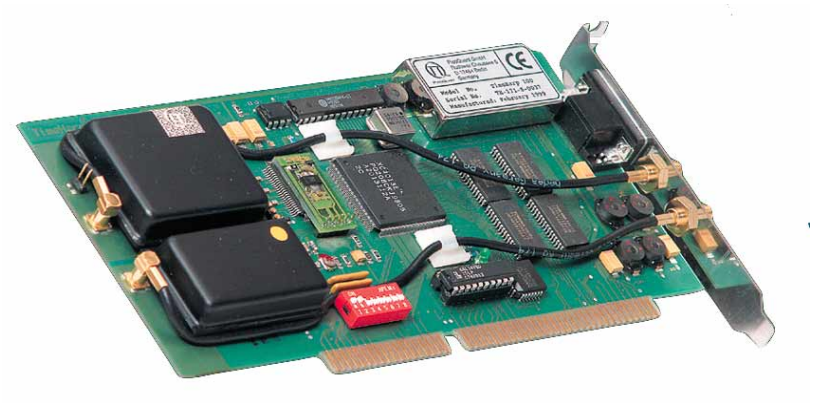
TimeHarp® 100

Time Measurement Histogram
Accumulating Real-time Processor

PicoQuant GmbH
Unternehmen für optoelektronische Forschung und Entwicklung

PC-board for Time-Correlated Single Photon Counting



THLIB.DLL Driver Library User's Manual

Version 3.0 - March 2001

Table of Contents

1. Introduction 3

2. General Notes 4

3. Firmware Update 5

4. Installation of the DLL 5

5. Running the Demo Applications 6

6. Using the DLL in Custom Applications 7

7. Data Types 7

8. Functions Exported by THLIB.DLL 8

9. Problems, Tips & Tricks 12

1. Introduction

The TimeHarp 100 is a compact and easy-to-use TCSPC system on a single PC board. It includes all components traditionally contained in bulky racks. A new solid state design keeps cost down, improves reliability and simplifies calibration. The circuit allows high measurement rates up to 3 Mcounts/s in continuous mode and provides a time resolution of less than 40ps. The input triggers (detector and sync) are programmable for a wide range of input signals. The detector input has a programmable Constant Fraction Discriminator (CFD). These specifications qualify the TimeHarp 100 for use with most common inexpensive single photon detectors such as Photomultiplier Tube (PMT) modules and Single Photon Avalanche Photodiodes (SPADs). The time resolution is well matched to these detectors and the overall Instrument Response Function (IRF) will usually not be dominated by the TimeHarp electronics. Similarly inexpensive and easy-to-use diode lasers such as the PDL 800 can be used as an excitation source well matched to the time resolution offered by the detector and the electronics. Overall IRFs of 300ps FWHM can be achieved with inexpensive PMTs and diode lasers. For information on the TimeHarp hardware and software please consult the TimeHarp manual. For details on the method of Time-Correlated Single Photon Counting, please consult our TechNote on TCSPC.

The monolithic TimeHarp standard software provides functions such as the setting of measurement parameters, display of measurement results, loading and saving of measurement parameters and measurement curves. Important measurement characteristics such as count rate, count maximum and position, histogram width (FWHM) are displayed continuously. While these features will meet many of the routine demands, advanced users may want to include the TimeHarp's functionality in their own automated measurement systems with their own software. In particular where the measurement must be interlinked or synchronised with other processes this approach may be of interest. For this purpose a driver library is provided as a Dynamic Link Library (DLL) for Windows 95/98/ME/NT and 2000 ™. This manual describes the installation and use of the TimeHarp driver library THLIB.DLL.

2. General Notes

This version of the TimeHarp driver DLL is suitable for Windows™ 95/98/ME/NT and 2000.

It has been tested with 32 bit Windows applications written in MSVC++4.0 to 6.0, Borland C++5.5 ™, Borland C++ Builder 3.0 ™ as well as with Delphi 4.0 ™, LabView 5.0 ™ and Visual Basic 6.0 ™.

This manual assumes that you have read the TimeHarp manual that came with your TimeHarp board. References to the TimeHarp manual will be made where necessary.

New in Version 3.0:

- TH_FPGAinit and TH_RegisterInit are now combined in TH_Initialize() which takes no parameters. The I/O address is obtained from the Windows Registry. (Maintained by the Device Manager)
- The library now also supports Windows NT/2000 which requires the driver THNTDIO.SYS to be installed and running and the device be configured in the Windows Registry.
- The Acquisition Time can now be up to 10h.
- Demo for LabVIEW added.

The library does not support TTTR measurements.

Users who purchased the library will receive free updates when they are available.

Data structures, program flow and function calls may change in future versions without advance notice.

Disclaimer

PicoQuant GmbH disclaims all warranties with regard to this software including all implied warranties of merchantability and fitness. In no case shall PicoQuant GmbH be liable for any direct, indirect or consequential damages or any material or immaterial damages whatsoever resulting from loss of data, time or profits arising from use or performance of this software. Demo code is provided ‘as is’ without any warranties as to fitnes for any purpose.

License and Copyright Notice

With this product you have purchased a license to use the TimeHarp 100 driver DLL on a single PC. You have not purchased the software itself. The software is protected by copyright and intellectual property laws. You may not distribute the software to third parties or reverse engineer, decompile or disassemble the software or part thereof. You may use and modify demo code to create your own software. Original or modified demo code may be re-distributed, provided that the original disclaimer and copyright notes are not removed from it.

TimeHarp is a registered trademark of PicoQuant GmbH.

Other trademarks related to operating systems, programming languages and their components are property of their respective owners and are used here for explanatory purposes only.

3. Firmware Update

The TimeHarp driver DLL requires a firmware update of your TimeHarp board, unless you already bought it with the DLL option installed.
The update is performed by DLLUPD.EXE. The update only needs to be done once.
If necessary, perform the following steps to install the update:

(see your TimeHarp manual for steps 1..3)

1. Make sure your TimeHarp board is installed correctly in the PC.

2. Check that the standard TimeHarp software runs correctly.

3. Make sure to exit the TimeHarp software.

4. Start the program DLLUPD.EXE directly from the floppy disk.

5. Follow the instructions. Do not interrupt the actual update progress, it may take a minute or so. The program will report successful completion.

You are then ready to use the TimeHarp DLL. See the sections below for hints how to install and use it. See the subfolder DEMO in your THLib installation folder for sample code.

4. Installation of the DLL

The TimeHarp driver DLL will not be installed by the standard TimeHarp software setup.
The standard interactive TimeHarp software does not require the DLL. It is provided for custom application programming only.

To install the DLL please run the setup program SETUP.EXE on the THLIB installation disk.
If you received the setup files as a ZIP archive, please unpack them to a temporary directory on your hard disk (or a floppy) and run Setup from there.
You need to register the board as a Windows device and install the board if you have not done so before (see your TimeHarp manual).

5. Running the Demo Applications

Please note that all demo code provided is correct to our best knowledge, however we must discaim all warranties as to fitness for a particular purpose of this code. It is provided 'as is' for no more than explanatory purposes.

The C and Delphi Demos

Consult the TimeHarp manual for setting up your TimeHarp board and establish a measurement setup that runs correctly and generates useable test data.
Save a Timeharp data file (*.thd) with the settings from this measurement run.

Place the file together with the test and demo files from this distributionin in a directory on your machine with the TimeHarp board installed. (Choose the files for your preferred programming language from the installation directories named correspondingly.)

You may now try the C or Delphi demo program DLLDEMO.EXE that comes with the DLL distribution.
Open a DOS box and change to the directory where you placed the files.
The usage of the demo is as follows:

```
dlldemo.exe inputfile outputfile
```

Inputfile is the .thd file you created, outputfile will be ASCII.

The demo will initilalise the TimeHarp board with the settings from the input file and then perform one or more measurement runs based on these settings.
The results of the last run will be written to the output file.

There is also a demo for continuous mode (contmode.c) which currently is provided in C only.

The Visual Basic Demo

The VB demo is simpler than the C and Delphi demos. It does not read in a pre-created thd file. There is no pre-compiled exe since you need to have VB installed on your machine anyway to get the VB runtime libraries. All measurement parameters are set directly in the VB code (DLLDEMO.BAS). However, you should run the standard TimeHarp program first to see if your hardware setup is correct. The otput from the demo run will be placed in the file DLLDEMO.OUT. This ASCII file will contain a single column of integer numbers representing the counts from the 4096 histogram channels. You can use any visualisation program to inspect the histogram.

The LabVIEW Demo

A demo VI (LabView 5.0) is provided in Demo\Labview\TimeHarp.llb. The top level VI is TimeHarp.vi. This is the most sophisticated demo, it closely resembles the standard TimeHarp software with input fields for all settable parameters. You need to have a running licencse of LabVIEW 5 or higher. Run the toplevel VI timeHarp.vi. It will first initilaize and calibrate the hardware. The status of initialization an calibration will be shown in the top left display area. Make sure you have a running TCSPC setup with sync and detector correctly connected. You can then adjust the sync level until you see the expected sync rate in the meter below. Then you can click the *Run* button below the histogram display area. The demo implements a simple *Oscilloscope mode* of the TimeHarp. Make sure to set an acquisition time of no more than e.g. a second, otherwise you will see nothing for a long time. If the input discriminator settings are correct you should see a histogram. You can stop the measurement with the same *Run* button.

6. Using the DLL in custom applications

C++ Users:

The provided C code in the CPP directory may be a useful starting point.
Consult thlib.h, thdefin.h and thlib.doc for reference.
In order to make the exports of thlib.dll known to the rest of your
application you may use thlib.exp or link directly with the import library thlib.lib.
MSCV++6.0 users can use the makefile (dlldemo.mak) or the project file (dlldemo.dsw)
where linking with thlib.lib is already set up.
With Borland C++ 5.5 and C++Builder 3.0 you can use the Borland Utility IMPLIB to create an import library very conveniently.

Delphi Users:

Delphi users refer to the DELPHI directory.
Everything for the demo is in DLLDEMO.DPR. For reference see CALLING.TXT.
If you update to a new DLL version please check the function parameters of your existing code against THLIB.H in the CPP directory.

LabVIEW Users:

You need to access the DLL routines via the 'Call Library Function' of LabVIEW. For details refer to the LabVIEW application note 088 'How to Call Win32 Dynamic Link Libraries (DLLs) from LabVIEW' from National Instruments. Consult thlib.h or the manual section further down for the parameter types etc. Make sure to specify the correct calling convention (stdcall).
A demo VI (LabView 5.0) is provided in Demo\Labview\TimeHarp.llb. The top level VI is TimeHarp.vi.

Visual Basic Users:

VB users should start their work from DLLDEMO.BAS. The code should be fairly self explanatory.
If you update to a new DLL version please check the function parameters of your existing code against THLIB.H in the CPP directory. The demo program uses a simple console for user I/O which can be replaced by the usual visual components to build a GUI. Note that where pointers to C-arrays are passed as function arguments you must pass the first element of the corresponding VB array by reference. In the VB function declaration you must declare the argument as the element type, not as an array.

7. Data Types

The TimeHarp DLL is written in C and the data types used correspond to standard C/C++ data types as follows:

char	byte or character in ASCII
int	32 bit signed integer
unsigned int	32 bit unsigned integer
long int	32 bit signed integer
unsigned long	32 bit unsigned integer
float	32 bit floating point

These types are supported by all major programming languages.

8. Functions exported by THLIB.DLL

See **thdefin.h** for predefined constants given in capital letters here.

```
int TH_GetLibraryVersion(char* vers);

arguments:     vers = pointer to a buffer for at least 7 characters

return value:  =0      success
               <0      error


int TH_GetBaseResolution();

arguments:     void

return value:  >0      base resolution of the installed board
               <0      error


int TH_GetHardwareVersion(char* vers);

arguments:     vers = pointer to a buffer for at least 7 characters

return value:  =0      success
               <0      error


int TH_GetBaseResolution();

arguments:     void

return value:  >0      base resolution of the installed board
               <0      error


int TH_Initialize();

arguments:     void

return value:  =0      success
               <0      error:          -2      error opening device driver (NT and 2000)
                                       -3      could not find device entry in registry
                                       -4      invalid I/O address entry in registry
                                       -5      error getting I/O permission (NT and 2000)
                                       -7      unsupported operating system
                                       <-8     hardware error

int TH_Calibrate();

arguments:     void

return value:  =0      success
               <0      error


int TH_SetCFDDiscrMin(int value);

arguments:     value =         CFD discriminator level in millivolts
                                       minimum = DISCRMIN
                                       maximum = DISCRMAX

return value:  =0      success
               <0      error


int TH_SetCFDZeroCross(int value);

arguments:     value = CFD zero cross level in millivolts
                       minimum = ZCMIN
                       maximum = ZCMAX

return value:  =0      success
               <0      error
```

```
int TH_SetSyncLevel(int value);

arguments:     value =        Sync level in millivolts
                                  minimum = SYNCMIN
                                  maximum = SYNCMAX

return value:  =0     success
               <0     error


int TH_SetRange(int range);

arguments:     value = Measurement range code
                               minimum = 0           (smallest, i.e. base resolution)
                               maximum = RANGES-1    (largest)

return value:  =0     success
               <0     error

note:          range code 0 = base resolution, 1 = 2 x base resolution and so on.


int TH_SetOffset(int offset);

arguments:     value =        Offset in nanoseconds
                                  minimum = OFFSETMIN
                                  maximum = OFFSETMAX

return value:  >=0 new offset
               <0 error

note:          The new offset is an approximation of the desired offset
               The typical step size is around 2.5 ns


int TH_NextOffset(int direction);

arguments:     value = direction of the desired offset change
                                  minimum = -1 (down)
                                  maximum = +1 (up)

return value:  >=0 new offset
               <0 error

note:          The offset changes at a step size of around 2.5 ns


int TH_ClearHistMem();

arguments:     void

return value:  =0     success
               <0     error


int TH_SetMMode(int mmode, int tacq);

arguments:     mmode = 0
                       must always be 0 in current version

               tacq =  acquisition time in milliseconds
                               minimum = ACQTMIN
                               maximum = ACQTMAX

return value:  =0     success
               <0     error


int TH_StartMeas();

arguments:     void

return value:  =0     success
               <0     error


int TH_StopMeas();

arguments:     void

return value:  =0     success
               <0     error
```

int TH_CTCStatus();

```
arguments:     void

return value:  0       acquisition time still running
               >0      acquisition time has ended
```

int TH_SetSyncMode();

```
arguments:     void

return value:  =0      success
               <0      error

note:          This function must be called before TH_GetCountRate()
               if the sync rate is to be measured. Allow at least 500ms to
               get a stable sync rate reading.
```

int TH_SetStopOverflow(int StopOnOfl);

```
arguments:     StopOn Ofl (0 = do not stop, or 1=do stop on Overflow)

return value:  =0      success
               <0      error

note:          This setting determines if a measurement run will stop if any channel
               reaches the maximum of 65535 counts.
```

long int TH_GetBlock(unsigned long *chcount);

```
arguments:     *chcount =     pointer to an array of double words (32bit) of BLOCKSIZE
                              where the histogram data can be stored

return value:  >=0     total number of counts in this histogram
               <0      error

note:          On most 32 bit systems int is the same as long int (32 bit)
               The distinction here is only for compatibility with the 16 bit version.
               The current version always counts only up to 65535 (16 bit).
               This may change in the future.
```

float TH_GetResolution();

```
arguments:     void

return value:  >0      resolution (channel width) in current range (from last calibration)
               <0      error
```

long int TH_GetCountRate();

```
arguments:     void

return value:  >=0     current count rate or sync rate (see note)
               <0      error

note:          The function returns either the current count rate if previously
               TH_SetMmode() was called or the current sync rate if previously
               TH_SetSyncMode() was called. Allow at least 500ms to get a stable
               rate meter reading in both cases.
```

int TH_GetFlags();

```
arguments:     void

return value:  current flag status (bit pattern)
               bit 0 (lsb)    = Overflow in histogram cell (65535)
               bit 1          = Sync error

note:          Currently only bit 0 is of significance to users.
               Use the predefined macros in thdefin.h ( e.g. FLAG_OVERFLOW )
               to extract individual bits through a bitwise logical AND.
               You can call this function anytime during measurement.
```

FUNCTIONS FOR CONTINUOUS MODE

int TH_GetBank(unsigned short *buffer, int chanfrom, int chanto);

arguments: *buffer - pointer to an array of words (16bit) of BANKSIZE
where the histogram data can be stored

chanfrom, chanto - limits of the channels to be fetched per histogram
these must be **EVEN NUMBERS**! (min 0, max 4094)

return value: sum of all counts in the channels fetched

note: See the C program demo **contmode.c** for how to use this function.

int TH_GetLostBlocks();

arguments: void

return value: number of lost blocks

note the lost blocks will be counted to a maximum of 255, then remain at 255
See the C program demo **contmode.c** for how to use this function.

9. Problems, Tips & Tricks

PC Performance Issues

Performance issues with the DLL are the same as with the standard TimeHarp software. The TimeHarp board and its software interface are a complex real-time measurement system demanding considerable performance both from the host PC and the operating system. This is why a reasonably modern, fast CPU (100MHz min.) and at least 32MB of memory are recommended. A PCI or AGP graphics interface is also strongly recommended. However, as long as you do not use continuous mode, these issues should be of little impact. If you do intend to use continuous mode you should also have a fast modern hard disk with UDMA mode enabled.

Device Access

The TimeHarp DLL will attempt to find the TimeHarp board in the Windows Registry. You need to make sure the device is registered ('Add Hardware' in the Windows Control Panel). Use the Windows Device Manager to verify the I/O address settings for the board. Make sure there is only one instance of the 'TimeHarp 100' installed. On Windows NT use the THNTCONF utility to configure the boards I/O address. Refer to your TimeHarp manual for trouble-shooting.

Version tracking

While PicoQuant will always try to maintain a maximum of continuity in further hardware and software development, changes in the interest of technical progress cannot always be avoided. It may therefore happen, that data structures, calling conventions or program flow will change. In order to design programs that will recognize such changes with a minimum of trouble we strongly recommend that you make use of the functions provided for version retrieval of hardware and DLL. In any case your software should issue a warning if it detects versions other than those it was tested with.

Other Issues

Note that the DLL will not allow to start two or more instances of any software using the DLL on one machine. However, it will not be aware of running instances of the standard TimeHarp software. However, you must ensure that you do not run the TimeHarp software at the same time you run your custom program. Since there is only one piece of hardware, this would lead to conflicts during hardware access. Observe the warming up and calibration recommendations given for the standard TimeHarp software also in your own developments.

Support and Bug Reports

The TimeHarp TCSPC system has now gone through three iterations of hardware improvement and several software updates. It has become very reliable indeed and many of you have already collected valuable experimental data. Nevertheless we would like to offer you our support in any case of trouble with the system. Do not hesitate to contact PicoQuant in case of difficulties with your TimeHarp or the DLL. Should you observe errors or bugs caused by the TimeHarp system please try to find a reproduceable error situation. E-mail a detailed description of the problem and relevant circumstances to *photonics@pq.fta-berlin.de.* Your feedback will help us to improve the product and documentation.

PicoQuant GmbH
Unternehmen für optoelektronische Forschung und Entwicklung
Rudower Chaussee 29 (IGZ), 12489 Berlin, Germany
Telephone: +49 / (0)30 / 6392 6560
Fax: +49 / (0)30 / 6392 6561
e-mail: photonics@pq.fta-berlin.de
WWW: http://www.picoquant.com

All information given here is reliable to our best knowledge. However, no responsibility is assumed for possible inaccuracies or omissions. Specifications and external appearence are subject to change without notice.