLongitude = -120.0;
    Position.fElevation = 100.0;

    // preset the GPS initial position
    if (TT_PresetPosition(hDevice, &Position) == TT_SUCCESS)
    {
        // The GPS position has been set
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_PresetTime Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_PresetTime
(
HANDLE *hDevice* **,**
FILETIME* *pFileTime* **,**
TT_TIME_CONVERT *eConvertFlag*
)

Description

This function sets the current time in the device, converting the supplied time as specified by *eConvertFlag*. In Timecode mode, this function is used to set the year since year information is not encoded in the time code reference. Year data is necessary to handle end of year rollover correctly for leap years. Year information is saved in EEPROM and automatically increments at the end of each year.

This function requires **FILE_GENERIC_WRITE** access and is available in Generator, 1 PPS, and Timecode modes.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pFileTime
Address of **FILETIME** structure containing the time

eConvertFlag
The **TT_TIME_CONVERT** conversion to apply to the time specified by *pFileTime*.

Example The following example illustrates the use of this function.

```
HANDLE      hDevice;
FILETIME    FileTime;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Set time for the desired device
    if (TT_PresetTime(hDevice, &FileTime, TT_CONVERT_LOCAL2UTC) ==
TT_SUCCESS)
    {
        // The specified time is successfully set as the current time in
the device, converting
        // it from Local Time to UTC
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_ReadDiagnosticRegister Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_ReadDiagnosticRegister
(
HANDLE *hDevice* **,**
TT_DIAG_ERROR* *peDiagnostic* **,**
WORD* *pwDAC*
)

Description This function returns the error and oscillator values from the diagnostic register.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

peDiagnostic
Address of **TT_DIAG_ERROR** to receive hardware error status flags.

pwDAC
Address of location to receive current setting of frequency control DAC.

Example The following example illustrates the use of this function.

```
HANDLE          hDevice;
TT_DIAG_ERROR   Diagnostic;
```

```
WORD            wDAC;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Read the Diagnostic Register for the desired device
    if (TT_ReadDiagnosticRegister(hDevice, &Diagnostic, &wDAC) ==
TT_SUCCESS)
    {
        // The Diagnostic Information is successfully retrieved from the
device
        ...
    }
    // Close the device
    TT_CloseDevice(hDevice);
}
```

TT_ReadGpsInfo Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_ReadGpsInfo
(
   HANDLE *hDevice* **,**
   TT_POSITION* *pPosition* **,**
   TT_GPS_SIGNALS* *pGpsSignals* **,**
   TT_ANTENNA* *pAntenna*
)

Description

This function returns the GPS position consisting of latitude, longitude, and elevation, and satellite signal information for up to six satellites.

This function requires **FILE_GENERIC_READ** access and is only available in GPS mode.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
   A handle to the TrueTime device, returned by **TT_OpenDevice**.

pPosition
   Address of a **TT_POSITION** structure to receive the current position. If this value is **NULL**, no position data is returned.

pGpsSignals
   Address of a **TT_GPS_SIGNALS** structure to receive the current signal data. If this value is **NULL**, no signal data is returned.

pAntenna
   Address of a **TT_ANTENNA** structure to receive the antenna status.

Example

The following example illustrates the use of this function.

```
HANDLE          hDevice;
TT_POSITION     Position;
TT_GPS_SIGNALS  GpsSignals;
TT_ANTENNA      Antenna;

// open device 0 for Read/Write access
```

```
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Read the GPS Information for the desired device
    if (TT_ReadGpsInfo(hDevice, &Position, &GpsSignals, &Antenna) ==
TT_SUCCESS)
    {
        // The GPS information is successfully retrieved from the device
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_ReadTime Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_ReadTime
(
  HANDLE *hDevice* **,**
  FILETIME* *pFileTime* **,**
  TT_TIME_CONVERT *eConvertFlag*
)

Description

This function retreives the current time value from the device, converting it as specified by *bLocalTime*. In GPS mode, time is always maintained in UTC. In other modes, time is application specific. This function requires **FILE_GENERIC_READ** access.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

pFileTime
  Address of **FILETIME** structure to receive the time

eConvertFlag
  The **TT_TIME_CONVERT** conversion to apply to the time read from the clock and returned in *pFileTime*.

Example

The following example illustrates the use of this function.

```
HANDLE      hDevice;
FILETIME    FileTime;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Read the current Freeze Time for the desired device
    if (TT_ReadTime(hDevice, &FileTime, TT_CONVERT_UTC2LOCAL) ==
TT_SUCCESS)
    {
        // The current freeze time is successfully read from the device
and
        // converted it from UTC to Local time.
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

TT_ReadTimecodeInfo Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_ReadTimecodeInfo
(
  HANDLE *hDevice* **,**
  BOOL* *bLocked* **,**
  BOOL* *bValid*
)

Description This function reads the Locked/Valid status for the TimeCode Input

This function requires **FILE_GENERIC_READ** access and is available in timecode mode.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

bLocked
  Address of **BOOL** to receive TimeCode Locked status.

bValid
  Address of **BOOL** to receive TimeCode Valid status.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;
BOOL    bLocked;
BOOL    bValid;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // get timecode information
    if (TT_ReadTimecodeInfo(hDevice, &bLocked, &bValid) == TT_SUCCESS)
    {
        // the timecode information is available
        ...
    }
}
```

TT_RegisterCallback Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_RegisterCallback
(
  HANDLE *hDevice* **,**

TT_CALLBACK *pCallback* **,**
PVOID *pContext*
)

Description

This function registers a callback routine to be executed upon external, periodic, or time compare events.

This function requires **FILE_GENERIC_WRITE** access and is available in all modes. The calling application must unregister the callback before exit.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pCallback
Address of a **TT_CALLBACK** routine to call upon hardware events. A value of **NULL** cancels callbacks.

pContext
Address of user-defined context, passed to callback routine.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
TT_STATUS (MyCallback) (TT_EVENT, FILETIME*, DWORD, PVOID);

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set callback address
    if (TT_RegisterCallback(hDevice, MyCallback, NULL) == TT_SUCCESS)
    {
        // The callback address has been set
        ...
    }
}
```

See Also

TT_Callback

TT_SetExternalEvent Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetExternalEvent
(
HANDLE *hDevice* **,**
BOOL *bEnable*
)

Description

This function enables/disables the external input event. If a callback has been registered (**TT_RegisterCallback**), it will be called at the external event.

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

bEnable
If **TRUE**, the external event input is enabled, and the registered callback is called upon occurrance.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set external event
    if (TT_SetExternalEvent(hDevice, TRUE) == TT_SUCCESS)
    {
        // external events are enabled
        ...
    }
}
```

See Also

TT_RegisterCallback TT_GetExternalEvent

TT_SetExternalEventTriggerEdge Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetExternalEventTriggerEdge
(
HANDLE *hDevice* **,**
TT_EXTERNAL_EVENT_TRIGGER_EDGE *eExternalEventTriggerEdge*
)

Description

This function sets the trigger of the external event to rising or falling This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eExternalEventTriggerEdge
The external event trigger polarity (see **TT_EXTERNAL_EVENT_TRIGGER_EDGE**).

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the trigger to falling
    if (TT_SetExternalEventEdge(hDevice, TT_FALLING) == TT_SUCCESS)
    {
        // the external event if now set to trigger on the falling edge
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_GetExternalEventTriggerEdge**

TT_SetExternalEventTriggerSource Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetExternalEventTriggerSource
(
    HANDLE *hDevice* **,**
    TT_EXTERNAL_EVENT_TRIGGER_SOURCE
eExternalEventTriggerSource
)

Description This function sets the source of the trigger for the external event This allows more accurate real time triggers to freeze the time and removes the PCI buss latency inherent in a software freeze. This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eExternalEventTriggerSource
The external event trigger source **TT_EXTERNAL_EVENT_TRIGGER_SOURCE**.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the trigger source to the synthesizer
    if (TT_SetExternalEventSource(hDevice, TT_SYNTHESIZER) ==
TT_SUCCESS)
    {
        // the external event will now be triggered by the synthesizer
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_GetExternalEventTriggerSource**

TT_SetLeapSecond Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetLeapSecond
(
    HANDLE *hDevice* **,**

BOOL *bEnable*
)

Description This function enables/disables the hardware to add a leap second at the end of the current day. This function requires **FILE_GENERIC_WRITE** access and is available in 1 PPS mode.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

bEnable
Flag to enable/disable the hardware

Example The following example illustrates the use of this function.

```
HANDLE hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the leap second flag
    if (TT_SetLeapSecond(hDevice, TRUE) == TT_SUCCESS)
    {
        // the leap second flag is set
        ...
    }
}
```

See Also **TT_GetLeapSecond**

TT_SetMode Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetMode
(
HANDLE *hDevice* **,**
TT_OPERATION_MODE *eOperationMode* **,**
TT_SYNCH_SOURCE *eSynchSource* **,**
TT_TIMECODE *eTimeCode*
)

Description This function sets the current operating mode, which is maintained by the hardware in non-volatile memory. Since the hardware is pre-configured by the factory, it is not necessary for any application to call this function. However, it is available for situations where the factory settings must be changed. This function requires **FILE_GENERIC_WRITE** access.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eOperationMode
The desired operation mode (see **TT_OPERATION_MODE**).

eSynchSource
When *eOperationMode* is **TT_MODE_SYNCHRONIZED**, *eSynchSource* specifies the desired synchronization source (see **TT_SYNCH_SOURCE**).

eTimeCode
When *eSynchSource* is **TT_SYNCH_TIMECODE**, *eTimeCode* specifies the desired timecode standard (see **TT_TIMECODE**).

Example The following example illustrates the use of this function.

```
HANDLE             hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Set the Current Operating Mode for the desired device
    if (TT_SetMode(hDevice, TT_MODE_SYNCHRONIZED, TT_SYNCH_TIMECODE,
TT_TIMECODE_IRIGA_DC) == TT_SUCCESS)
    {
        // The device is successfully synchronized to an external
timecode source of
        // IRIG-A, in DC mode
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_GetMode**

TT_SetOutputBNCSource Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetOutputBNCSource
(
HANDLE *hDevice* **,**
TT_OUTPUT_BNC_SOURCE *eOutputBncSource*
)

Description This function sets the source that will be available on the output BNC This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eOutputBncSource
The source setting for the output BNC as specified in **TT_OUTPUT_BNC_SOURCE**.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
```

```
{
    // Select the sunthesizer to go out of the output BNC
    if (TT_SetOutputBNCSource(hDevice,TT_OUTPUT_SYNTHESIZER) ==
TT_SUCCESS)
    {
        // the synthesizer is now being output at the output BNC
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_GetOutputBNCSource**

TT_SetPhaseCompensation Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetPhaseCompensation
(
  HANDLE *hDevice* **,**
  SHORT *lOffset*
)

Description Sets the phase compensation for the device. This function requires **FILE_GENERIC_WRITE** access, and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
  A handle to the TrueTime device, returned by **TT_OpenDevice**. LONG lOffset // @parm Specifies the desired compensation from -1000 to +1000 microseconds.

lOffset
  Specifies the desired compensation from -1000 to +1000 microseconds.

Example The following example illustrates the use of this function.

```
HANDLE hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set phase compensation
    if (TT_SetPhaseCompensation(hDevice, 340) == TT_SUCCESS)
    {
        // the phase compensation is set to +340 microseconds
        ...
    }
}
```

See Also **TT_GetPhaseCompensation**

TT_SetRateGenerator Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetRateGenerator
(
  HANDLE *hDevice* **,**
  TT_GENERATOR_RATE *eRate* **,**
  BOOL *bEnableInterrupt*
)

Description

This function enables/disables the rate generator event. If a callback has been registered (**TT_RegisterCallback**), it will be called at the requested rate.

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eRate
The rate at which pulses should be generated (see **TT_GENERATOR_RATE**). A value of **TT_RATE_DISABLE** will disable pulse generation.

bEnableInterrupt
A flag to enable or disable the rate generator interrupt. The rate generator interupt will be automatically turned off if interrupts arrive faster than the system can process them.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set rate generator events
    if (TT_SetRateGenerator(hDevice, TT_RATE_10PPS, TRUE) == TT_SUCCESS)
    {
        // 10 PPS rate generator events are enabled
        ...
    }
}
```

See Also

TT_RegisterCallback TT_GetRateGenerator

TT_SetRegister Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetRegister
(
  HANDLE *hDevice* **,**

ULONG *ulRegisterOffset* **,**
UCHAR *ucRegisterValue*
)

Description

This function sets the contents of the register at the address specified. This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

ulRegisterOffset
The address of the register to write.

ucRegisterValue
The address of the byte to write to the register.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
ULONG   RegisterOffset;
UCHAR   RegisterValue;


// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the register data
    if (TT_SetRegister(hDevice, RegisterOffset, RegisterValue) ==
TT_SUCCESS)
    {
        // the register has now been set
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also

TT_GetRegister

TT_SetSynthesizer Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetSynthesizer
(
HANDLE *hDevice* **,**
DWORD *dwFrequency* **,**
BOOL *bEnableInterrupt*
)

Description

This function sets the frequency and enables/disables the synthesizer event. If a callback has been registered (**TT_RegisterCallback**), it will be called at the requested frequency.

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

dwFrequency
The frequency at which pulses should be generated. A value of 0 will disable pulse generation.

bEnableInterrupt
A flag to enable or disable the synthesizer interrupt. The synthesizer interupt will be automatically turned off if interrupts arrive faster than the system can process them.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set synthesizer settings
    if (TT_SetSynthesizer(hDevice, 1000, TRUE) == TT_SUCCESS)
    {
        // Synthesizer is set to 1000 Hz and events are enabled
        ...
    }
}
```

See Also **TT_RegisterCallback TT_GetSynthesizer TT_SetSynthesizerRunStatus**

TT_SetSynthesizerOnTimeEdge Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetSynthesizerOnTimeEdge
(
HANDLE *hDevice* **,**
TT_SYNTHESIZER_ON_TIME_EDGE *eSynthesizerOnTimeEdge*
)

Description This function sets the polarity of the on time edge of the synthesizer This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

eSynthesizerOnTimeEdge
The syntesizer on-time edge polarity
TT_SYNTHESIZER_ON_TIME_EDGE.

Example The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // set the on time edge of the synthesizer to falling
    if (TT_SetSynthesizerOnTimeEdge(hDevice,TT_SYNTHESIZER_FALLING) ==
TT_SUCCESS)
    {
        // the synthesizer will now be on time on the falling edge
        ...
    }
    // close the device
    TT_CloseDevice(hDevice);
}
```

See Also **TT_GetSynthesizerOnTimeEdge**

TT_SetSynthesizerRunStatus Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetSynthesizerRunStatus
(
  HANDLE *hDevice* **,**
  BOOL *bRun*
)

Description

This function sets Run/Stop status of the synthesizer

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

bRun
  A flag to Start or Stop the synthesizer.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // Stop the synthesizer
    if (TT_SetSynthesizerRunStatus(hDevice,FALSE) == TT_SUCCESS)
    {
        // Synthesizer is stopped
        ...
    }
}
```

See Also **TT_GetSynthesizerRunStatus TT_GetSynthesizer TT_SetSynthesizer**

TT_SetTimeCompare Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SetTimeCompare
(
HANDLE *hDevice* **,**
FILETIME* *pFileTime* **,**
TT_TIME_CONVERT *eConvertFlag* **,**
TT_TIME_COMPARE *eCompareFlag* **,**
BOOL *bEnableInterrupt*
)

Description

This function enables/disables the time compare event. If a callback has been registered (**TT_RegisterCallback**), it will be called at the requested times.

This function requires **FILE_GENERIC_WRITE** access and is available in all modes.

Return Value

The function returns **TT_STATUS**.

Parameters

hDevice
A handle to the TrueTime device, returned by **TT_OpenDevice**.

pFileTime
Address of **FILETIME** structure containing the event time to match.

eConvertFlag
The **TT_TIME_CONVERT** conversion to apply to the time specified by *pFileTime*.

eCompareFlag
Flag indicating the significant digits of *pFileTime* to compare see (**TT_TIME_COMPARE**).

bEnableInterrupt
A flag to enable or disable the Time Compare interrupt.

Example

The following example illustrates the use of this function.

```
HANDLE  hDevice;
FILETIME FileTime;
BOOL bEnableInterrupt = TRUE;

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // initialize Filetime structure
    FileTime ...;

    // set time compare events
    if (TT_SetTimeCompare(hDevice, &FileTime, TT_CONVERT_NONE,
TT_TIME_COMPARE_THR, bEnableInterrupt) == TT_SUCCESS)
    {
        // time compare events are enabled
        ...
    }
}
```

See Also

TT_RegisterCallback TT_GetTimeCompare

TT_StartGenerator Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_StartGenerator
(
  HANDLE *hDevice*
)

Description This function enables the device to accumulate time. This function requires **FILE_GENERIC_WRITE** access and is only available in Generator mode.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

Example The following example illustrates the use of this function.

```
HANDLE hDevice

// open device 0 for Read/Write access
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // start the device
    if (TT_StartGenerator(hDevice) == TT_SUCCESS)
    {
        // the device is running
        ...
    }
}
```

See Also **TT_StopGenerator**

TT_StopGenerator Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_StopGenerator
(
  HANDLE *hDevice*
)

Description This function stops time accumulation of the device. This function requires **FILE_GENERIC_WRITE** access and is only available in Generator mode.

Return Value The function returns **TT_STATUS**.

Parameters *hDevice*
  A handle to the TrueTime device, returned by **TT_OpenDevice**.

Example The following example illustrates the use of this function.

```
HANDLE hDevice

// open device 0 for Read/Write access
```

```
if (TT_OpenDevice(0, GENERIC_READ | GENERIC_WRITE, &hDevice) ==
TT_SUCCESS)
{
    // stop the device
    if (TT_StopGenerator(hDevice) == TT_SUCCESS)
    {
        // the device is stopped
        ...
    }
}
```

See Also **TT_StartGenerator**

TT_SystemTimeExToFileTime Function

Defined in: TrueTimeSDK.h

TT_API TT_STATUS TT_SystemTimeExToFileTime
(
  SYSTEMTIME_EX* *lpSystemTime* **,**
  FILETIME* *lpFileTime*
)

Description This function converts a system time to 64-bit file time format. The *wDayOfWeek* member of the **SYSTEMTIME_EX** structure is ignored.

Return Value If the function fails, the return value is zero. To get extended error information, call GetLastError.

Parameters *lpSystemTime*
  Pointer to a **SYSTEMTIME_EX** structure to converted.

lpFileTime
  Pointer to a **FILETIME** structure to receive the converted file time.

Example The following example illustrates the use of this function.

```
FILETIME        FileTime;
SYSTEMTIME_EX   SystemTime;

// Convert the SystemTimeEx to FileTime format
if (TT_SystemTimeExToFileTime(&SystemTime, &FileTime) == TT_SUCCESS)
{
    // SystemTimeEx is successfully converted to the FileTime
    ...
}
```

FILETIME Structure

Defined in: TrueTimeSDK.h

```
typedef struct
{
    DWORD dwLowDateTime;
    DWORD dwHighDateTime;
} FILETIME;
```

Description

A 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601. A FILETIME structure can represent time values of approximately 29,000 years.

Members

dwLowDateTime
Specifies the low-order 32 bits of the file time.

dwHighDateTime
Specifies the high-order 32 bits of the file time.

Comments

The FILETIME structure is compatible with the **LARGE_INTEGER** structure. Therefore, to perform arithmetic on FILETIME data, convert it to a LARGE_INTEGER structure.

FILETIME is a standard Win32 time format. See **About Windows Time** for more information.

SYSTEMTIME_EX Structure

Defined in: TrueTimeCmn.h

```
typedef struct
{
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
    WORD wMicroseconds;
    WORD wNanoseconds;
} SYSTEMTIME_EX;
```

Description

The SYSTEMTIME_EX structure represents a date and time using individual members for the month, day, year, weekday, hour, minute, second, millisecond, microsecond, and nanosecond. The first 8 WORDS of this structure are identical to, and therefore interchangable with, the Win32 **SYSTEMTIME** structure.

Members

wYear
Specifies the current year.

wMonth
Specifies the current month; January = 1, February = 2, and so on.

wDayOfWeek
Specifies the current day of the week; Sunday = 0, Monday = 1, and so on.

wDay
Specifies the current day of the month.

wHour
Specifies the current hour.

wMinute
Specifies the current minute.

wSecond
Specifies the current second.

wMilliseconds
Specifies the current millisecond.

wMicroseconds
Specifies the current microsecond.

wNanoseconds
Specifies the current nanosecond.

Comments It is not recommended that you add and subtract values from the SYSTEMTIME_EX structure to obtain relative times. Instead, you should convert the SYSTEMTIME_EX structure to a FILETIME structure and use normal 64-bit arithmetic on the LARGE_INTEGER value.

TT_ANTENNA Structure

Defined in: TrueTimeCmn.h

```
typedef struct
{
    BOOL bShort;
    BOOL bOpen;
} TT_ANTENNA;
```

Description Satellite Antenna status.

Members *bShort*
Flag to indicate if Antenna Shorted

bOpen
Flag to indicate if Antenna Open

TT_CONFIGURATION_SETTINGS

Defined in: TrueTimeCmn.h

```
enum TT_CONFIGURATION_SETTINGS {
    TT_USE_TIME_QUALITY,
    TT_SAVE_DAC,
    TT_SAVE_CURRENT_CONFIGURATION,
    TT_RESTORE_SAVED_SETTINGS,
    TT_RESTORE_FACTORY_DEFAULTS
};
```

Description Values specifying the configuration functions.

Members *TT_USE_TIME_QUALITY*
Specifies to use the time quality flags to determine signal validity.

TT_SAVE_DAC
Specifies to save the DAC setings in eeprom.

TT_SAVE_CURRENT_CONFIGURATION
Specifies to save the boards settings in eeprom.

TT_RESTORE_SAVED_SETTINGS
Specifies to restore the board settings saved in eeprom.

TT_RESTORE_FACTORY_DEFAULTS
Specifies to set the board to its factory defaults.

TT_DIAG_ERROR Structure

Defined in: TrueTimeCmn.h

```
typedef struct
{
    BOOL bClockError:1;
    BOOL bRamError:1;
    BOOL bDacLimit:1;
    BOOL bHardwareError:1;
} TT_DIAG_ERROR;
```

Description Hardware diagnostic register.

Members

bClockError:1
Indicates processor clock failure.

bRamError:1
Indicates onboard RAM failure.

bDacLimit:1
Indicates DAC setting near the limit.

bHardwareError:1
Indicates general hardware error.

TT_EVENT

Defined in: TrueTimeCmn.h

```
enum TT_EVENT {
    TT_EVENT_EXTERNAL,
    TT_EVENT_PERIODIC,
    TT_EVENT_TIME_COMPARE,
    TT_EVENT_SYNTHESIZER
};
```

Description Values specifying an event.

Members

TT_EVENT_EXTERNAL
The event is a pulse input on pin 1 of the 9-pin connector. These events can occur at a maximum rate of 100 pulses per second, and must be at least 3 ms apart.

TT_EVENT_PERIODIC
The event is a pulse output on pin 7 of the 9-pin connector with one of five fixed rates. The signal is synchronous with the board timing and the rising edge is on time.

TT_EVENT_TIME_COMPARE
The event is a pulse output on pin 9 of the 9-pin connector at a preset time. The rising edge of this pulse is on-time.

TT_EVENT_SYNTHESIZER
The event is a pulse output on IRIG OUT of the BNC connector The signal is synchronous with the board timing and the trailing edge is on time.

TT_EXTERNAL_EVENT_TRIGGER_EDGE

Defined in: TrueTimeCmn.h

```
enum TT_EXTERNAL_EVENT_TRIGGER_EDGE {
    TT_FALLING,
    TT_RISING
};
```

Description Values specifying the external event trigger polarity.

Members

TT_FALLING
Specifies external event falling trigger.

TT_RISING
Specifies external event rising trigger.

TT_EXTERNAL_EVENT_TRIGGER_SOURCE

Defined in: TrueTimeCmn.h

```
enum TT_EXTERNAL_EVENT_TRIGGER_SOURCE {
    TT_EXTERNAL_EVENT_TRIGGER,
    TT_SYNTHESIZER_TRIGGER,
    TT_RATE_GENERATOR_TRIGGER,
    TT_TIME_COMPARE_TRIGGER
};
```

Description Values specifying the external event trigger source.

Members

TT_EXTERNAL_EVENT_TRIGGER
Specifies to trigger the external event with the external trigger.

TT_SYNTHESIZER_TRIGGER
Specifies to trigger the external event with the synthesizer.

TT_RATE_GENERATOR_TRIGGER
Specifies to trigger the external event with the rate generator.

TT_TIME_COMPARE_TRIGGER
Specifies to trigger the external event with the time compare.

TT_GENERATOR_RATE

Defined in: TrueTimeCmn.h

```
enum TT_GENERATOR_RATE {
    TT_RATE_DISABLE,
    TT_RATE_10KPPS,
    TT_RATE_1KPPS,
    TT_RATE_100PPS,
    TT_RATE_10PPS,
    TT_RATE_1PPS,
    TT_RATE_100KPPS,
    TT_RATE_1MPPS,
    TT_RATE_5MPPS,
    TT_RATE_10MPPS
};
```

Description Values specifying the rate of generated output pulses.

Members

TT_RATE_DISABLE
Disable pulse generation.

TT_RATE_10KPPS
Generate 10,000 pulses per second.

TT_RATE_1KPPS
Generate 1,000 pulses per second.

TT_RATE_100PPS
Generate 100 pulses per second.

TT_RATE_10PPS
Generate 10 pulses per second.

TT_RATE_1PPS
Generate 1 pulse per second.

TT_RATE_100KPPS
Generate 100,000 pulses per second.

TT_RATE_1MPPS
Generate 1,000,000 pulses per second.

TT_RATE_5MPPS
Generate 5,000,000 pulses per second.

TT_RATE_10MPPS
Generate 10,000,000 pulses per second.

TT_GPS_SIGNAL Structure

Defined in: TrueTimeSDK.h

```
typedef struct
{
    DWORD dwPRN;
    float fSignalStrength;
} TT_GPS_SIGNAL;
```

Description GPS satellite signal strength.

Members

dwPRN
Satellite ID.

fSignalStrength
Signal strength after correlation or de-spreading.

TT_GPS_SIGNALS Structure

Defined in: TrueTimeSDK.h

```
typedef struct
{
    DWORD dwCount;
    TT_GPS_SIGNAL satellite[TT_MAX_SATELLITES];
    BOOL bGPSLock;
} TT_GPS_SIGNALS;
```

Description Signal strength data for up to six satellites.

Members

dwCount
Number of satellites acquired and locked.

satellite[TT_MAX_SATELLITES]
Signal strength data (see **TT_GPS_SIGNAL**).

bGPSLock
Flag to indicate the GPS Lock/Unlock Status

TT_HARDWARE_STATUS Structure

Defined in: TrueTimeCmn.h

```
typedef struct
{
    BOOL AntennaPositionReady;
    BOOL SoftwareTimeRequestReady;
    BOOL AntennaShorted;
    BOOL AntennaOpen;
    BOOL SynthesizerPulseOccured;
    BOOL RateGeneratorPulseOccured;
    BOOL TimeComparePulseOccured;
    BOOL ExternalEventPulseOccured;
} TT_HARDWARE_STATUS;
```

Description Hardware status.

Members

AntennaPositionReady
Flag to indicate position data is valid

SoftwareTimeRequestReady
Flag to indicate freeze time data is valid

AntennaShorted
Flag to indicate the GPS antenna is shorted

AntennaOpen
Flag to indicate the GPS antenna is open

SynthesizerPulseOccured
Flag to indicate a pulse occured on the synthesizer

RateGeneratorPulseOccured
Flag to indicate a pulse occured on the rate generator

TimeComparePulseOccured
Flag to indicate a pulse occured on the time compare

ExternalEventPulseOccured
Flag to indicate a pulse occured on the external event

TT_MODEL

Defined in: TrueTimeCmn.h

```
enum TT_MODEL {
    TT_MODEL_5900,
    TT_MODEL_5901,
    TT_MODEL_5905,
    TT_MODEL_5906,
    TT_MODEL_5907,
    TT_MODEL_5908,
    TT_MODEL_5950,
    TT_MODEL_5951
};
```

Description Values specifying the model of the device.

Members

TT_MODEL_5900
Model 5900 - PCI bus time card

TT_MODEL_5901
Model 5901 - PCI bus time card with GPS receiver

TT_MODEL_5905
Model 5905 - PCI bus time card with Synthesizer

TT_MODEL_5906
Model 5906 - PCI bus time card with Synthesizer and GPS receiver

TT_MODEL_5907
Model 5907 - PCI2 bus time card

TT_MODEL_5908
Model 5908 - PCI2 bus time card with GPS receiver

TT_MODEL_5950
Model 5950 - Compact PCI bus time card

TT_MODEL_5951
Model 5951 - Compact PCI bus time card with GPS receiver

TT_OPERATION_MODE

Defined in: TrueTimeCmn.h

```
enum TT_OPERATION_MODE {
    TT_MODE_GENERATOR,
    TT_MODE_SYNCHRONIZED
};
```

Description Values specifying the operational mode of the device.

Members

TT_MODE_GENERATOR
The device is running on its internal oscillator

TT_MODE_SYNCHRONIZED
The device is synchronized to an external source

TT_OUTPUT_BNC_SOURCE

Defined in: TrueTimeCmn.h

```
enum TT_OUTPUT_BNC_SOURCE {
    TT_OUTPUT_IRIG_AM_TIMECODE,
    TT_OUTPUT_IRIG_DC_TIMECODE,
    TT_OUTPUT_RATE_GENERATOR,
    TT_OUTPUT_SYNTHESIZER,
    TT_OUTPUT_TIME_COMPARE,
    TT_OUTPUT_1PPS
};
```

Description Values specifying the source to appear on the output BNC.

Members

TT_OUTPUT_IRIG_AM_TIMECODE
Specifies to send the AM timecode to the output BNC.

TT_OUTPUT_IRIG_DC_TIMECODE
Specifies to send the DC timecode to the output BNC.

TT_OUTPUT_RATE_GENERATOR
Specifies to send the Rate Generator to the output BNC.

TT_OUTPUT_SYNTHESIZER
Specifies to send the synthesizer to the output BNC.

TT_OUTPUT_TIME_COMPARE
Specifies to send the Time compare pulse to the output BNC.

TT_OUTPUT_1PPS
Specifies to send the 1 pps to the output BNC.

TT_POSITION Structure

Defined in: TrueTimeSDK.h

```
typedef struct
{
    float fLatitude;
    float fLongitude;
    float fElevation;
} TT_POSITION;
```

Description Position data consisting of latitude, longitude, and elevation.

Members

fLatitude
Latitude in degrees, from -90 (south) to +90 (north).

fLongitude
Longitude in degrees, from -180 (east) to +180 (west).

fElevation
Elevation in meters, above and below sea level.

TT_STATUS

Defined in: TrueTimeSDK.h

```
enum TT_STATUS {
    TT_SUCCESS,
    TT_STATUS_INVALID_ID,
    TT_ERROR_SERVICE_MANAGER,
    TT_INVALID_HANDLE,
    TT_INVALID_ACCESS_REQUESTED,
    TT_INVALID_ACCESS,
    TT_INVALID_MODE,
    TT_FAIL_UTCTOLOCAL,
    TT_FAIL_LOCALTOUTC,
    TT_SYSTEM_ERROR,
    TT_MODE_NOT_SUPPORTED,
    TT_FAILED_TO_CREATE_THREAD,
    TT_ANTENNA_ERROR,
    TT_GPS_SIGNAL_INFO_NOT_AVAILABLE,
    TT_TIMEOUT_ERROR
};
```

Description Status values returned by the SDK.

Members

TT_SUCCESS
The requested operation was successful

TT_STATUS_INVALID_ID
The requested device doesn't exist

TT_ERROR_SERVICE_MANAGER
Service Manager failed loading driver.

TT_INVALID_HANDLE
The Invalid device handle

TT_INVALID_ACCESS_REQUESTED
The Invalid access requested

TT_INVALID_ACCESS
The process doesn't have valid access for this operation

TT_INVALID_MODE
The Device doesn't have valid Mode set for this operation

TT_FAIL_UTCTOLOCAL
Conversion from UTC to Local File Time failed

TT_FAIL_LOCALTOUTC
Conversion from Local File Time to UTC failed

TT_SYSTEM_ERROR
A system error occured, call GetLastError for details.

TT_MODE_NOT_SUPPORTED
This mode is not supported by the device

TT_FAILED_TO_CREATE_THREAD
An error occurred creating the callback thread.

TT_ANTENNA_ERROR
An antenna is Open/Shorted.

TT_GPS_SIGNAL_INFO_NOT_AVAILABLE
The GPS Signal Info not available.

TT_TIMEOUT_ERROR
A function call has timed out.

TT_SYNCH_SOURCE

Defined in: TrueTimeCmn.h

```
enum TT_SYNCH_SOURCE {
    TT_SYNCH_1PPS,
    TT_SYNCH_GPS,
    TT_SYNCH_TIMECODE
};
```

Description Values specifying a synchronization source.

Members

TT_SYNCH_1PPS
The synchronization signal is a 1 pulse per second input.

TT_SYNCH_GPS
The synchronization signal is a global positioning system receiver.

TT_SYNCH_TIMECODE
The synchronization signal is a timecode source.

TT_SYNTHESIZER_ON_TIME_EDGE

Defined in: TrueTimeCmn.h

```
enum TT_SYNTHESIZER_ON_TIME_EDGE {
    TT_SYNTHESIZER_FALLING,
    TT_SYNTHESIZER_RISING
};
```

Description Values specifying the synthesizer ontime edge polarity.

Members

TT_SYNTHESIZER_FALLING
Specifies the synthesizer to be ontime on the falling edge.

TT_SYNTHESIZER_RISING
Specifies the synthesizer to be ontime on the rising edge.

TT_TIME_COMPARE

Defined in: TrueTimeCmn.h

```
enum TT_TIME_COMPARE {
    TT_TIME_COMPARE_ALL,
    TT_TIME_COMPARE_TDAY,
    TT_TIME_COMPARE_UDAY,
    TT_TIME_COMPARE_THR,
    TT_TIME_COMPARE_UHR,
    TT_TIME_COMPARE_TMIN,
    TT_TIME_COMPARE_UMIN,
    TT_TIME_COMPARE_TSEC,
    TT_TIME_COMPARE_USEC,
    TT_TIME_COMPARE_HMS,
    TT_TIME_COMPARE_TMS,
    TT_TIME_COMPARE_UMS,
    TT_TIME_COMPARE_DISABLE
};
```

Description Values specifying the significant digits of a time comparison.

Members

TT_TIME_COMPARE_ALL
Compare all digits.

TT_TIME_COMPARE_TDAY
Compare through tens of days.

TT_TIME_COMPARE_UDAY
Compare through units of days.

TT_TIME_COMPARE_THR
Compare through tens of hours.

TT_TIME_COMPARE_UHR
Compare through units of hours.

TT_TIME_COMPARE_TMIN
Compare through tens of minutes.

TT_TIME_COMPARE_UMIN
Compare through units of minutes.

TT_TIME_COMPARE_TSEC
Compare through tens of seconds.

TT_TIME_COMPARE_USEC
Compare through units of seconds.

TT_TIME_COMPARE_HMS
Compare through hundreds of milliseconds.

TT_TIME_COMPARE_TMS
Compare through tens of milliseconds.

TT_TIME_COMPARE_UMS
Compare through units of milliseconds.

TT_TIME_COMPARE_DISABLE
Disable the Compare event.

TT_TIME_CONVERT

Defined in: TrueTimeSDK.h

```
enum TT_TIME_CONVERT {
    TT_CONVERT_NONE,
    TT_CONVERT_UTC2LOCAL,
    TT_CONVERT_LOCAL2UTC
};
```

Description Values specifying conversion between UTC and local time.

Members

TT_CONVERT_NONE
No conversion.

TT_CONVERT_UTC2LOCAL
Convert from UTC to local time, using the system **TIME_ZONE_INFORMATION** values.

TT_CONVERT_LOCAL2UTC
Convert from local to UTC time, using the system **TIME_ZONE_INFORMATION** values.

TT_TIMECODE

Defined in: TrueTimeCmn.h

```
enum TT_TIMECODE {
    TT_TIMECODE_IRIGA_DC,
    TT_TIMECODE_IRIGA_AM,
    TT_TIMECODE_IRIGB_DC,
    TT_TIMECODE_IRIGB_AM
};
```

Description Values specifying a timecode standard.

Members

TT_TIMECODE_IRIGA_DC
The timecode is IRIG-A, in DC mode.

TT_TIMECODE_IRIGA_AM
The timecode is IRIG-A, in AM mode.

TT_TIMECODE_IRIGB_DC
The timecode is IRIG-B, in DC mode.

TT_TIMECODE_IRIGB_AM
The timecode is IRIG-B, in AM mode.