Time Tag Unit

User Manual

Version 1.9

Table of Contents

Table of Contents 2
Revision History 3
Device Features 4
High channel count 4
Continuous timing 4
High rate 4
Modular design 4
Easy to use 4
Hardware 5
Connectors 5
Time-tag Inputs 5
10 MHz Input 5
Led 6
Power Supply 6
Safety of operation: 6
Disclaimer: 6
Driver Installation 7
Windows 8 7
Pre Windows 8 7
Setup of Timetag Explorer 8
Using Timetag Explorer 12
Connection and Calibration 12
Calibrate Button 13
File Menu 13
Tab TimeTag 15
Use10MHz 15
UseLevelGate 15
UseEdgeGate 16
FilterMaxTime FilterMinCount 16
FilterExceptions 16
ExceptionMask 16
Tab SaveTags 17
Tab Inputs 20
Name 20
Level 20
Edge 20
Active 20
Tabs Delays and Logic 20
Software Interface 21
.NET 21
Simple -NET Program 22
How to use the .NET Demo Source Code 23
C++ 24
Simple Program Fragment 24
How to use the C++ Demo Source Code 25
Basic Functions 26
Time Tag Readout 28
Edge Sensitive Gating 29

Level Sensitive Gating 29
Time-Tag Filter 30
10 MHz Input 31

Revision History

Date	Document Version	FPGA Version	Dll Version	Remark
2010-03-29	1.0			Initial revision
2010-04-14	1.1	2.10		Outputs
2010-08-14	1.2	2.11		Added TimeTagFilter
2010-10-27	1.3			Special Features removed
2011-09-08	1.4	2.14		Removed “Windows XP only” Specified power supply ReadTags returns two arrays 10 MHz rise time specified
2012-03-12	1.5	2.15		Timetag explorer Hardware Revision 3 Demo sourcecode
2012-03-23	1.6			Minor corrections Removed tabs “Delay” and “Logic”
2012-07-28	1.7	2.16		USB 3.0 warning SetGateWidth Filter Exceptions
2014-04-23	1.8	2.17		Better documentation of ReadTags return values Setup under Windows 8 Documentation of C++ Interface
2014-05-22	1.9			Software Version 2.17.5 TagsPresent()
2014-09-30		2.18	2.18.6	SetWindowWidthEx() Dlls are renamed according to version

Support

The success of IQD and dotfast consulting is based on tight interaction to our customers. Please don't hesitate to contact I us in case you experience any problems.

This product is subject to constant development and improvement.
You have an idea for a new feature? You would like to have a change in the software ?
Please contact us at any time.

Device Features

High channel count

The device offers up to 16 inputs with a timing resolution of 156,25 ps or 78,125 ps, depending on the device type.

Continuous timing

The time starts with the first tag and measures the time over arbitrary intervals which can be up to a year.
The long-term accuracy is only limited by the quality of the external timing reference (10 MHz)

High rate

The maximum burst rate is 200 MHz on each input. 1024 pulses can be accepted at the maximum burst rate on each input.

The sustained on-board processing rate is 100 MHz in total. (All inputs together)
The data is stored on the on-board SRAM with this a maximum rate of 100 MHz, hence a block of 512.000 tags can be stored with a rate of 100 MHz.

The continuous transmission rate over USB is 11 MHz.

Modular design

Because of the flexible und modular design, it is easy to add custom features to the device. Example features are histograms, event counting or quantum cryptography.

Easy to use

With the USB connection, the device can be connected to different computers easily, including notebook computers.
The .NET interface can be used in many software environments.
In addition to the standard .NET interface, a C++ interface is provided as well.

Hardware

Connectors

Time-tag Inputs

The device offers up to 16 inputs. They are located on the front side of the device.

Connector	SMA
Termination	50 Ohm
Time resolution	156,25 ps or 78,125 ps
Threshold	Adjustable from -2 V to 2 V
Coupling	DC
Relevant Edge	Positive or negative

10 MHz Input

The 10 MHz input is located on the rear side of the device.

It can be used to increase the stability of long-term measurements by connecting an stable clock source.

Connector	BNC
Termination	1kOhm
Coupling	AC

Levels

There are two possibilities to drive the 10 MHz input:

	Level	**Termination**
Sinus	1 Vpp nominal	1 kOhm internal
Digital	Minimum level 500 mVpp	External 50 Ohm termination required

The input is AC coupled. For this reason it can be driven by a variety of digital signals.

When using digital signals the rise and fall time should be greater than 5 ns to avoid ringing. As long as the corresponding 10 MHz error flags do not raise, the time base can be considered as valid.

Led

There are three LED on the front panel.

		Termination
Power	Blue	On when the unit is working.
USB	Blue	On when there is an USB connection
Error	Red	On when there is an error occurs and the corresponding error flag that is not read out already.

After powering up all LEDS are shining softly. This indicates that the power is on and the unit is booting.
After booting the blue POWER led lights brightly.

Power Supply

The unit uses the following pins of the nim connector

Signal	**Pin**	**Current**
GND	34	
+12 V	16	Typical 0.9 A continuous Typical 2.7 A on power up

Older devices with hardware revisions before 3 need two voltages: -6 V and +12 V

Safety of operation:

The modules are only designed for operation within a NIM crate, which is powered with a properly rated and installed power supply. In particular, in order to ensure safe operation of the logic module and any connected instruments or computers, it is critical that the NIM crate and its power supply are connected to electric ground through the power connector, as required by the local safety standards.

Ensure the module is operated vertically, and that the air free is to flow through the cooling vents in the top and bottom sides of the unit.
Never operate the module horizontally, or with covered air vents - this could lead to overheating and consequent damage.

Disclaimer:

Please note that we cannot take responsibility for any harm or damage caused to or by the logic unit or any connected devices or instruments, in the case that the unit is operated outside of a NIM crate, and/or not powered through a properly installed NIM power supply.

Driver Installation

Windows 8

Devices with FPGA version greater than 2.17 do not need a driver installed on Windows 8.

Just plug in the device. The driver loads automatically.

Do not connect a device with firmware version below 2.17 to a computer running Windows 8.

Windows stores the presence of an automatic driver in the registry. When it doesn’t find a driver on the first attempt, it will never ask the device again.

Pre Windows 8

Installation of the driver is very straight forward:

1.) Extract the Zip file with the USB driver and remember the location.
2.) Plug the unit into the NIM crate
3.) Connect the unit to the computer using a USB cable
4.) Power on the NIM crate
5.) Windows recognizes the new device and asks for a driver.
6.) Choose manual driver selection when prompted
7.) Navigate to the folder with the extracted files and open the sub folder „USB Driver“
8.) The driver is installed and the device is ready for use.

Setup of Timetag Explorer

Timetag Explorer is a small Windows application that helps you to get familiar with the unit without having to write your own software.

Please Note: The links in the start menu are renamed each installation. You can have several versions of TimeTag Explorer and ttInterface.dll at the same time.

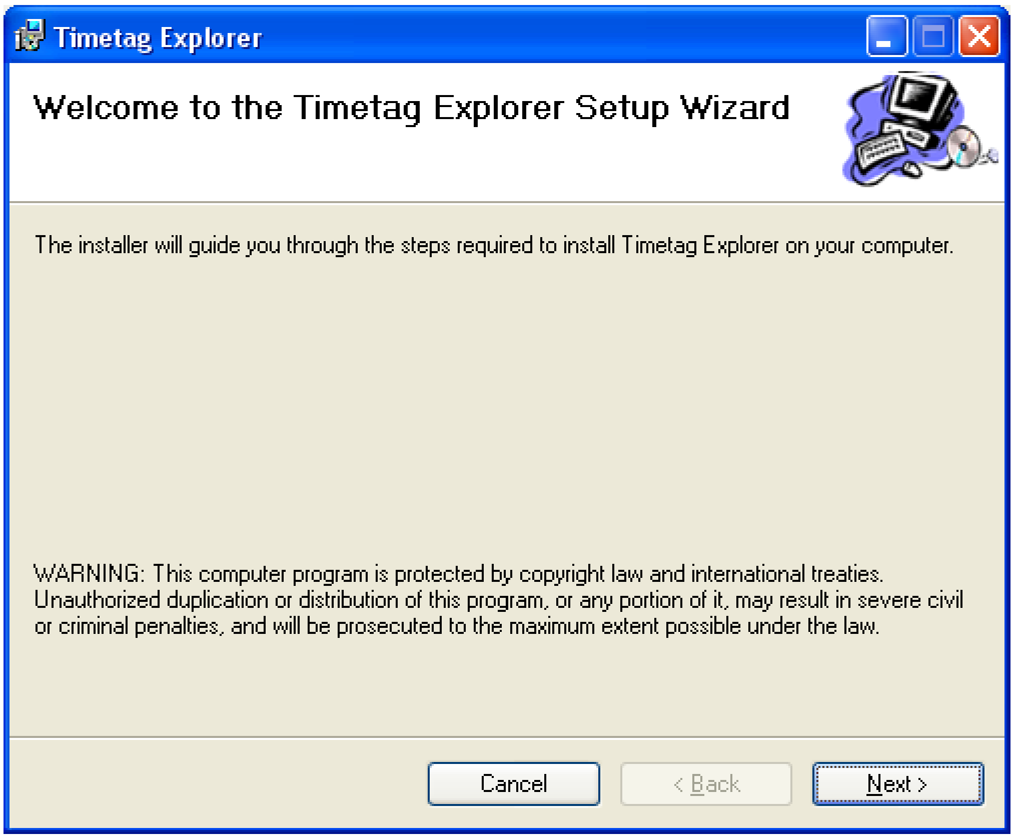
Prerequisites

Before installing Timetag explorer, please take sure that the following components are installed on your computer:

- Microsoft .NET Framework 4.0 (The “Client Profile” version will do)
  It can be downloaded here:
  `www.microsoft.com/en-US/download/details.aspx?id=17718`
- Microsoft Visual C++ 2010 Redistributable Package
  It can be downloaded here:
  `www.microsoft.com/en-US/download/details.aspx?id=5555`

Installation

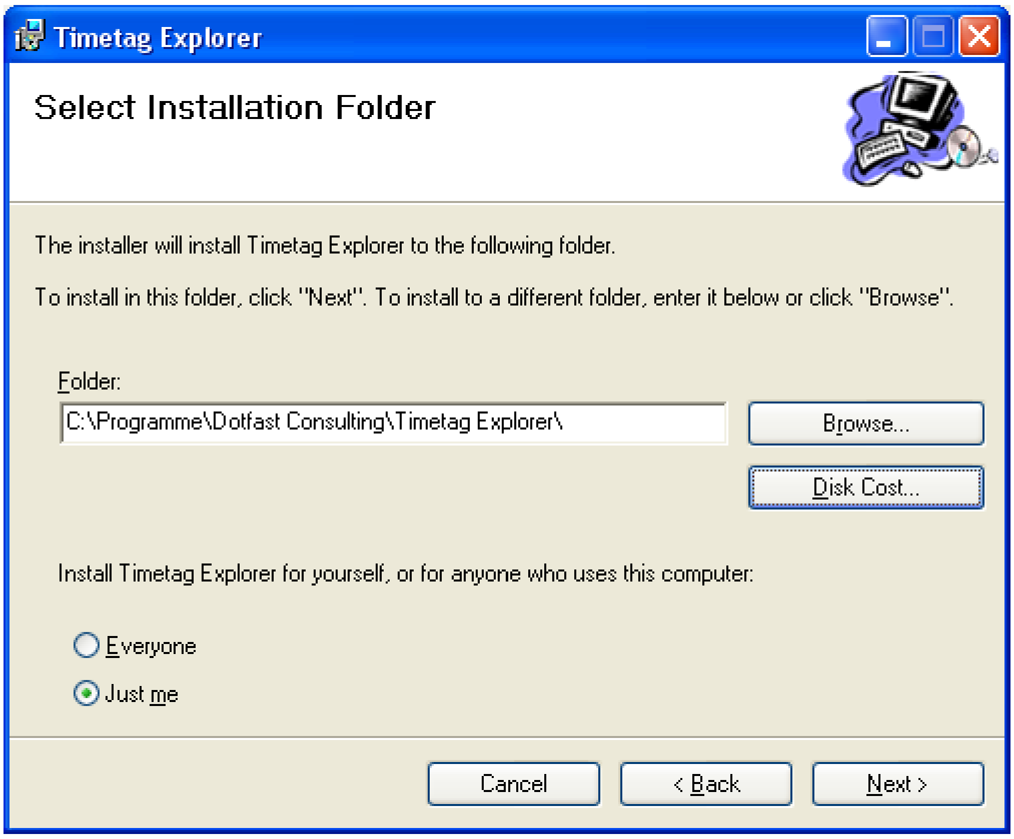
Double click on TimeTagExplorerSetup.msi. The program installs like any standard windows application.



Welcome to the Timetag Explorer Setup Wizard

The installer will guide you through the steps required to install Timetag Explorer on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.



Timetag Explorer

Select Installation Folder

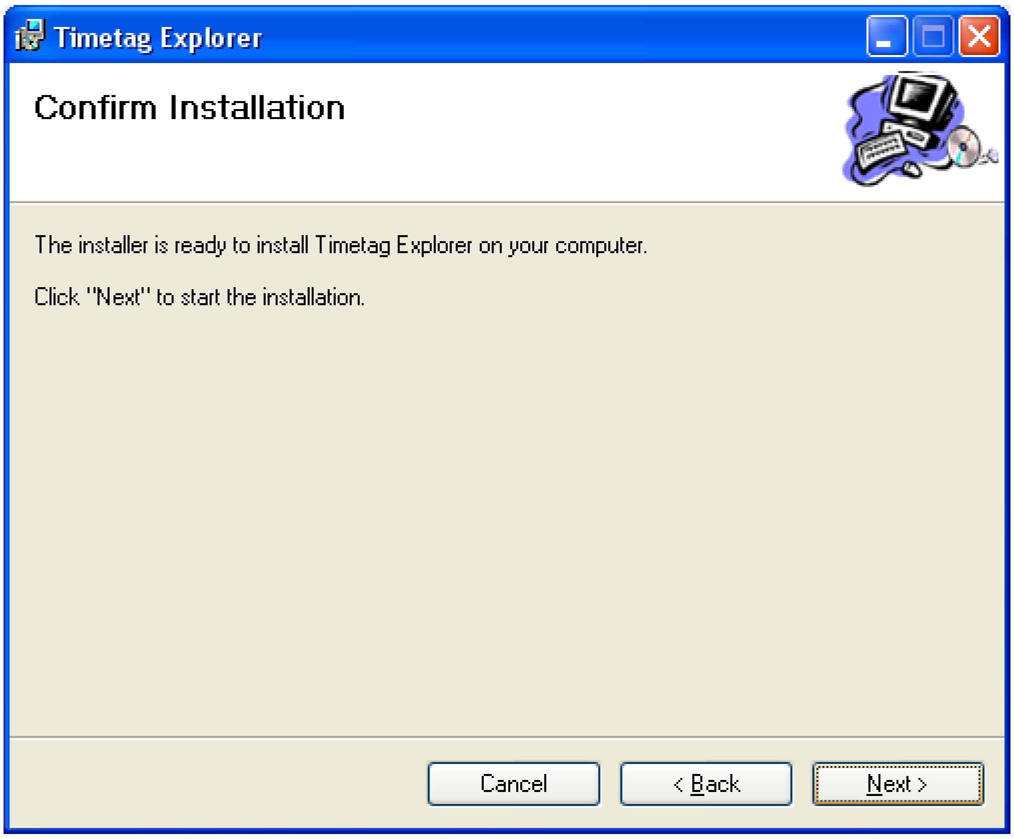
The installer will install Timetag Explorer to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Programme\Dotfast Consulting\Timetag Explorer\

Install Timetag Explorer for yourself, or for anyone who uses this computer:

Everyone [✓ Just me]

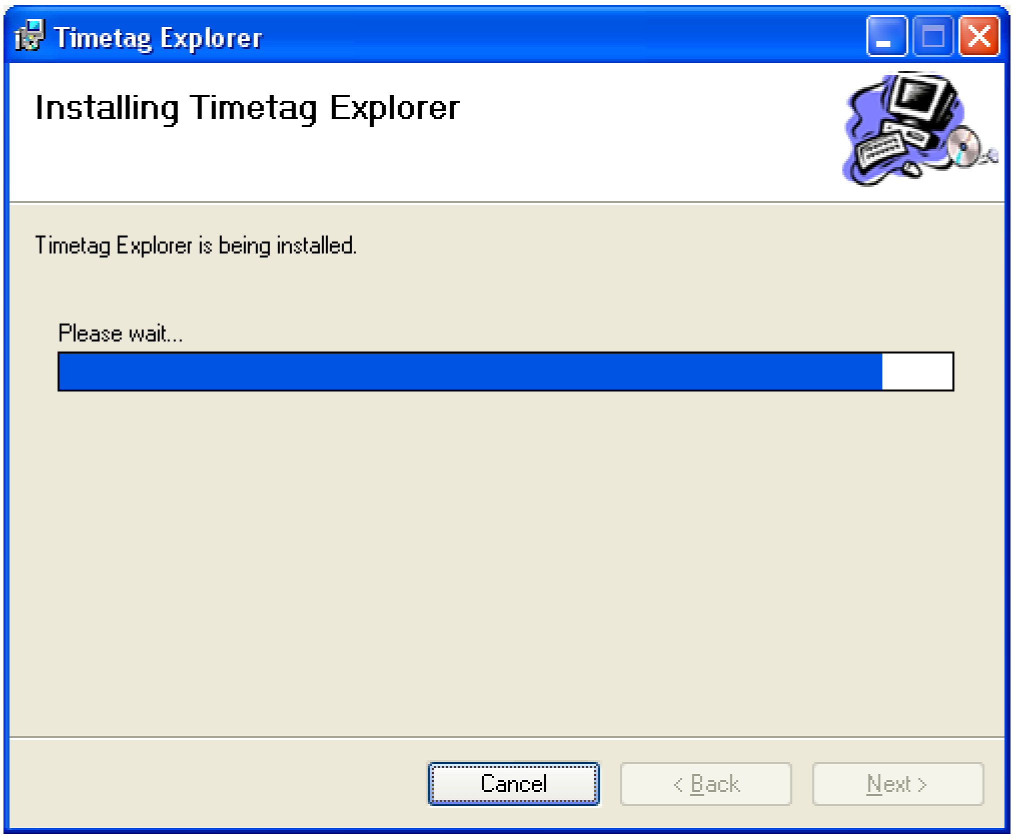


Timetag Explorer

Confirm Installation

The installer is ready to install Timetag Explorer on your computer.

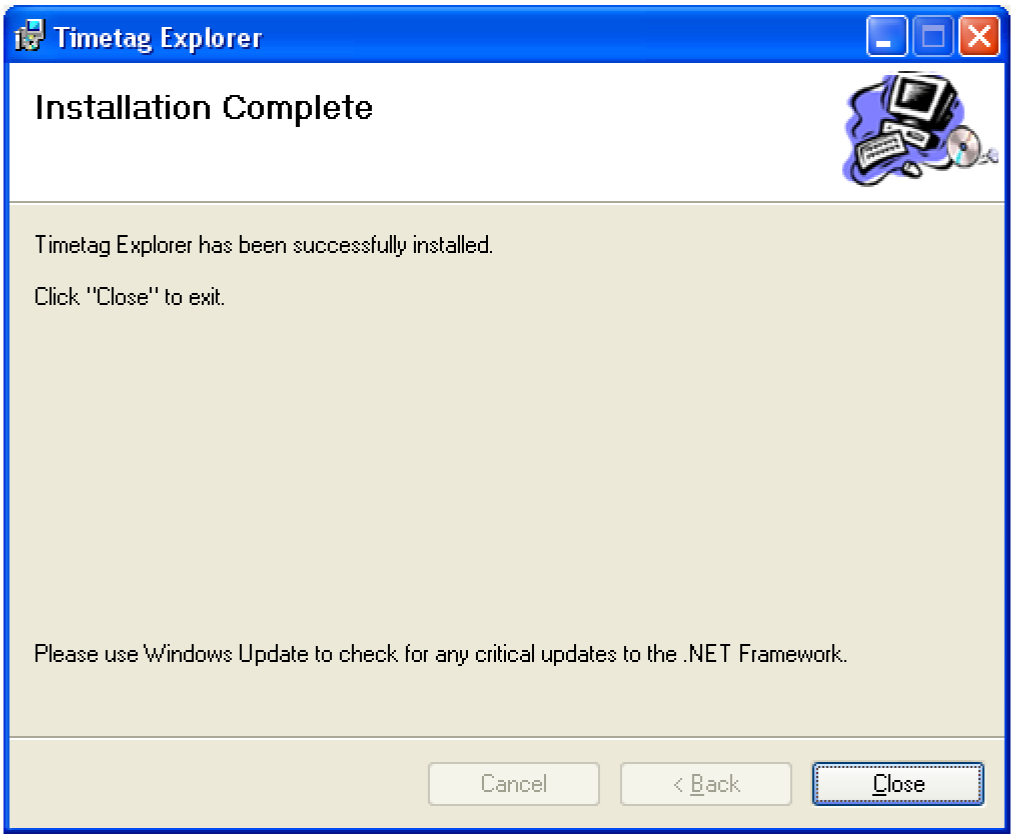
Click "Next" to start the installation.



Installing Timetag Explorer

Timetag Explorer is being installed.

Please wait...



Installation Complete

Timetag Explorer has been successfully installed.

Click "Close" to exit.

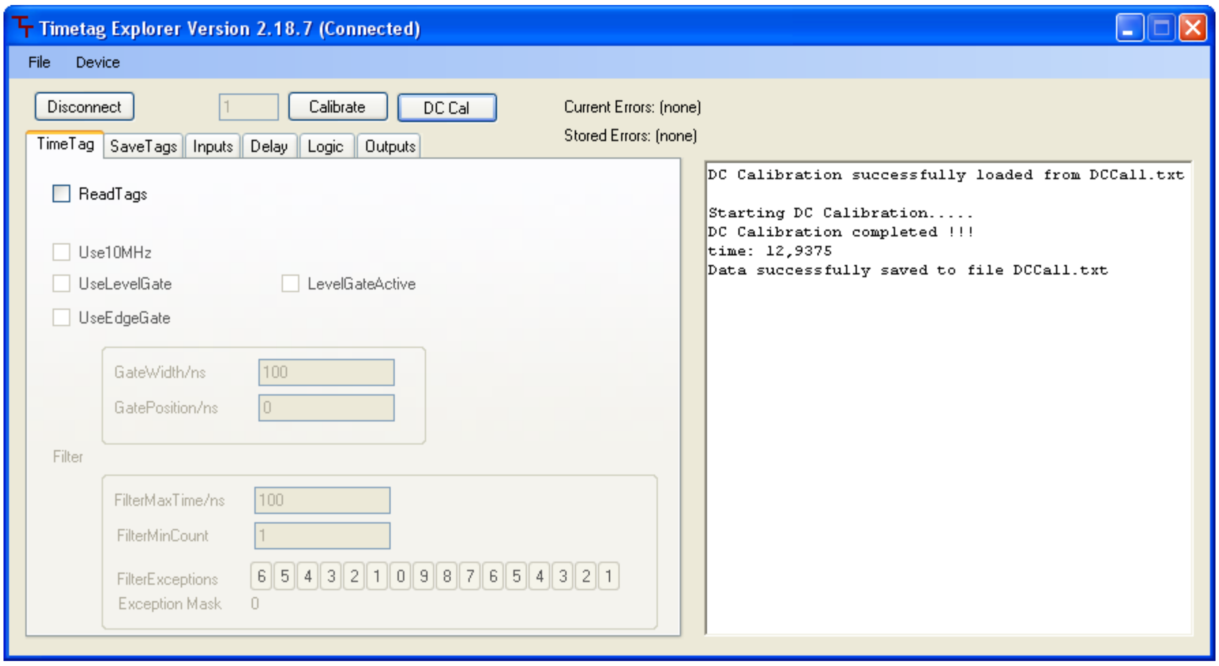
Please use Windows Update to check for any critical updates to the .NET Framework.

Using Timetag Explorer

The Timetag Explorer is an easy to use application that can be used to verify the function of the device.

Connection and Calibration

After start-up, only the “Connect” button can be selected. All other functions are disabled.
A click to the “Connect” button establishes the connection to the device.

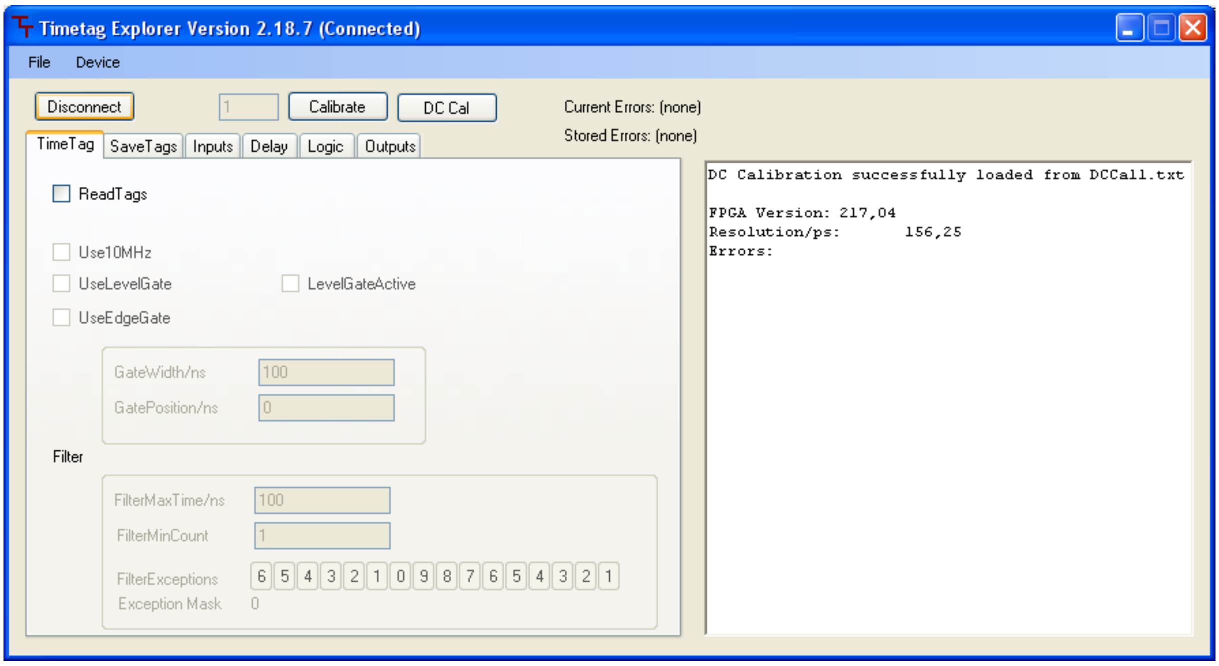


Current Errors: (none)
Stored Errors: (none)

GateWidth/ns: 100
GatePosition/ns: 0
FilterMaxTime/ns: 100
FilterMinCount: 1
Exception Mask: 0

```
DC Calibration successfully loaded from DCCal1.txt

Starting DC Calibration.....
DC Calibration completed !!!
time: 12,9375
Data successfully saved to file DCCal1.txt
```

After successful connection the user interface looks like this:



```
DC Calibration successfully loaded from DCCal1.txt

FPGA Version: 217,04
Resolution/ps:       156,25
Errors:
```

Calibrate Button

Clicking this button calibrates the unit. This operation needs about 4-10 seconds to complete.

Calibration is an optional process that increases accuracy.

Calibration should be done before any other feature is selected.

File Menu

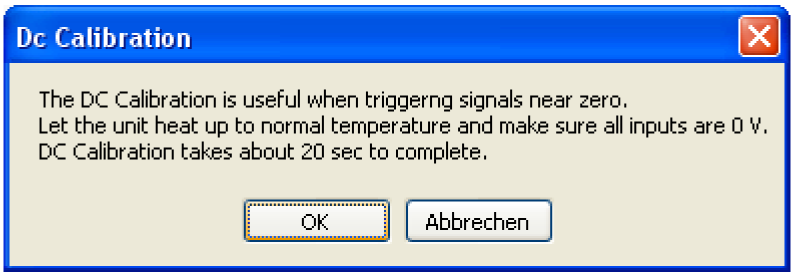
The menu offers the standard “Open”, “Save” and “Save As” functions to manage parameter files. The parameter files have the extension “.timetag”.

DC Calibration

DC calibration is a useful feature when triggering very small signals near zero. DC Calibration only has to be done only once for each unit.

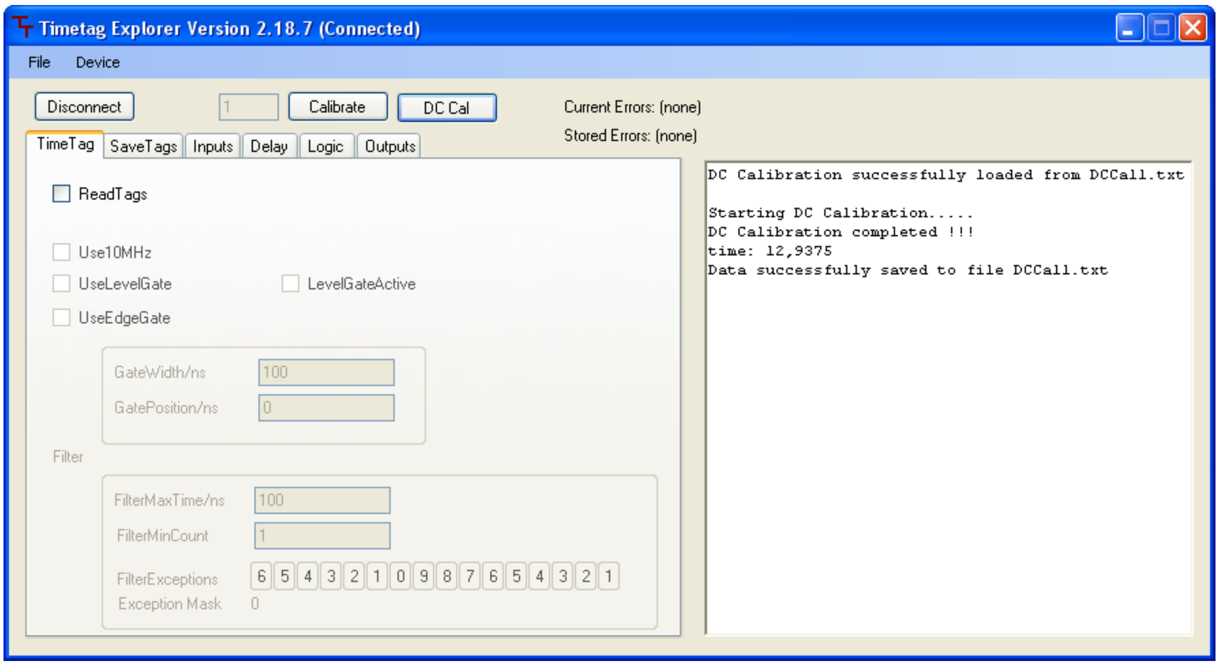
Before starting DC calibration please make sure that all inputs have zero voltage. This is done best by disconnecting all SMA connectors.

Then press the button “DC Cal”.
The following dialog appears. Press “Ok”


Dc Calibration

The DC Calibration is useful when triggerng signals near zero.
Let the unit heat up to normal temperature and make sure all inputs are 0 V.
DC Calibration takes about 20 sec to complete.

After a few seconds the calibration is completed and the calibration data is saved to disk.


Timetag Explorer Version 2.18.7 (Connected)

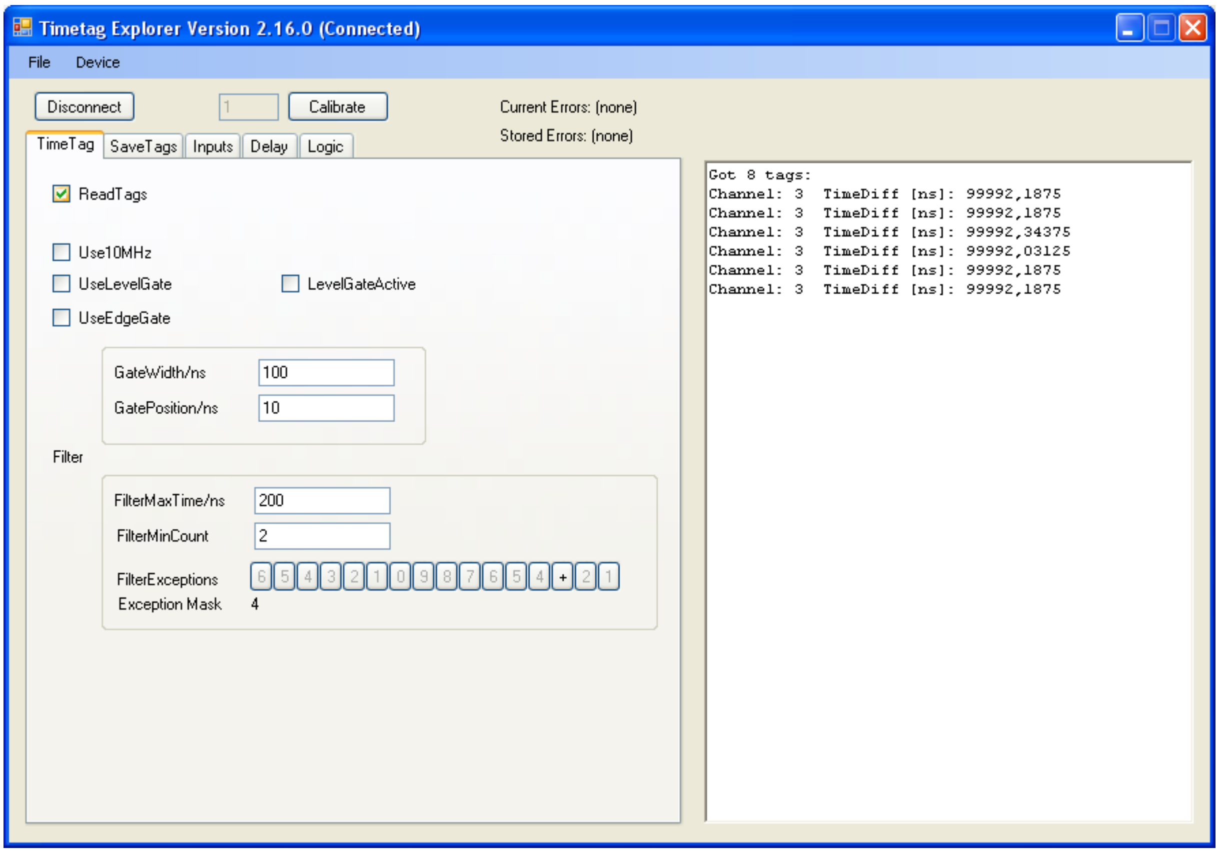
Current Errors: (none)
Stored Errors: (none)

```
DC Calibration successfully loaded from DCCal1.txt

Starting DC Calibration.....
DC Calibration completed !!!
time: 12,9375
Data successfully saved to file DCCal1.txt
```

The unit number is encoded in the file name. When there are several units connected to one computer, each unit load its correct correlation data. This is true as long as you don’t change the USB wiring.

Tab TimeTag

This tab contains the very basic operation modes. When selecting “ReadTags” the tags are automatically read. Displayed are the channel number and the time difference to the previous time value.
The values are displayed on the right side of the window.



Current Errors: (none)
Stored Errors: (none)

- [✓] ReadTags
- [ ] Use10MHz
- [ ] UseLevelGate
- [ ] LevelGateActive
- [ ] UseEdgeGate

GateWidth/ns	100
GatePosition/ns	10
FilterMaxTime/ns	200
FilterMinCount	2
FilterExceptions	6 5 4 3 2 1 0 9 8 7 6 5 4 + 2 1
Exception Mask	4

```
Got 8 tags:
Channel: 3  TimeDiff [ns]: 99992,1875
Channel: 3  TimeDiff [ns]: 99992,1875
Channel: 3  TimeDiff [ns]: 99992,34375
Channel: 3  TimeDiff [ns]: 99992,03125
Channel: 3  TimeDiff [ns]: 99992,1875
Channel: 3  TimeDiff [ns]: 99992,1875
```

Use10MHz

This checkbox selects the external timing reference. When no appropriate signal is connected on the rear side of the device, an error flag is shown.

When using the 10 MHz reference, input 16 on the 156 ps devices and input 8 on the 78 ps device can not be used.

UseLevelGate

The level gate is a very easy to use gating mode. When the input 9 is high, tags are processed normally.
When it is low, the input signals are ignored.
The checkbox “LevelGateActive” is automatically checked, when the gate signal is high.
Please note that the level gate signal has in internal jitter of 5 ns.

UseEdgeGate

The edge gate is a sophisticated gating mode. The gate opens with the active edge of the gate signal (input 8) and is open for a specified time. (Parameter GateWidth)

The position of the gate can be adjusted to compensate cable delays and similar effects.

Please note, that the gate position can be a negative and positive time interval.

FilterMaxTime
FilterMinCount

The filter can be used to reduce USB bandwidth. Tags are transmitted only, when they appear in groups. A group must have FilterMinCount entries. The entries of the group must be separated no more then FilterMaxTime.

FilterExceptions

All Inputs marked with a „+“ are excluded from the filter. They are always transmitted. (e.g. 1pps pulse)

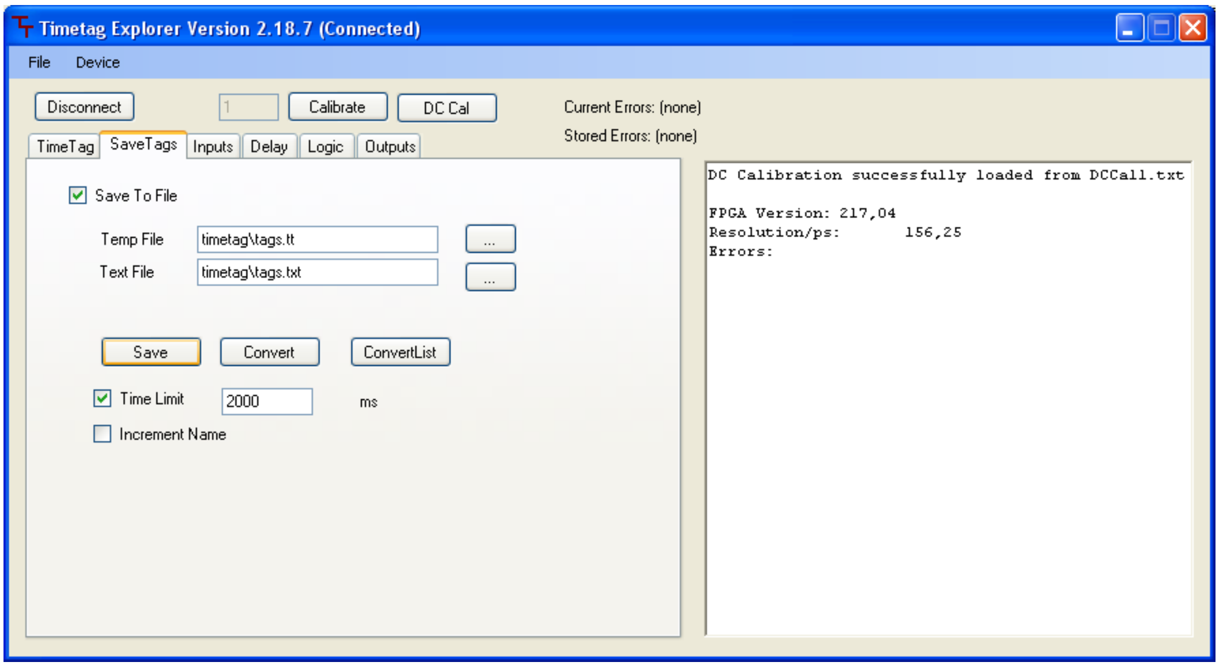
ExceptionMask

This value is changes automatically when FilterExceptions is changed. This is the value to be transmitted to SetFilterExceptions().

Tab SaveTags

This tab can be used to store the time tags to an ACSII file. To improve the transmission rate, the data is written to a temporary file first. This way the full data rate can be saved to disk.

The names of the temporary file an the output file can be entered in the two corresponding text boxes. The path can be a relative or an absolute path. When the directory is not present, it is created.

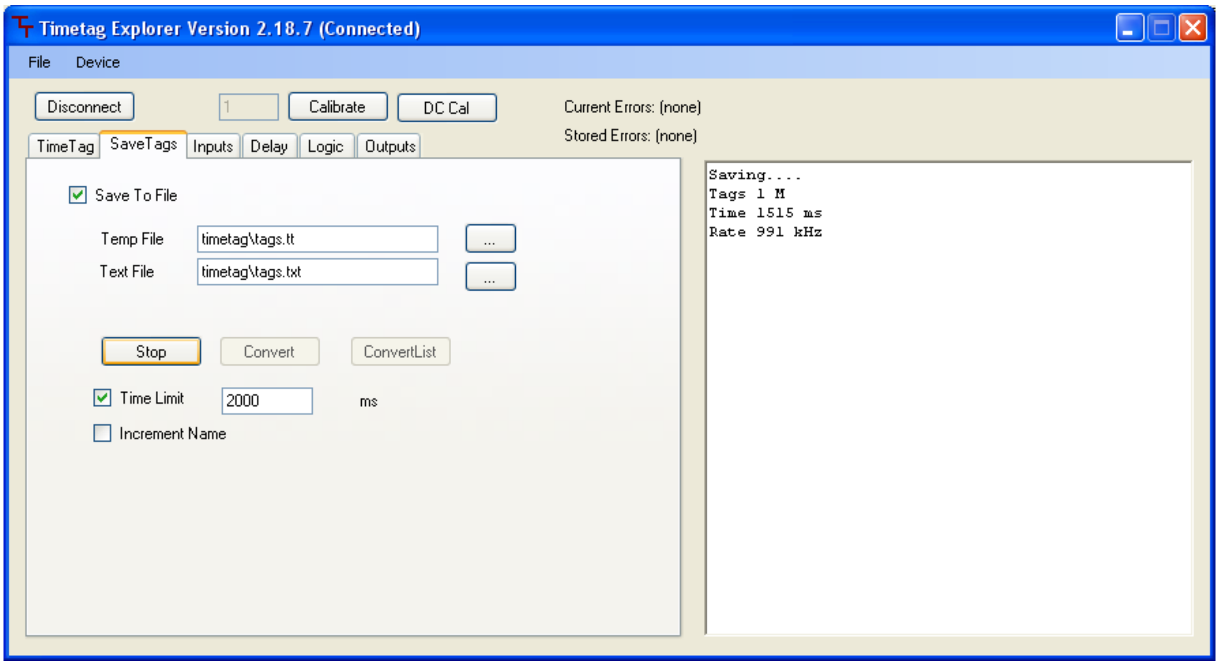


Current Errors: (none)
Stored Errors: (none)

[✓] Save To File
Temp File: timetag\tags.tt
Text File: timetag\tags.txt
[✓] Time Limit: 2000 ms
[ ] Increment Name

```
DC Calibration successfully loaded from DCCal1.txt

FPGA Version: 217,04
Resolution/ps:       156,25
Errors:
```

Clicking the “Save” button initiates the capturing of the data. Data is saved until the “Stop” button is pressed. Alternatively one can enter a time limit. In that case data is saved for a predefined amount of time, e.g. 2 sec.

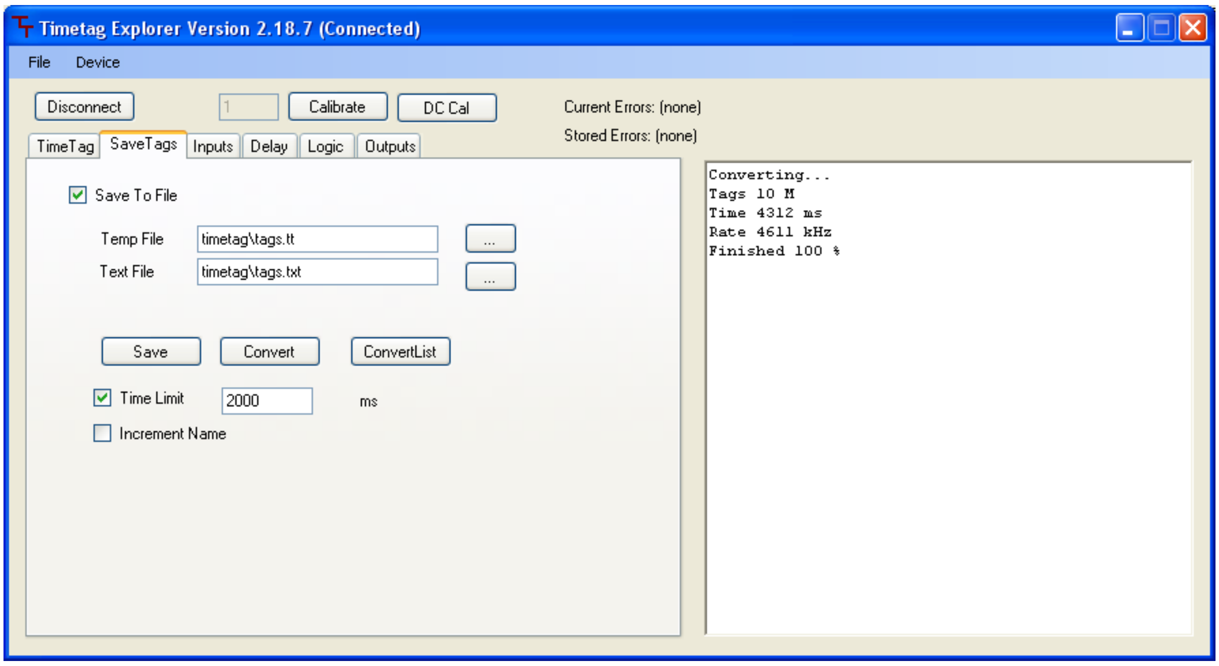


Current Errors: (none)
Stored Errors: (none)

[✓] Save To File
Temp File: timetag\tags.tt
Text File: timetag\tags.txt
[✓] Time Limit: 2000 ms
[ ] Increment Name

```
Saving....
Tags 1 M
Time 1515 ms
Rate 991 kHz
```

After pressing “Convert” the temporary file is converted to a text file.



Current Errors: (none)
Stored Errors: (none)

Save To File: ✓
Temp File: timetag\tags.tt
Text File: timetag\tags.txt
Time Limit: ✓ 2000 ms
Increment Name: ☐

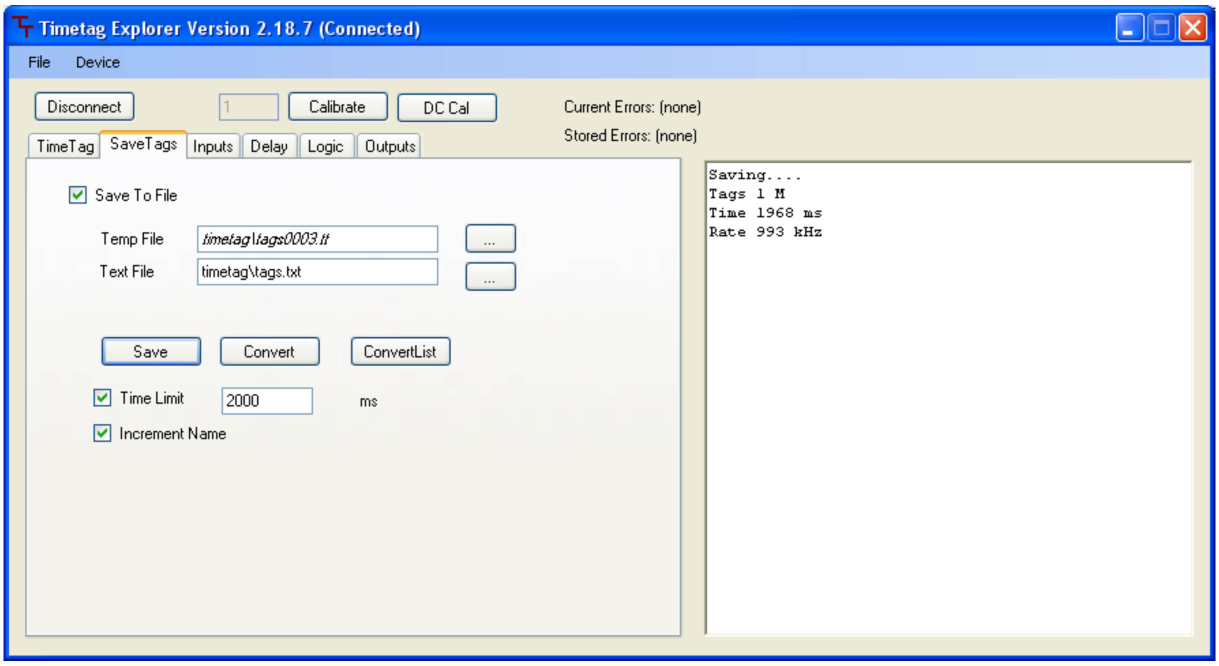
```
Converting...
Tags 10 M
Time 4312 ms
Rate 4611 kHz
Finished 100 %
```

The converted text file contains the channel number [1..16] and the corresponding time. They are separated by a tab character.
The time is stored as multiples of the internal timing resolution. The reference time is the time of the first tag.
In case of an overflow, a special tag 28 is inserted in the data stream.

Example: (1 MHz on channel 1)

```
1	0
1	6399
1	12799
1	19199
1	25598
1	31998
1	38397
1	44797
1	51197
1	57595
1	63995
1	70394
1	76794
```

By choosing “IncrementName” the temporary filename is automatically changed each time “Stop” is pressed. This features is particularly usefull in conjunction with the “ConvertList” feature.



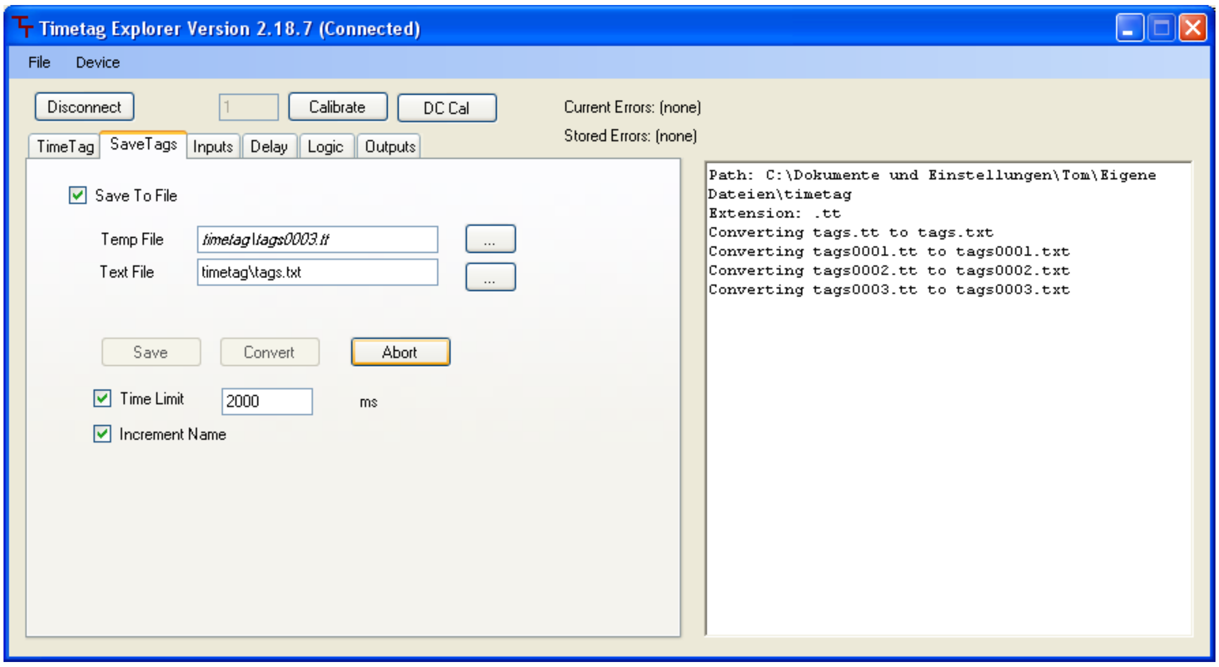
Current Errors: (none)
Stored Errors: (none)

Save To File ✓
Temp File: timetag\tags0003.tt
Text File: timetag\tags.txt
Time Limit ✓ 2000 ms
Increment Name ✓

```
Saving....
Tags 1 M
Time 1968 ms
Rate 993 kHz
```

The “Convert List” feature automatically converts all temporary files in a certain folder. The folder and the file extension are taken from the “Temp File” text box. The text box just has to link to one temporary file. “Convert List” automatically converts all files.

This feature is also quite helpful when saving a list of temporary files using the API (e.g. using LabView)

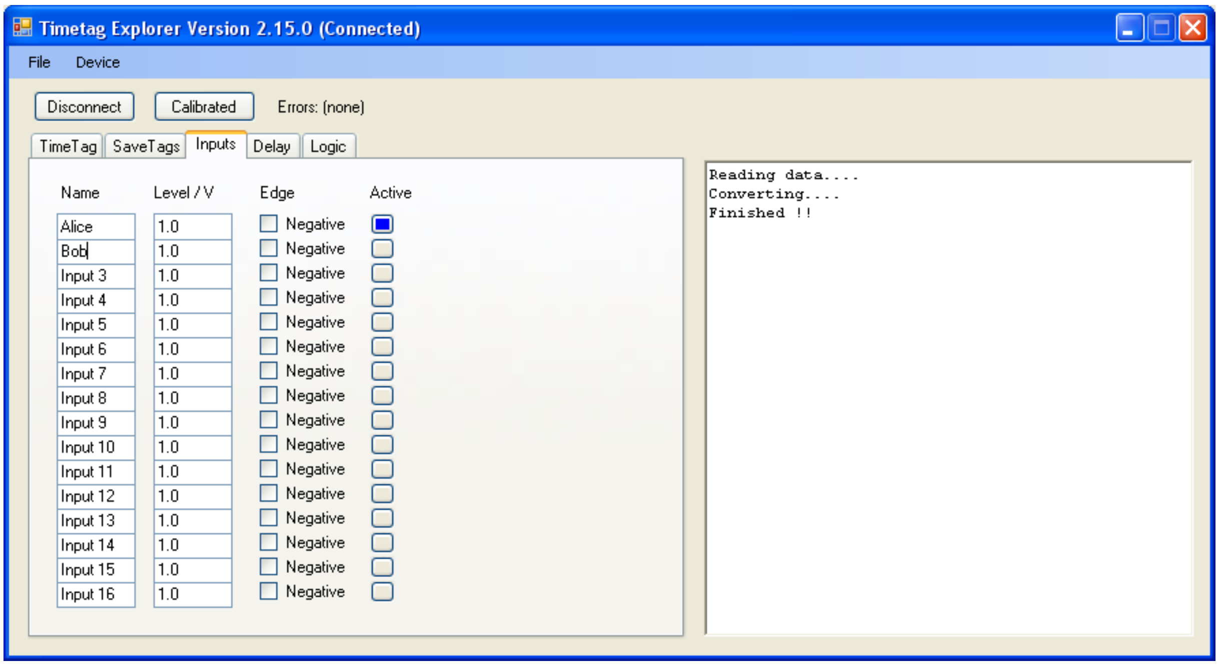


Current Errors: (none)
Stored Errors: (none)

Save To File ✓
Temp File: timetag\tags0003.tt
Text File: timetag\tags.txt
Time Limit ✓ 2000 ms
Increment Name ✓

```
Path: C:\Dokumente und Einstellungen\Tom\Eigene
Dateien\timetag
Extension: .tt
Converting tags.tt to tags.txt
Converting tags0001.tt to tags0001.txt
Converting tags0002.tt to tags0002.txt
Converting tags0003.tt to tags0003.txt
```

Tab Inputs



Name	Level / V	Edge	Active
Alice	1.0	☐ Negative	🟦
Bob	1.0	☐ Negative	☐
Input 3	1.0	☐ Negative	☐
Input 4	1.0	☐ Negative	☐
Input 5	1.0	☐ Negative	☐
Input 6	1.0	☐ Negative	☐
Input 7	1.0	☐ Negative	☐
Input 8	1.0	☐ Negative	☐
Input 9	1.0	☐ Negative	☐
Input 10	1.0	☐ Negative	☐
Input 11	1.0	☐ Negative	☐
Input 12	1.0	☐ Negative	☐
Input 13	1.0	☐ Negative	☐
Input 14	1.0	☐ Negative	☐
Input 15	1.0	☐ Negative	☐
Input 16	1.0	☐ Negative	☐

```
Reading data....
Converting....
Finished !!
```

Name

In this text box a name can be assigned to each input. This is optional.

Level

This is the threshold voltage of the input.

Edge

The “Edge” checkbox toggles the relevant edge. If not selected, the positive edge is relevant.

Active

When the input has an active level, the color of this field turns to blue. The active level depends on the selected edge. When using positive edge, the active level is high.

Tabs Delays and Logic

These tabs cover special features and are only active when special versions of the timetag unit are connected. These tabs are described in the “Logic User Manual”.

Software Interface

The unit is delivered with a sophisticated software interface. This software interface has built in multi processing support.

One processor can do the usb communication and time-tag preprocessing. The other processor(s) is / are free for user handling of the read tags. (Storage / real time calculation ..)

In this configuration very high data rates can be transmitted over USB.

.NET

.NET assemblies can be integrated in most programming environments, including LabView and MatLab.

The software interface is contained in the class ttInterface that is present in the file ttInterface.dll. The dll file can be found in the folder where TimeTagExplorer is installed.

ttinferface.dll uses .NET framework 4.0 which is not officially supported by older versions of LabView. But it still works because none of the 4.0 features is actually used.

To use 4.0, the following configuration file is required:

The configuration file must be placed next to `LabVIEW.exe` and must be named `LabVIEW.exe.config`. The following example instructs LabVIEW to load the CLR 4.0:

```
<configuration>
<startup useLegacyV2RuntimeActivationPolicy="true">
<supportedRuntime version="v4.0.30319"/>
</startup>
</configuration>
```

Simple -NET Program

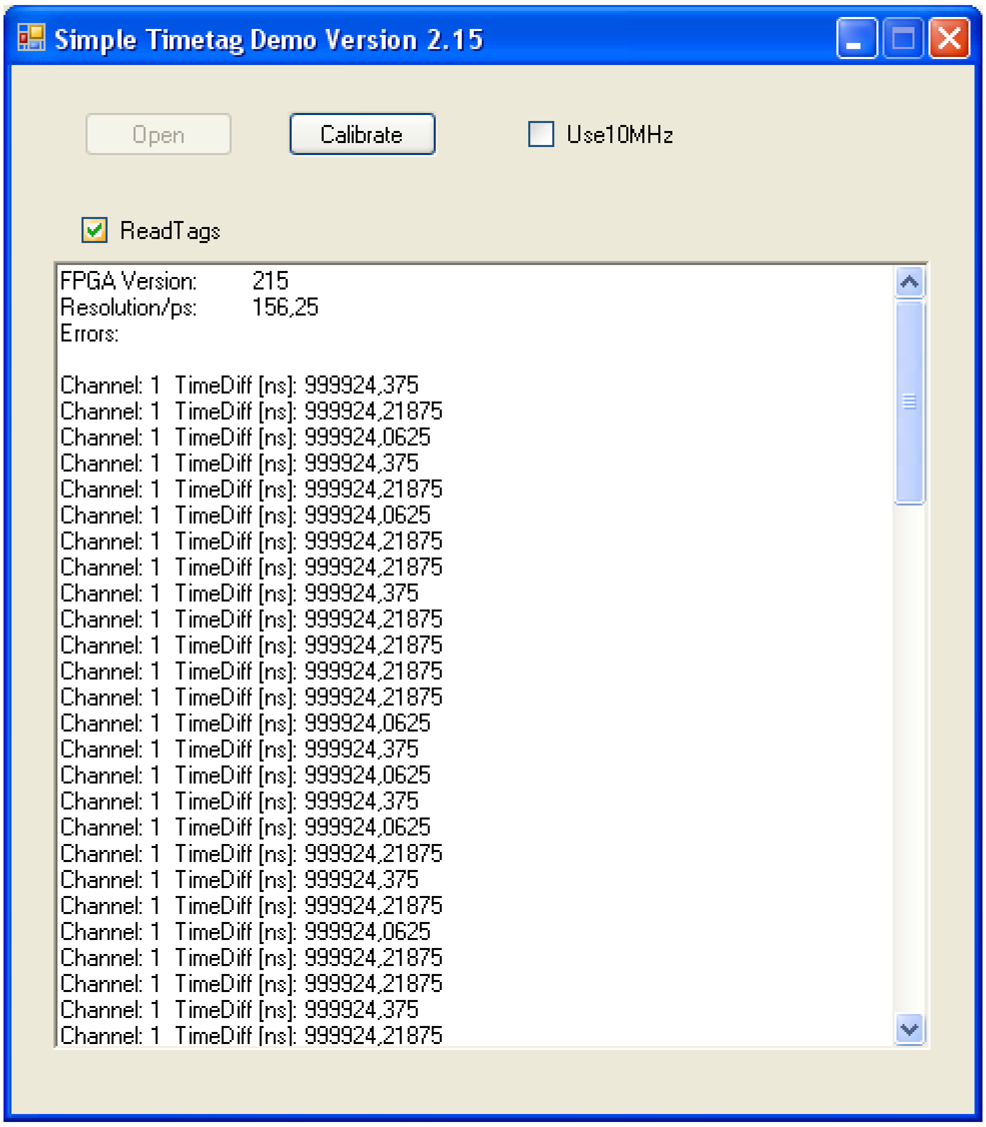
This simple program fragment shows how to use the interface:

```
ttInterface = new TTInterface(); //Create object
ttInterface.Open(1);              //Open device No 1
ttInterface.Calibrate();
ttInterface.StartTimetags();     //Start measurement

byte[] channels;
long[] times;
int count = ttInterface.ReadTags(channels, times);

ttInterface.StopTimetags();
ttInterface.Close();
```

On error conditions, an exception of type `TimeTag.UsbException ist thrown.`

How to use the .NET Demo Source Code

The demo source code is a very simple C# project that shows the basic usage of the time tag interface. The project file is “SimpleTimetagDemo.sln”. it can be opened with Microsoft Visual Studio 2010. The “Express” version of Visual Studio will also work.



```
FPGA Version:       215
Resolution/ps:      156,25
Errors:

Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,0625
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,21875
Channel: 1  TimeDiff [ns]: 999924,375
Channel: 1  TimeDiff [ns]: 999924,21875
```

C++

A C++ interface is available for Windows as well as Linux.

The Windows interface is compiles using Microsoft Visual Studio 2013.
The files are contained in the folder “CTimeTag”.

File	Description
include\CTimeTag.h	General header file
include\CLogic.h	Header file for Logic special functions
Win32\CTimeTagLib.lib Win32\CTimeTagLibDebug.lib	Windows 32 bit
Win64\CTimeTagLib.lib Win64\CTimeTagLibDebug.lib	Windows 64 bit
Linux\libtimetag32.so	Linux 32 bit
Linux\libtimetag64.so	Linux 64 bit

Simple Program Fragment

```
CTimeTag timetag;
timetag.Open(1);              //Open device No 1
timetag.Calibrate();
timetag.StartTimetags();       //Start measurement
unsigned char *chan;
long long *time;
int count= timetag.ReadTags(chan, time);

timetag.StopTimetags();
timetag.Close();
```

On error conditions, an exception of type `TimeTag::Exception ist thrown.`

How to use the C++ Demo Source Code

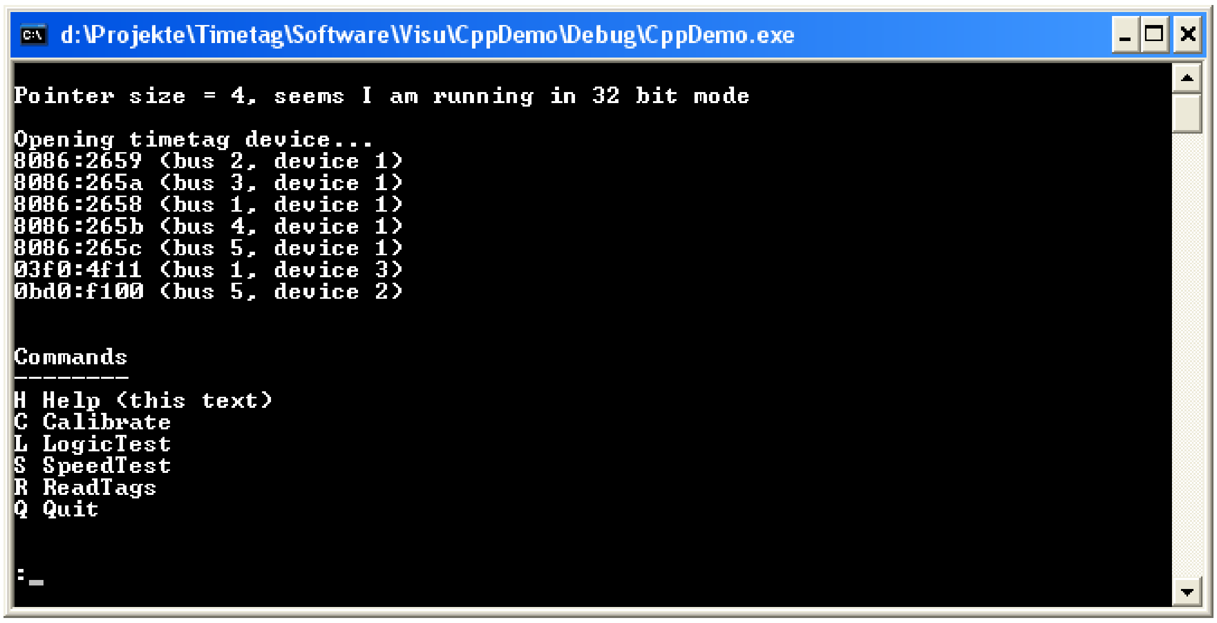
To use the demo program, just copy the two Folders “CppDemo” and “CTimeTag” into the same parent folder.

Windows:
On Windows, open CppDemo.sln using Visual Studio 2010 and compile.

Linux:
On Linux, copy libtimetag32.so or libtimetag64.so to your lib directory. If you have not already done so, get the g++ compiler and libusb development version.

```
sudo apt-get install g++-4.6-multilib
sudo apt-get install libusb-1.0-0-dev
```

Then navigate to the CppDemo folder and type “make CppDemo32” or “make CppDemo64”
To run the program, you need administrator privileges.


d:\Projekte\Timetag\Software\Visu\CppDemo\Debug\CppDemo.exe

```
Pointer size = 4, seems I am running in 32 bit mode

Opening timetag device...
8086:2659 (bus 2, device 1)
8086:265a (bus 3, device 1)
8086:2658 (bus 1, device 1)
8086:265b (bus 4, device 1)
8086:265c (bus 5, device 1)
03f0:4f11 (bus 1, device 3)
0bd0:f100 (bus 5, device 2)


Commands
--------
H Help (this text)
C Calibrate
L LogicTest
S SpeedTest
R ReadTags
Q Quit


:_
```

Basic Functions

The following description applies to all configurations of the device. They are written in C# syntax but the C++ methods have the exact same signature.

```
public void Open(int Number)
```

Number	>=1	The parameter is used only when more than one units are connected at the same time. If only one unit is used, this value must be set to 1

This function connects to the device. It has to be called before any other function is called.

```
public void Close()
```

This function should be called before the program terminates.

```
public void Calibrate()
```

The calibration function increases the accuracy of the device. It needs 4-10 seconds to execute.

When the calibration is used, it should be the first function to be called after the “Open” function.

```
public int ReadErrorFlags()
```

This function returns the internal error flags.
Calling this function clears the flags in the device.

Flag No	**Mask Value**	**Flag Name**	
0	1	`DataOverflow`	An overflow in the 512 k values SRAM FIFO has been detected. The time-tag generation rate is higher than the USB transmission rate.
1	2	`NegFifoOverflow`	Internal reason, should never occur
2	4	`PosFifoOverflow`	Internal reason, should never occur
3	8	`DoubleError`	One input had two pulses within the coincidence window.
4	16	`InputFifoOverflow`	More than 1024 successive tags were detected with a rate greater 100 MHz
5	32	`10MHzHardError`	The 10 MHz input is not connected or connected to a wrong type of signal.
6	64	`10MHzSoftError`	The 10 MHz input is connected, but the frequency is not 10 MHz.
7	128	`OutFifoOverflow`	Internal error, should never occur

8	256	`OutDoublePulse`	An output pulse was generated, while another pulse was still present on the same output. The pulse length is too long for the given rate.
9	512	`OutTooLate`	The internal processing was too slow. This is because the output event queue is too small for the given rate. Increase the value with `SetOutputEventQueue()`

```
public string GetErrorText(int flags)
```

This function translates the error flags to a short text that can be displayed on the user interface.

```
public int GetNoInputs()
```

This function returns the number of inputs installed on the device. It is used for debugging purposes only.

```
public double GetResolution()
```

This function returns the time resolution of the device. It should be used to calculate absolute time values.
The function returns either 78.125E-12 or 156.25E-12.

```
public void SetInputThreshold(int input, double voltage)
```

Input	[1..16]	Number of the input to change.
Voltage	[-2.0..2.0]	Threshold voltage in volt

```
public void SetInversionMask(int mask)
```

The inputs are edge sensitive. The positive edge is used as standard
With this function the relevant edge can be changed to the negative edge.
The mask is coded binary. When the corresponding bit is high, die negative edge is used.

```
public void SetDelay(int input, int delay)
```

All input signals can be delayed internally. This is useful to compensate external cable delays.

Input	[1..16]	Number of the input to change.
Delay	18 bit value	Delay in internal units. (See GetResolution())

```
public int GetFpgaVersion()
```

This function returns the current version of the FPGA design. It is used for debugging purposes only.

```
public void SetLedBrightness(int percent)
```

The Brightness of the LED on the front panel can be changed.

Time Tag Readout

The time-tag readout is an option not present in all devices.

In this mode, the internal ram is used as a fifo. 512 k tags can be stored in the device. The tags are transmitted to the computer by a background thread.
With this configuration very high transmission rates can be achieved.

At an internal fifo overflow, the `DataOverflow` flag is set and the error led on the front panel is lit.

The following functions are implemented in the class TimetagReader. You get a handle to the TimetagReader by calling TTInterface.GetReader();

```
public void StartTimetags()
```

This function puts the device into time-tag readout mode. The time tags are written into the internal fifo.
The background thread starts to read the time tags into the RAM of the computer.

```
public void StopTimetags()
```

The tags are no longer written into internal fifo.
The background thread is stopped.

```
public bool TagsPresent()
```

```
This function returns true when tags are ready to read.
This function is optional.
```

```
public int ReadTags(out byte[] channels, out long[] times)
(C# Syntax)
public: int ReadTags(unsigned char * channels, long long * times)
(C++ Syntax)
```

Return value: size of arrays
time[n] is the absolute time in internal units. The internal units are 78.125ps or 156.25 ps. The correct value is returned by `GetResolution()`.
Channel [n] is the number of the corresponding input. The first input is number 1.

The “virtual channel” 30 is used for the overflow tag. The overflow tag is sent whenever a data overflow error occurs. The data overflow tag indicates, that some data is missing at this point.

When there are no tags present, this function may wait up to 300 ms before it returns a zero result. When you don’t want this behavior, use TagsPresent() to check availability first.

The array is sorted by time. When two tags occur at the same time, then the tag with the smaller channel number is transmitted first.

The array is allocated by the driver software and returned to user code with an out parameter. Returning with an out parameter is similar to returning values by a pointer or reference in C / C++;

The arrays can be used until the next call of ReadTags. When the data is needed for a longer time it has to be copied to a different array.

Edge Sensitive Gating

This gating offers fine grained timing control and very low jitter. The width of the gating window can be adjusted in steps of the internal resolution.

The position of the gate can be adjusted too. This offers very flexible control of the gate.

The gate is opened a fixed time after the active edge of input 8.

This fixed time can be set with

```
    SetDelay(8,  delay);
```

The rising edge is the standard active edge.

This can be changed by `SetInversionMask();`

A negative gate delay is possible too. To achieve this, the delay of input 8 must be set to 0 and the delay of all other inputs must be set to the magnitude of the desired delay value.

The gate is open for a fixed time interval. This interval can be adjusted in in internal units (See below)

```
public void UseTimetagGate(bool use)
```

This function switches edge sensitive gating on or off.

```
public void SetGateWidth(int duration)
```

This function sets the width of the gate. The parameter duration is given in internal units.

Level Sensitive Gating

This feature is not present in all devices.

When the voltage on the gate Input is higher than the threshold voltage, then the device operates normally: The time tags of all inputs are stored in internal RAM and transmitted via USB.

When the voltage is below the threshold voltage, then the input signals are ignored and no tags are stored in internal RAM.

The threshold voltage is the voltage of channel 9.

This input has jitter and timing resolution of 5 ns.

```
public void UseLevelGate (bool)
```

False: Normal operation, the gate input will be ignored.
True: Level Gate Operation, tags are stored only when the gate input is high.

```
public bool LevelGateActive()
```

Returns true, when the gate input is above the input threshold

Time-Tag Filter

When the time-tag filter is on, the time-tags are transmitted only, when they appear in groups. This feature helps to reduce the USB bandwidth and PC load.

The filter is able to process rates of up to 190 MHz.

```
public void SetFilterMinCount (int MinCount)
```

This function defines the minimum size of a group to be transmitted.
MinCount can be set between 1 and 10 counts.
Setting MinCount to 1 switches the filter off, all tags are transmitted.

```
public void SetFilterMaxTime (int MaxTime)
```

MaxTime defines the maximum time between two pulses in the same group. When the time between two pulses is bigger than “MaxTime”, the two pulses are considered to be in different groups.
MaxTime ist given in internal units.

Example: When FilterMinCount is 10 and FilterMaxTime ist 1 us, then the maximum possible group size would be 9 us.

```
public void SetFilterExceptions (int Exceptions)
```

Some inputs can be excluded from the filter. (E.g. 1pps pulses.)
Excluded inputs are always transmitted. They do not participate in groups.
The filter exceptions are bit-coded. To exclude Input n from the filter, set the bit n-1 in the exception mask.
Examples:

0	No exceptions
1	Input 1 excluded
2	Input 2 excluded
3	Inputs 1 and 2 excluded
4	Input 3 excluded
.	.
.	.

10 MHz Input

The 10 MHz input can be used to increase the long-term stability of the device. It has to be switched on by software to be used.

```
public void Use10MHz(bool use)
```

When the 10 MHz input is switched on, but no valid signal is connected to the input, an error flag is set and the error led on the front panel is lit.

When using the 10 MHz reference, input 16 on the 156 ps devices and input 8 on the 78 ps devices can not be used.

Save to file

The software interface offers a high performance interface to save data to disk. The data is saved in a compressed, binary format and can be converted to ASCII offline. This way the full rate of 11 MHz can be saved to disk on a modern computer.

It is best explained by an example:

Saving:

```
TTInterface tti= new TTInterface();
tti.Open()
TimeTagReader r= tti.GetReader();

r.StartSaving(“rawdatafile.tt”);
//Wait some time
//Monitor r.SavedTags when you like
r.StopSaving();
tti.Close();
```

Converting

```
TTInterface tti= new TTInterface();
TimeTagReader r= tti->GetReader();
r.StartConverting(“rawdatafile.tt”, “tags.txt”);
r.WaitUntilConversionFinished();
```

For converting you don’t have to call Open() and you don’t even have to have a unit connected.

Of course you can do the conversion right after StopSaving().