

TCG-100 API Documentation

document version 1.0

Copyright © 2000 W Clark & Associates, Ltd.

Introduction

TCG DLL is an application programming interface to the TCG-100 time code reader card kernel-mode driver for Windows NT. TCG DLL requires an installed and correctly functioning base version of the TCG-100 Windows NT software and drivers. TCG DLL does not require TCGSYNC-NT software to operate.

TCG DLL supports SMTPE time code (all formats) and IRIG-B(1). Full control and monitoring of generation features are available. TCG DLL also provides an asynchronous notification mechanism where by a user-supplied callback function is called when significant events occur during time code generation. The time provided with such events is time stamped with the Windows NT high-performance timer as well as system time/date at the time the event occurs. These timestamps can be used to derive very precise real-time information.

TCG DLL can support (currently) up to four TCG-100 devices operating simultaneously in a single PC computer. *TCG DLL API library requires at least TCG-100 firmware version 2.0.*

Legal Information

The information contained in this document is subject to change without notice. W Clark & Associates, Ltd. (hereinafter W C & A) provides the TCG DLL development software package on an as-is basis. W C & A makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. W C&A shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Technical Support

Technical support is not provided, except on a custom basis. General comments or bug reports concerning TCG DLL development package may be addressed, via Internet e-mail, to <support@masterclock.com>.

Installation Instructions

A Visual C++ application will link to TCGDLL.LIB and will require TCGDLL.DLL to be in the system PATH environment variable at run-time. A Visual Basic application does not need to link to TCGDLL.LIB but will still require TCGDLL.DLL to be in the path at run time. Any modules of a Visual C++ using API functions in TCG DLL should include TCGAPI.H. Equivalent definitions for Visual Basic programs are provided in the documentation under Appendix D.

It is a good idea to familiarize yourself with the VCDemo demonstration utility provided with the API library. This utility, written in Visual C++, demonstrates all of the API calls/features and can also be used to diagnose problems with a TCG-100 installation. Do not begin development of a custom project until VCDemo has been observed to perform all API's successfully on a given computer installation.

TCG-100 API AND TCGSYNC-NT SOFTWARE

The TCG-100 API does not provide control or interrogation services for the TCGSYNC-NT time synchronization software package. If developing a custom application for a TCG-100 device that is also being controlled by TCGSYNC-NT keep in mind that exercising manual control of the TCG-100 may negatively impact the performance of or even break desired operation of TCGSYNC-NT software.

Furthermore, TCGSYNC-NT may elect to change the operation of the TCG-100 device without the knowledge of your application.

TCG Interface DLL API Function Reference

TCGGetDLLInitStatus

The **TCGGetDLLInitStatus** API returns the DLL internal initialization status.

DWORD TCGGetDLLInitStatus(void)

Parameters

This function has no parameters.

Return Values

The return value is one of TCGDLL_INIT_XXX defined in TCGAPI.H:

TCGDLL_INIT_OK	The DLL initialized successfully.
TCGDLL_INIT_OSISWIN95	The DLL is being used under Windows 95 which is not supported.
TCGDLL_INIT_NOKMD	The TCG-100 kernel-mode driver is not loaded. See software troubleshooting section of TCG-100 manual.
TCGDLL_INIT_SOMEFAILED	Some installed TCG-100 devices did not initialize successfully. Use TCGGetDeviceInit() to find out which devices failed to initialize and why.
TCGDLL_INIT_DRIVEROLD	The API requires at least TCG-100 kernel-mode driver version 2.0. Install the latest version of the base TCG-100 software for Windows NT.
TCGDLL_INIT_UNEXPECTED	Unexpected initialization failure.

Remarks

The DLL initializes an instance of itself on behalf of each calling process. Application should always call this function to determine the DLL initialization status before using any other API functions.

TCGGetDeviceInit

**DWORD TCGGetDeviceInit(
 DWORD *device***

);

The **TCGGetDeviceInit** API returns the initialization status of an individual TCG-100 device.

Parameters

device

The number of the TCG-100 device to interrogate.

Return Values

The return value is one of DEVICE_INIT_XXX defined in TCGAPI.H, currently:

DEVICE_INIT_OK	The device initialized successfully.
DEVICE_INIT_CFGERROR	The device's registry configuration is corrupt or invalid.
DEVICE_INIT_NOTPRESENT	The kernel-mode driver was unable to find the device during startup and therefore can provide no services to the API for this device.

Returns Win32 error code ERROR_NOT_SUPPORTED when API called for an invalid/unsupported TCG-100 device number.

Remarks

The TCG-100 kernel-mode driver and API software can support, for simultaneous control/operation, up to MAX_TCG_DEVICES (defined in TCGAPI.H). Use this API to determine if a registry-defined TCG-100 device was detected/initialized properly.

TCGGetTimeCodeType

```
DWORD TCGGetTimeCodeType(
    DWORD device,
    DWORD *tcType,
    DWORD *genStatus
);
```

The **TCGGetTimeCodeType** API returns the current time code generation mode and status for the given device.

Parameters

device

The number of the TCG-100 device to interrogate.

tcType

A pointer to a variable that will receive the current mode of time code generation, one of TCGTC_XXX (defined in TCGAPI.H). This pointer can be NULL if the information is not needed.

genStatus

A pointer to a variable that will receive the status of time code generation. A non-zero value indicates active generation. A value of zero indicates the generator is stopped. This pointer can be NULL if the information is not needed.

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	The API call completed successfully.
ERROR_NOT_SUPPORTED	The API was called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

TCGSetTimeCodeType

The **TCGSetTimeCodeType** API sets/changes time code generation mode on the given TCG-100 device.

```

DWORD TCGSetTimeCodeType(
    DWORD device,
    DWORD tcType,
    BOOL startImmediate
);

```

Parameters

device

The number of the TCG-100 device to manage.

tcType

The new time code generation mode to set, once of TCGTC_XXX (defined in TCGAPI.H).

startImmediate

Set to TRUE if time code generation should begin immediately (from 00:00:00) after the generator's mode is changed.

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	The API call completed successfully.
ERROR_NOT_SUPPORTED	The API was called for an invalid/unsupported TCG-100 device number.
ERROR_INVALID_PARAMETER	The value in the <i>tcType</i> parameter is invalid.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

Changing the time code generation mode will stop any active time code generation and reset the internal time registers to 00:00:00.

TCGPollGenTime

The **TCGPollGenTime** API polls current generate time on the given TCG-100 device.

```

DWORD TCGPollGenTime(
    DWORD device,
    PTCGTIME tcgt
);

```

Parameters

device

The number of the TCG-100 device to interrogate.

tcgt

A pointer to a TCGTIME data structure to received polled time data.

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	Call completed successfully.
ERROR_INVALID_DATA	API detected that the <code>timeValidity</code> member of <code>TCGTIME</code> was set to <code>FALSE</code> , indicating that time data was corrupted during bus transfer.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

The `intPerfCount` and `sysTime` members of `TCGTIME` are not set when calling this API. See Appendix A: Data Structures for information on the various members of `TCGTIME` data structure and how they are set.

TCGIsTcSmpte

The **TCGIsTcSmpte** API returns `TRUE` when the current mode of time code generation (active or inactive) is one of the SMPTE family of time code formats.

```
BOOL TCGIsTcSmpte (  
    DWORD device  
);
```

Parameters

device

The number of the TCG-100 device to interrogate.

Return Values

`TRUE` when current mode of time code generation (active or inactive) is one of the SMPTE family of time code formats.

TCGEnableCallback

The **TCGEnableCallback** API enables one or more asynchronous notifications to a user-supplied callback function.

```
DWORD TCGEnableCallback (  
    DWORD device,  
    WORD cbToAdd,  
    DWORD *newMask,  
    PTCGASYNC_CALLBACK_ROUTINE cbFunction  
);
```

Parameters

device

The TCG-100 device number for which to modify callback features.

cbToAdd

A bit mask, containing one or more bitwise-or'd values from ASYNC_XXX (defined in TCGAPI.H).

newMask

A pointer to a variable to receive the new callback mask (enabled modes). This value includes the mode(s) indicated in *cbToAdd* as well as any previously enabled mode(s).

cbFunction

A user-supplied callback function of type PTCGASYNC_CALLBACK_ROUTINE. A skeleton example of the callback function would be:

```
DWORD _stdcall TCGAsyncCallback(DWORD device, DWORD dwStatus, PVOID data,
    DWORD dataLen)
{
    // code here to handle asynchronous notifications
}
```

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

Calling this API enables the new callback mode(s) specified in the *cbToAdd* parameter. The mode(s) in the *cbToAdd* parameter will be enabled in addition to any previously enabled callback mode(s). Call the TCGDisableCallback to disable callback mode(s).

A process may only have one callback routine registered by this function. Registering a callback routine then calling the function again with a different routine will have the effect of moving all enabled asynchronous notifications to the new routine.

See Appendix B: Callback Functions for more information.

TCGDisableCallback

The **TCGDisableCallback** API disables/turns off one or more asynchronous notifications to the user-supplied callback routine.

```
DWORD TCGDisableCallback(
    DWORD device,
    DWORD cbToRemove,
    DWORD *newMask
);
```

Parameters

device

The TCG-100 device number for which to modify callback features.

cbToRemove

A bit mask, containing one or more bitwise-or'd values from ASYNC_XXX (defined in TCGAPI.H).

newMask

A pointer to a variable to receive the new callback mask (modes still enabled).

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_NO_MORE_ITEMS	Called for one or more ASYNC_XXX notification number(s) that are not enabled.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

See Appendix B: Callback Functions for more information.

TCGStartGenerate

The **TCGStartGenerate** API initiates time code generation for the given TCG-100 device.

```
DWORD TCGStartGenerate (  
    DWORD device  
);
```

Parameters

device

The TCG-100 device number for which to initiate time code generation.

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

TCGStopGenerate

The **TCGStopGenerate** API stops time code generation for the given TCG-100 device.

```
DWORD TCGStopGenerate (  
    DWORD device  
);
```

Parameters

device

The TCG-100 device number for which to stop time code generation.

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

Stopping time code generation has the effect of freezing internal time registers. Starting generation again will resume with the time values in those registers unless another API has been called to adjust them.

TCGJamToClock

The **TCGJamToClock** API sets the time registers on the given TCG-100 device to within the resolution of the system clock (+/- 10 milliseconds on the Windows NT platform).

```
DWORD TCGJamToClock(  
    DWORD device,  
    BOOL localOption  
);
```

Parameters

device

The TCG-100 device for which to jam-sync the time code registers.

localOption

When TRUE jams time from the platform's local clock (per operating system time zone configuration).
When FALSE, jams time from the platform's UTC/GMT system clock.

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_BAD_COMMAND	Called when the TCG-100 generator was not generating time code.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

This API should be used infrequently as it ties up the platform for up to 10ms at a privileged execution level – very frequent use may cause noticeable performance degradation.

```

DWORD TCGLoadGenerateTime(
    DWORD device,
    TCGTIME *time,
    BOOL useNative,
    BOOL startOption,
    BOOL jamOption
);

```

Parameters

device

The TCG-100 device for which to set generate time code registers.

time

A pointer to a TCGTIME data structure, the contents of which will be used to set generate time code registers.

useNative

When TRUE, generate registers will be set from the native time code members of TCGTIME (*irig* member of the *rawTc* union for IRIG-B, *smpTe* member of the *rawTc* union for SMPTE). When FALSE, appropriate time will be derived from the *wHour*, *wMinute*, *wSecond*, and *wMilliseconds* members of TCGTIME.

startOption

When TRUE, indicates that time code generation should start immediately after the new time values are loaded. This option applies only when time code generation is not already active.

jamOption

When TRUE, indicates that the new time values should be jam-synced into the time code registers. This means that time code generation will be stopped, new time values loaded, and generation re-started. This option has an effect only when the TCG-100 device is actively generating time code.

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

TCGLoadGenerateTime () behaves differently depending on current mode of operation on TCG-100 and specified parameters.

When time code generation is active and this API is called the new time values will be loaded but will not take until the next frame boundary occurs. Although this provides for no “hiccup” in time code generation the new time value may be in error by up to one frame unit by the time the generators takes it. Set the *jamOption* parameter to TRUE if accuracy is desired over uninterrupted time code generation.

When time code generation is inactive and this API is called the new time values will be loaded to the TCG-100 device then frozen in the time code registers. Time will not begin advancing until time code generation is started. To start generation immediately after the load set the *startOption* parameter to TRUE.

Currently, the TCG-100 generator card does not support date manipulation in time code. Loading date via the user bits of SMPTE or the control functions of IRIG-B (e.g. IEEE1344) will work but the TCG-100 will not properly increment these values at the rollover of midnight – application software would be responsible for resetting the encoded date values correctly at the beginning of each day.

TCGLoadSlewAdj

The **TCGLoadSlewAdj** API provides the capability of adjusting the frequency of the time code generator for the purposes of slewing time synchronization.

```
DWORD TCGLoadSlewAdj(  
    DWORD device,  
    WORD frames,  
    WORD bits,  
    BYTE usAdj  
);
```

Parameters

device

The TCG-100 device for which to manage generator frequency adjustments.

frames

The number of generator frames during which to apply frequency adjustments. A frame in this context is considered one fundamental timing unit for the given time code type. This means one second for IRIG-B time code, 33.3ms (one frame) for SMPTE 30 frames/second, etc. As a special case, a value of 65535 (0xffff) will load a permanent (non-expiring) frequency adjustment.

bits

The number of frame bits upon which to apply the timing adjustment specified in *usAdj*. The duration of a bit and the number of bits per frame for a given time code type has special meaning for the TCG-100 generator. See Appendix C: Generator Frequency Adjustments for more information.

usAdj

A bit-level timing adjustment value in microseconds. This value is treated as a 7-bit signed number by the TCG-100 generator. Therefore, timing adjustments can be positive or negative (decreasing or increasing frequency, respectively).

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

Please carefully read Appendix C: Generator Frequency Adjustment before using this API.

TCGSetLeapYearFlag

The **TCGSetLeapYearFlag** API sets or clears the leap year flag for the given TCG-100 device. Relevant only for IRIG-B time code generation.

```
DWORD TCGSetLeapYearFlag(  
    DWORD device,  
    BOOL leapOption  
);
```

Parameters

device

The TCG-100 device for which to manipulate the leap year flag.

leapOption

Set to TRUE to indicate that the current year is a leap year, set to FALSE to indicate a non-leap year.

Return Values

A Win32 error code, typically:

ERROR_SUCCESS	Call completed successfully.
ERROR_BAD_COMMAND	Called when the TCG-100 device is not in IRIG-B mode.
ERROR_NOT_SUPPORTED	Called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

Although the TCG-100 generator does not manage date it does manage day-of-year encoding for IRIG-B time code. To determine how the day-of-year value should expire at the end of the year (i.e. from 365 to 366 to 1, or from 365 to 1) the leap year flag must be set by application software so that the TCG-100 knows the number of days that should be in the current year. The TCG-100 sets this flag to FALSE (non-leap year) at device power-up.

TCGGetDLLVersion

The **TCGGetDLLVersion** API returns the software version of the TCG API DLL library.

```
DWORD TCGGetDLLVersion(  
    PUINT verMaj,  
    PUINT verMin,  
    PUINT verRev  
);
```

Parameters

verMaj

A pointer to a variable that will receive the major version of the TCG API DLL library.

verMin

A pointer to a variable that will receive the minor version of the TCG API DLL library.

verRev

A pointer to a variable that will receive the revision level of the TCG API DLL library.

TCGGetDriverVersion

The **TCGGetDriverVersion** API returns the software version of the TCG-100 kernel-mode driver.

DWORD TCGGetDriverVersion(

DWORD *device*,

PUINT *verMaj*,

PUINT *verMin*,

PUINT *verRev*

);

Parameters

device

The number of the device to interrogate.

verMaj

A pointer to a variable that will receive the major version of the TCG-100 kernel-mode driver.

verMin

A pointer to a variable that will receive the minor version of the TCG-100 kernel-mode driver.

verRev

A pointer to a variable that will receive the revision level of the TCG-100 kernel-mode driver.

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	The API call completed successfully.
ERROR_NOT_SUPPORTED	The API was called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

Remarks

The TCG-100 kernel-mode driver and TCG-100 devices are installed and configured by the TCG-100 base software package for Windows NT. TCG-100 devices can be added, removed, and re-configured via the TCG-100 control panel applet.

TCGGetDeviceVersion

The **TCGGetDeviceVersion** API returns the firmware version for the given TCG-100 device.

```
DWORD TCGGetDeviceVersion(  
    DWORD device,  
    PUINT verMaj,  
    PUINT verMin  
);
```

Parameters

device

The number of the device to interrogate.

verMaj

A pointer to a variable that will receive the major version of the firmware for the given TCG-100 device.

verMin

A pointer to a variable that will receive the minor version of the firmware for the given TCG-100 device.

Return Values

A Win32 error code, typically one of:

ERROR_SUCCESS	The API call completed successfully.
ERROR_NOT_SUPPORTED	The API was called for an invalid/unsupported TCG-100 device number.
ERROR_DEV_NOT_EXIST	The device is not present, or did not initialize successfully at DLL startup.

TCGShutdown

The **TCGShutdown** API performs a graceful cleanup of API resources allocated on behalf of the calling process.

```
void TCGShutdown(void)
```

Parameters

This function has no parameters.

Remarks

Do not call any other TCG DLL API functions after calling the TCGShutdown() API.

Appendix A: Data Structures

The following data structures are used throughout the TCG DLL API library. These data structures and other definitions can be found in the TCGAPI.H header file.

```
//
// a structure defining a SMPTE time code frame
//

typedef struct _TCGSMPTE
{
    UCHAR          fu;          // frame units
    UCHAR          u1;          // user group 1
    UCHAR          ft;          // frame tens
    UCHAR          u2;          // user group 2
    UCHAR          su;          // second units
    UCHAR          u3;          // user group 3
    UCHAR          st;          // second tens
    UCHAR          u4;          // user group 4
    UCHAR          mu;          // minute units
    UCHAR          u5;          // user group 5
    UCHAR          mt;          // minute tens
    UCHAR          u6;          // user group 6
    UCHAR          hu;          // hour units
    UCHAR          u7;          // user group 7
    UCHAR          ht;          // hour tens
    UCHAR          u8;          // user group 8
} TCGSMPTE, *PTCGSMPTE;

//
// a structure defining an IRIG-B time code frame
//

typedef struct _TCGIRIG
{
    UCHAR          su;          // second units
    UCHAR          st;          // second tens
    UCHAR          mu;          // minute units
    UCHAR          mt;          // minute tens
    UCHAR          hu;          // hour units
    UCHAR          ht;          // hour tens
    UCHAR          du;          // day units
    UCHAR          dt;          // day tens
    UCHAR          dh;          // day hundreds
    UCHAR          cf1;         // control functions 1
    UCHAR          cf2;         // control functions 2
    UCHAR          cf3;         // control functions 3
    UCHAR          cf4;         // control functions 4
    UCHAR          cf5;         // control functions 5
    UCHAR          cf6;         // control functions 6
    UCHAR          cf7;         // control functions 7
    UCHAR          tod1;        // time of day nibble 1
    UCHAR          tod2;        // time of day nibble 2
    UCHAR          tod3;        // time of day nibble 3
    UCHAR          tod4;        // time of day nibble 4
    UCHAR          tod5;        // time of day nibble 5
}
```

```

} TCGIRIG, *PTCGIRIG;

typedef struct _TIME_FIELDS_WCA {
    USHORT Year;           // range [1601...]
    USHORT Month;         // range [1..12]
    USHORT Day;           // range [1..31]
    USHORT Hour;          // range [0..23]
    USHORT Minute;        // range [0..59]
    USHORT Second;        // range [0..59]
    USHORT Milliseconds; // range [0..999]
    USHORT Weekday;       // range [0..6] == [Sunday..Saturday]
} TIME_FIELDS_WCA;

typedef struct _TCGTIME {

    // raw time code values

    union
    {
        TCGIRIG      irig;
        TCGSMPTE     smpte;
    } rawTc;

    // if a time given asynchronously (by an interrupt on TCR),
    // contains high-performance counter
    // reading at time of interrupt

    LARGE_INTEGER     intPerfCount;
    TIME_FIELDS_WCA   sysTime;

    // time validity (set to FALSE if time considered invalid for any
    // reason)

    BOOL               timeValidity;

    // generic time

    WORD               wHour;
    WORD               wMinute;
    WORD               wSecond;
    WORD               wMilliseconds;
    WORD               wMonth;
    WORD               wDay;
    WORD               wYear;

    // debugging use only, use boolean 'timeValidity' member for checksum
    // verification
    WORD               rchksum;
    WORD               lchksum;

} TCGTIME, *PTCGTIME;

```

Appendix B: Callback Functions

Callback function support is a special feature of the TCG DLL API library that provides the capability for a function in your application to be asynchronously notified (called) when a special event occurs during the normal course of time code generation on the TCG-100 card. These events are specific points in real-time which may be of interest for custom applications. Callback support provides a convenient interface between the physical hardware-level interrupts that the TCG-100 card generates and your application. Your application will not need to deal with the nuances of interrupt service routines or driver interface programming.

The library supports the following types of asynchronous notifications for a callback function:

ASYNC 1HZ

When enabled, your designated function can be called once per second (1 Hz frequency) at the on-time mark beginning of each new second during time code generation.

ASYNC FRAMERATE

When enabled, your designated function can be called once per frame at the on-time mark beginning of each new frame during time code generation. The frequency of this notification depends on the duration of a time code frame, which in turn depends on the type of time code being generated. The time code formats supported by the TCG-100 have the following frame rates:

<u>Time Code Format</u>	<u>Duration of Frame</u>
SMPTE (30 frames / second)	33.3 milliseconds
SMPTE (25 frames / second)	40.0 milliseconds
SMPTE (24 frames / second)	41.6 milliseconds
IRIG-B(1)	1000 milliseconds *

* ASYNC_FRAMERATE and ASYNC_1HZ are functionally equivalent notifications for IRIG-B time code.

To make use of asynchronous notifications your application will need a callback function. Find below a skeleton example of a callback function for Visual C++. See the VCDemo sample application for a practical implementation of a callback function.

```
DWORD _stdcall myCallback(DWORD device, DWORD dwStatus, PVOID data, DWORD dataLen)
{
    PTCGTIME        time;

    switch(dwStatus)
    {
        case ASYNC_FRAMERATE:
            // handle once/frame notification

            time = (PTCGTIME)data;

            if (time->timeValidity)
            {
                // time is valid, process in some manner
            }

            break;

        case ASYNC_1HZ:
            // handle once/second notification
```

```

        time = (PTCGTIME)data;

        if (time->timeValidity)
        {
            // time is valid, process in some manner
        }

        break;
    }

    return(0);
}

```

Additional Callback Notes

- Expect that your callback function will be subject to issues inherent in multithreaded/reentrant environments and design accordingly.
- Due to the fact that several layers of software become involved in handling asynchronous notifications, the utilization of ASYNC_FRAMERATE notifications for SMPTE time code may become significantly resource-intensive on all but the faster Windows NT platforms. Consider whether or not the high frequency of this notification is suitable for your application and your platform.
- In situations where user application has both ASYNC_FRAMERATE and ASYNC_1HZ notifications enabled, the designated callback function will occasionally be called twice at (effectively) the same point in real-time- once for each of the two enabled notifications, in a back-to-back fashion with ASYNC_1HZ generating the first notification.

Appendix C: Generator Frequency Adjustment

The TCGLoadSlewAdj() API provides the advanced capability of performing frequency adjustments on the time code generator. This allows synchronizing the time and frequency of the TCG-100's generator to some reference without jam-synching time or otherwise interrupting the time code generation process. Use of this API requires an intimate knowledge of the time code specification for the time code type you are working with. This API can be used to generate time code that is out of specification which may then be unrecognizable to some time code decoding devices. Use with caution!

TCGLoadSlewAdj() accepts three parameters which control the type and duration of frequency adjustment, they are:

frames

The number of frames over which your specified frequency adjustment will be performed. A value of zero will immediately terminate any active adjustments. A value between 1 and 65534 specifies an expiring adjustment – once the given number of frames have been generated the adjustment expires and the TCG-100 reverts back to default generator frequency for current time code type. As a special case, specifying a value of 65535 (0xffff) indicates a permanent, non-expiring frequency adjustment. This type of adjustment will be applied forever until either changed by another TCGLoadSlewAdj() call, a TCGSetTimeCodeType() call, or a physical reset of the TCG-100 card.

adjBits

The number of bit periods during a frame that will receive a timing adjustment. The maximum value for this parameter depends on the time code type being generated.

Table 1 – Time Code Generator Timing Characteristics

<u>Time Code Type</u>	<u>Bit Period Duration (us)</u>	<u>Bits Periods / Frame</u>
SMPTE (30 frames / second)	208.3	160
SMPTE (25 frames / second)	250.0	160
SMPTE (24 frames / second)	260.4	160
IRIG-B(1)	125.0	8000

usAdj

This is a signed 7-bit value representing the number of microseconds of timing adjustment to be applied to each qualifying bit period as allowed by *adjBits* and *adjFrames*. Specifying a negative number here “shrinks” the duration of the adjusted bit periods ultimately increasing the frequency of the generator (making time tick ahead faster than normal). A positive number has the opposite effect, “expanding” the duration of the adjusted bit periods ultimately decreasing generator frequency (making time tick ahead slower than normal). As a rule of thumb, the value of this parameter should never exceed absolute value 2.

Frequency Adjustment Examples

- $adjFrames = 1000$, $adjBits = 10$, $usAdj = -2$, time code type = SMPTE 30 frames/second

This adjustment, when loaded, will persist over the next 1000 frames of SMPTE generated time code. The first 10 bit periods of each of those frames will be adjusted by -2 microseconds. After 1000 frames are generated this adjustment will expire. The end result of the adjustment will have been a temporary increase in generator frequency. The TCG-100 generator clock will have gained $1000 \times 10 \times 2 = 20,000\mu s$ (20ms, or slightly less than one frame) because of the adjustment.

- $adjFrames = 40000$, $adjBits = 8000$, $usAdj = 1$, time code type = IRIG-B

This adjustment, when loaded, will persist over the next 40,000 frames (seconds) of IRIG-B generated time code. In this example all 8000 bits of an IRIG-B frame will receive the 1us adjustment. After 40,000 frames (seconds) elapses the adjustment will expire. The end result of the adjustment will have been a temporary decrease in generator frequency. The TCG-100 generator clock will have lost $40,000 \times 8000 \times -1 = -320,000,000\text{us}$ (or -320 seconds) because of the adjustment.

- `adjFrames = 65535, adjBits = 2, usAdj = -1, time code type = SMPTE 25 frames / second`

This adjustment, when loaded, will be permanent (non-expiring). The adjustment will result in an increase of the generator frequency. The TCG-100's generator clock will gain time relative to real-time until the adjustment is disabled by the user application. For example, over a normal unadjusted 24-hour period 2,160,000 frames would be generated. However, with the first two bits of each of those frames adjusted by -1 microseconds, the TCG-100's generator clock will have gained 4,320,000 microseconds (4.32 seconds, or 108 frames).

Appendix D: Visual Basic 5 Interfacing Reference

Visual Basic Data Structure & Constant Definitions TCG DLL API

Option Explicit

' a structure to manage the two 32-bit components of a 64-bit number

Public Type LARGE_INTEGER

 LowPart As Long

 HighPart As Long

End Type

' a structure defining a SMPTE time code frame

Public Type TCGSMPTE

 fu As Byte

 u1 As Byte

 ft As Byte

 u2 As Byte

 su As Byte

 u3 As Byte

 st As Byte

 u4 As Byte

 mu As Byte

 u5 As Byte

 mt As Byte

 u6 As Byte

 hu As Byte

 u7 As Byte

 ht As Byte

 u8 As Byte

End Type

' a structure defining an IRIG-B1 time code frame

Public Type TCGIRIG

 su As Byte

 st As Byte

 mu As Byte

 mt As Byte

 hu As Byte

 ht As Byte

 du As Byte

 dt As Byte

 dh As Byte

 cf1 As Byte

 cf2 As Byte

 cf3 As Byte

 cf4 As Byte

 cf5 As Byte

 cf6 As Byte

 cf7 As Byte

 tod1 As Byte

 tod2 As Byte

 tod3 As Byte

```
    tod4 As Byte
    tod5 As Byte
```

```
End Type
```

```
' Time fields data structure, used in TCGTIME data type
```

```
Public Type TIME_FIELDS_WCA
```

```
    Year As Integer;
    Month As Integer;
    Day As Integer;
    Hour As Integer;
    Minute As Integer;
    Second As Integer;
    Milliseconds As Integer;
    Weekday As Integer;
```

```
End Type
```

```
Public Type TCGTIME
```

```
    ' raw time code values
```

```
    rawTc(1 To 20) As Byte    ' contains SMPTE or IRIG, depending on context
```

```
    ' if a time given asynchronously (by an interrupt on TCR), contains high-performance counter
```

```
    ' reading at time of interrupt
```

```
    intPerfCount As LARGE_INTEGER;
```

```
    sysTime As TIME_FIELDS_WCA;
```

```
    ' time validity (set to FALSE if time considered invalid for any reason)
```

```
    timeValidity As Boolean;
```

```
    ' decoded dtime
```

```
    wHour As Integer;
    wMinute As Integer;
    wSecond As Integer;
    wMilliseconds As Integer;
    wMonth As Integer;
    wDay As Integer;
    wYear As Integer;
```

```
    ' debugging use only, use boolean 'timeValidity' member for checksum verification
```

```
    rchecksum As Integer;
```

```
    lchecksum As Integer;
```

```
End Type
```

```
' hardware statuses
```

```
Public Const HW_STATUS_NORMAL = &H0
```

```
Public Const HW_STATUS_CARDCOMMFAIL = &H1
```

' maximum number of TCG-100 devices currently supported

Public Const MAX_TCG_DEVICES = &H4

' time code types

Public Const TCGTC_IRIG = &H0

Public Const TCGTC_SMPTE30 = &H1

Public Const TCGTC_SMPTE30DF = &H2

Public Const TCGTC_SMPTE25 = &H3

Public Const TCGTC_SMPTE24 = &H4

//DLL initialization statuses

' initialization was successful

Public Const TCGDLL_INIT_OK = &H0

' use of the DLL was attempted under Win '95, which is not supported

Public Const TCGDLL_INIT_OSISWIN95 = &H1

' unable to find the TCG's kernel-mode driver, which is required

Public Const TCGDLL_INIT_NOKMD = &H2

' some defined TCG devices failed to initialize

Public Const TCGDLL_INIT_SOMEFAILED = &H3

' cannot operate with TCG driver versions previous to 2.0

Public Const TCGDLL_INIT_DRIVEROLD = &H4

' unexpected failure (contact technical support)

Public Const TCGDLL_INIT_UNEXPECTED = &H255

' device initialization status

Public Const DEVICE_INIT_OK = &H0

Public Const DEVICE_INIT_CFGERROR = &H1

Public Const DEVICE_INIT_NOTPRESENT = &H2

' asynchronous notifications (maps to TCG interrupts)

' frame rate interrupts, varies for tc format

Public Const ASYNC_FRAMERATE = &H00000001

' 1Hz (once/second) interrupts

Public Const ASYNC_1HZ = &H00000002

Public Const ASYNC_ALL = &H7fffffff

Public Const ASYNC_MASK_ERROR = &H80000000

Visual Basic Function Declarations for the TCG DLL API

Declare Function TCGGetDLLInitStatus Lib "c:\filespec\tcgdll.dll" () As Long;

Declare Function TCGGetDeviceInit Lib "c:\filespec\tcgdll.dll" (ByVal device As Long) As Long;

Declare Function TCGGetTimeCodeType Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByRef tcType As Long, ByRef genStatus As Long) As Long;

Declare Function TCGSetTimeCodeType Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal tcType As Long, ByVal startImmediate As Boolean) As Long;

Declare Function TCGGetDLLVersion Lib "c:\filespec\tcgdll.dll" (ByRef verMaj As Integer, ByRef verMin As Integer, ByRef verRev As Integer) As Long;

Declare Function TCGGetDriverVersion Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByRef verMaj As Integer, ByRef verMin As Integer, ByRef verRev As Integer) As Long;

Declare Function TCGGetDeviceVersion Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByRef verMaj As Integer, ByRef verMin As Integer) As Long;

```

Declare Function TCGPollGenTime Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByRef tcgt As
    TCGTIME) As Long;
Declare Function TCGIsTcSmpTe Lib "c:\filespec\tcgdll.dll" (ByVal device As Long) As Boolean;
Declare Function TCGEnableCallback Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal
    cbToAdd As Integer, ByRef newMask As Long, ByVal cbFunction As Long) As Long;
Declare Function TCGDisableCallback Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal
    cbToRemove As Long, ByRef newMask As Long) As Long;
Declare Function TCGShutdown Lib "c:\filespec\tcgdll.dll" ();
Declare Function TCGStartGenerate Lib "c:\filespec\tcgdll.dll" (ByVal device As Long) As Long;
Declare Function TCGStopGenerate Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal
    stopImmediate As Boolean) As Long;
Declare Function TCGLoadGenerateTime Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByRef time
    As TCGTIME, ByVal useNative As Boolean, ByVal startOption As Boolean, ByVal jamOption
    As Boolean) As Long;
Declare Function TCGJamToClock Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal localOption
    As Boolean) As Long;
Declare Function TCGLoadSlewAdj Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal frames As
    Integer, ByVal bits As Integer, ByVal usAdj As Byte) As Long;
Declare Function TCGSetLeapYearFlag Lib "c:\filespec\tcgdll.dll" (ByVal device As Long, ByVal
    leapOption As Boolean) As Long;

```

Callback Function & Registration Process Example

```
TCGEnableCallback(0,cbToAdd, AddressOf myCallback)
```

```
Public Function myCallback(ByVal dwDevice As Long, ByVal dwReason As
    Long, ByRef myTime As TCRTIME, ByVal dwSize As Long) As Long
```

```

    Select Case dwReason
        Case ASYNC_1HZ
            'handle once/second case

            Case ASYNC_FRAMERATE
                'handle frame rate case

```

```
End Select
```

```
End Function
```