A. Fitting Functions

A.1. Gaussian

The mathematical description for a Gaussian pulse is:

$$G(t) = e^{-t^2} \qquad (A.1)$$

The autocorrelation of this pulse is given by the solution of the convolution integral

$$G_{ACF}(\tau) = \int_{-\infty}^{\infty} G(t)G(t-\tau)dt = \sqrt{\frac{\pi}{2}}\, e^{-\frac{\tau^2}{2}} \qquad (A.2)$$

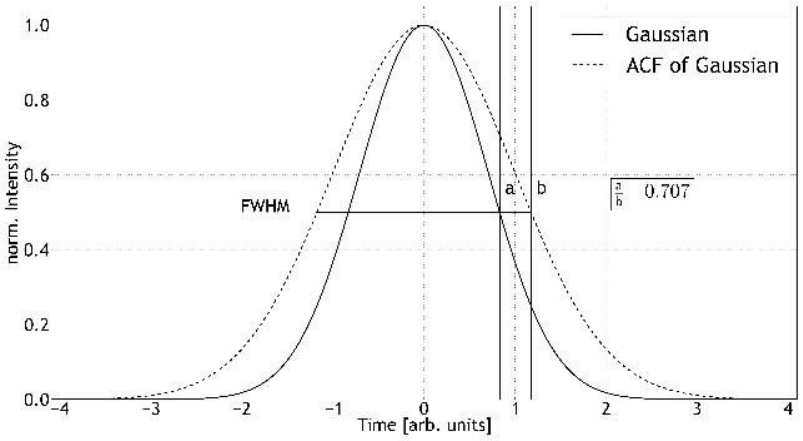

Figure A.1.: Gaussian function e^{-t^2} and its normalized autocorrelation $e^{-\frac{\tau^2}{2}}$ (dotted line)

Equating the normalized Gaussian functions with $\frac{1}{2}$ gives the time value at half amplitude.

$$G(a) = \frac{1}{2} \Rightarrow a = \sqrt{\ln(2)} \qquad (A.3)$$

$$G_{ACF}(b) = \sqrt{\frac{\pi}{2}} = \frac{1}{2} \Rightarrow b = \sqrt{2\ln(2)} \qquad (A.4)$$

The quotient of these time values supplies the transformation factor between the pulse width and the FWHM value of its autocorrelation function.

$$\frac{a}{b} = \frac{\sqrt{\ln(2)}}{\sqrt{2\ln(2)}} = 0.71 \qquad (A.5)$$

$$\int_{2\ln(2)}$$

A.2. lorentzian

The mathematical description for a Lorentzian pulse is:

$$L(t) = \frac{1}{1 + t^2}$$

The autocorrelation of this pulse is given by the solution of the folding integral

$$L_{ACF}(\tau) = \int_{-\infty}^{\infty} L(t)L(t-\tau)dt = \frac{2\pi}{4 + \tau^2}$$

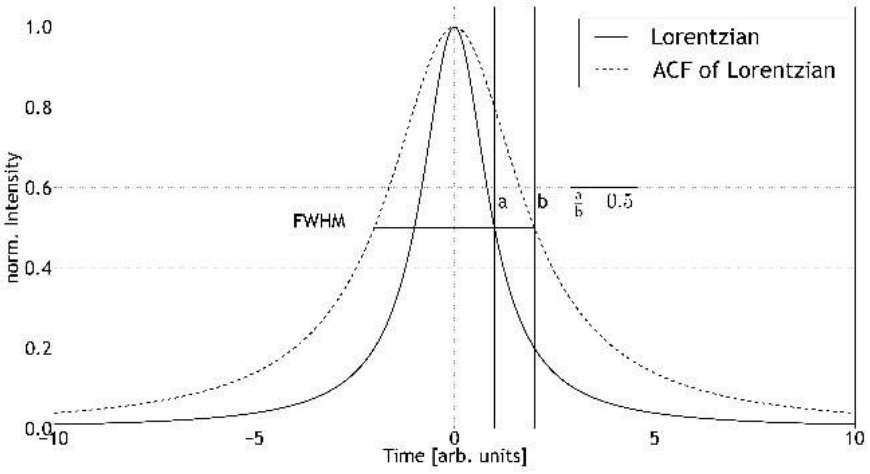

Figure A.2.: Lorentzian function $\frac{1}{1 + t^2}$ and its normalized autocorrelation $\frac{4}{4 + \tau^2}$ (dotted line)

Equating the normalized Lorentzian functions with $\frac{1}{2}$ gives the time value at half amplitude.

$$L(a) = \frac{1}{2} \quad \Rightarrow a = 1$$

$$L_{ACF}(b)\frac{2}{\pi} = \frac{1}{2} \quad \Rightarrow b = 2$$

The quotient of these time values supplies the transformation factor between the pulse width and the FWHM value of its autocorrelation function.

$$\frac{a}{b} = 0.5$$

A.3. sech^2

The autocorrelation of the $S(t) = \mathrm{sech}(t)^2$ pulse is given by the solution of its folding integral

$$S_{ACF}(\tau) = \int_{-\infty}^{\infty} S(t)S(t-\tau)dt = 4\mathrm{csch}(\tau)^3(\tau\cosh(\tau) - \sinh(\tau)) \qquad (A.3.1)$$

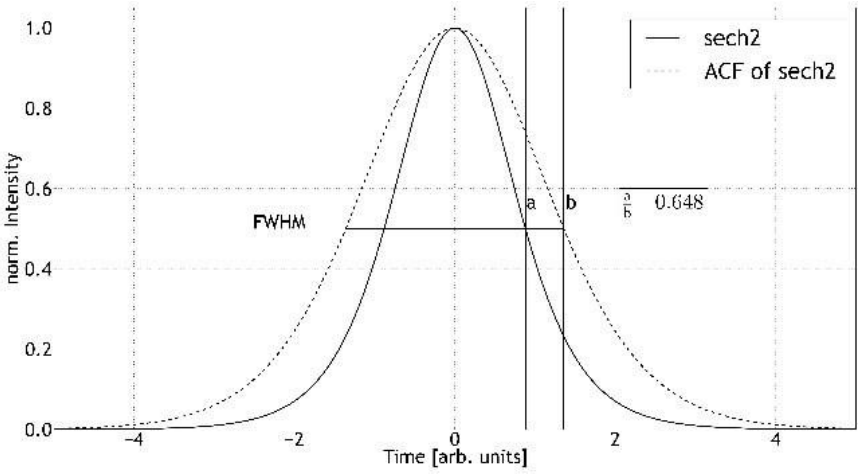

Figure A.3.: The function $\mathrm{sech}(t)^2$ and its normalized autocorrelation $3\mathrm{csch}(\tau)^3(\tau\cosh(\tau) - \sinh(\tau)$ (dotted line)

Equating the normalized functions with $\frac{1}{2}$ gives the time value at half amplitude.

$$S(a) = \frac{1}{2} \quad \Rightarrow a = 0.881374 \qquad (A.3.2)$$

$$L_{ACF}(b)\frac{3}{4} = \frac{1}{2} \quad \Rightarrow b = 1.35979 \qquad (A.3.3)$$

The quotient of these time values supplies the transformation factor between the pulse width and the FWHM value of its autocorrelation function.

$$\frac{a}{b} = 0.648 \qquad (A.3.4)$$

B. TCP/IP Command Set

This section provides a complete overview of the remote control commands of the **Mini TPA** . The command structure of the **Mini TPA** is mostly in agreement with the SCPl-standard. However, A·P·E does not state compliance nor conformance to the standard, since some standard commands are not yet implemented in the present version. Detailed information about the SCPl is found at: www.ivifoundation.org

For comprehensive usage and code examples in familiar programming languages such as C++, C#, labVIEW, Python, Matlab, and Ruby, please go to our webpage:

http://www.ape-berlin.de/en/software-interface-tcpip/

IMPORTANT NOTICE:
In order to remotly control the *mini*link the *mini*link sotware must run and a TCP/IP port must be set in the software. Do not attempt to send TCP/IP commands to the controller via USB!

The autocorrelator will execute the following commands:

***IDN?**

Get Device ldentification
<idn> string
Device lnformation (APE GmbH, Devicename, Serialnumber, Software Version, Firmware Version)

Example:

```
*lDN?
```

***RST**

Perform Device Reset

***STB?**

Get Status Byte
<stb> integer
SCPI Status Byte (8 Bit unsigned as decimal) | Bit0: reserved | Bit1: reserved | Bit2: Error | Bit3: reserved | Bit4: MAV | Bit5: ESB | Bit6: RSQ/MSS | Bit7: OPE

Example:

```
*STB?
```

***CIS**

Clear Status Byte (STB)

Example:

```
*CLS
```

***ESE<value>**

Set Event Status Enable Register
<value> integer, range: 0 ... 255
ESE Register Value

Example:

```
*ESE=127
```

***ESE?**

Get Event Status Enable Register
<value> integer, range: 0 ... 255
ESE Register Value

Example:

```
*ESE?
```

***SRE<value>**

Set Service Request Enable Register
<value> integer, range: 0 ... 255
SRE Register Value

Example:

```
*SRE=127
```

***SRE?**

Get Service Request Enable Register
<value> integer, range: 0 ... 255
SRE Register Value

Example:

```
*SRE?
```

***ESR?**

Get Event Status Register
<value> integer, range: 0 ... 255
ESE Register Value

Example:

```
*ESR?
```

***OPC?**

Get Operation Complete Status
<status> integer, range: 0 ... 255
OPC Stats Value (always "1", since multi-command interface is not available)

Example:

```
*OPC?
```

***OPER?**

Get Operation Status
<oper> integer
SCPl Operation Status (16 Bit unsigned as decimal ǀ Bit0: Disconnected ǀ Bit1: VlSA Connected ǀ Bit2: Device lnitialized ǀ Bit3: Device ready ǀ Bit4: Device busy ǀ Bit5: Standby (Delaymotor off) ǀ Bit6: Data Error (AFC not valid) ǀ Bit7: Software Error ǀ Bit8: Firmware Error (see *FRMW?) ǀ Bit9: Shutdown ǀ Bit10: Service Mode active ǀ Bit11: unused ǀ Bit12: unused ǀ Bit13: unused ǀ Bit14: unused ǀ Bit15: unused

Example:

```
*OPER?
```

***INIT?**

Get Device lnitialization Status
<init> integer
SCPl lNlT Status (8 Bit unsigned as decimal, upper 4 Bits are always "1" I Bit0: Config read OK I Bit1: Config parsing OK I Bit2: Link lnitialization OK I Bit3: Optic lnitialization OK I Bit4: unused, forced to "1" I Bit5: unused, forced to "1" I Bit6: unused, forced to "1" I Bit7: unused, forced to "1"

Example:

```
*lNlT?
```

***BUSY?**

Get Device Busy Status
<busy> integer
SCPl BUSY Status (8 Bit unsigned as decimal, upper 4 Bits are always "0" I Bit0: lDLE I Bit1: New data available I Bit2: Measurement running I Bit3: Curvefit running I Bit4: unused I Bit5: unused I Bit6: unused I Bit7: unused

Example:

```
*BUSY?
```

***ERR?**

Get Data Error Status
<errs> integer
SCPl DATA ERROR Status (8 Bit unsigned as decimal, Bit 7 is always "0" I Bit0: Signal too low I Bit1: Signal too high I Bit2: No Peak found I Bit3: ACF is asymmetric I Bit4: Dynamic range too low I Bit5: Scanrange too low I Bit6: Negative offset I Bit7: unused

Example:

```
*ERR?
```

***FRMW?**

Get Firmware Status
<frmw> integer
SCPl Firmware Error Status (16 Bit unsigned as decimal, Bit12..15 are always "0" I Bit0: Parser Error I Bit1: Parameter Error I Bit2: FRAM Error I Bit3: l2C-0 Error I Bit4: l2C-0 Error I Bit5: l2C Locked I Bit6: Configuration Error I Bit7: Optics Error I Bit8: Buffer Overflow I Bit9: DMA Error I Bit10: USB Error I Bit11: Data Timeout I Bit12: unused I Bit13: unused I Bit14: unused I Bit15: unused

Example:

```
*FRMW?
```

SYSTEM:DEVICE?
SYS:DEVICE?

Get Device Name
<name> string
Device Name

Example:

```
:sys:device?
```

SYSTEM:SNUMBER?
SYS:SNUMBER?

Get Device Serial number
<snr> string
Device Serial Number (S00000 - S99999)

Example:

```
:sys:snumber?
```

SYSTEM:SOFTWARE?
SYS:SOFTWARE?

Get Software Version
<version> string
Software Version

Example:

```
:sys:software?
```

SYSTEM:HARDWARE?
SYS:HARDWARE?

Get Hardware Version
<version> string
Hardware Version

Example:

```
:sys:hardware?
```

SYSTEM:FIRMWARE?
SYS:FIRMWARE?

Get Firmware Version
<version> string
Firmware Version

Example:

```
:sys:firmware?
```

SYSTEM:MOTOR?
SYS:MOTOR?

Get Motor Type
<version> string
Motor Type

Example:

```
:sys:motor?
```

SYSTEM:HElP?
SYS:HElP?

Get List of all SCPl Commands
<command_list> array of s in block data format
Command List as block data

Example:

```
:sys:help?
```

STATUS:AVERAGE<number>
STA:AVERAGE<number>

Set numer of measurements used for averaging
<number> integer, range: 0 ... 4
Number of Measurement | 0: Averaging OFF | 1: 2 Measurements | 2: 4 Measurements | 3: 8 Measurements | 4: 16 Measurements

Example:

```
:status:average 1
```

STATUS:AVERAGE?

STA:AVERAGE?

Get numer of measurements used for averaging
<number> integer, range: 0 ... 4
Number of Measurement | 0: Averaging OFF | 1: 2 Measurements | 2: 4 Measurements | 3: 8 Measurements | 4: 16 Measurements

Example:

```
:status:average?
```

STATUS:FITTYPE<type>
STA:FITTYPE<type>

Set type of curve fit to apply to measured ACF
<type> integer, range: 0 ... 3
Fittype | 0/OFF/NONE: No Curvefit, | 1/GAUSSIAN: Fit Gaussian Model, | 2/SECH2: Fit Sech2 Model, | 3/LORENTZ: Fit Lorentz Model

Example:

```
:status:fittype 1
```

STATUS:FITTYPE?
STA:FITTYPE?

Get type of calculated curve-fit
<type> integer, range: 0 ... 3
Fittype 0: No Curvefit | 1: Gaussian Model | 2: Sech2 Model | 3: Lorentz Model

Example:

```
:status:fittype?
```

STATUS:START?
STA:START?

Status of Measurement
<status> string
Status of measurement (1 = Measurement running, 0 = Measurement paused)

Example:

```
:status:start?
```

STATUS:FILTER<status>
STA:FILTER<status>

Set Status of ACF Filtering
<status> string

Example:

```
:status:filter 1
```

STATUS:FILTER?
STA:FILTER?

Get Status of ACF Filtering
<status> string
Status of ACF filtering (1 = filter active, 0 = filter not active)

Example:

```
:status:filter?
```

STATUS:TRIGGER<status>
STA:TRIGGER<status>

Toggle Triggermode
<status> string

Example:

```
:status:trigger
```

STATUS:TRIGGER?
STA:TRIGGER?

Get current Triggermode
<status> string
Triggermode (1 = active, 0 = inactive)

Example:

```
:status:trigger?
```

STATUS:MEASUREMENT?
STA:MEA?

Get Measurement
<status> string
Triggermode (1 = active, 0 = inactive)

Example:

```
:status:trigger?
```

STATUS:DETECTOR?
STA:DETECTOR?

Get PMT Detection Status
<status> string
PMT detected (1 = PMT found, 0 = no PMT)

Example:

```
:status:detector?
```

CRYSTAl:TUNING<number>
XTAl:TUN<number>

Set Crystal Position
<number> integer
Crystal Position

Example:

```
:xtal:tuning 1234
```

CRYSTAl:TUNING?
XTAl:TUN?

Get Crystal Position
<number> integer
Crystal Position

Example:

```
:xtal:tuning?
```

CRYSTAl:MOVE?
XTAl:MOV?

Get Status of Optics Movement
<number> integer
Optics Servo Status (1 - Servo moving, 0 - Servo not moving)

Example:

```
:xtal:mov?
```

MOTOR:SCANFREQUENCYNOAMP?
MOT:SFRNA?

Get ScanFrquency
<number> integer
ScanFrequency in Hz

Example:

```
:motor:sfrna?
```

MOTOR:SCANRANGE<scanrange>
MOT:SCR<scanrange>

Set ScanRange
<scanrange> integer, unit: fs
Scanrange 0/ZREO: Zeroscan | 1/150: 150 fs | 2/500: 500 fs | 3/1500: 1.5 ps | 4/5000: 5 ps | 5/15000: 15 ps | 6/30000: 30 ps (optional)

Example:

```
:motor:scr 15000
```

MOTOR:SCANRANGE?
MOT:SCR?

Get ScanRange
<scanrange> integer, unit: fs
Scanrange: 0: Zeroscan | 150: 150 fs | 500: 500 fs | 1500: 1.5 ps | 5000: 5 ps | 15000: 15 ps | 30000: 30 ps (optional)

Example:

```
:motor:scr?
```

DETECTOR:GAIN<value>
DET:GAIN<value>

Set PMT Gain value (Function has no effect, if standard optics units (w/o a PMT) is connected)
<value> integer, range: 300 ... 1000
PMT Gain Value

Example:

```
:detector:gain 450
```

DETECTOR:GAIN?
DET:GAIN?

Get PMT Gain value
<value> integer, range: 300 ... 1000
PMT Gain Value

Example:

```
:detector:gain?
```

DETECTOR:AUTOGAIN<number>
DET:AUG<number>

Activate Autogain Feature
<number> integer
Status of autogain feature

Example:

```
:detector:autogain 1
```

DETECTOR:AUTOGAIN?
DET:AUG?

Get Autogain status (0=OFF, 1=ON)
<number> integer
Status of autogain feature

Example:

```
:detector:autogain?
```

DETECTOR:SENSITIVITY<number>
DET:SEN<number>

Set Sensitivity
<number> integer
Detector Sensitivity ❙ 1: Low Sensitivity ❙ 10: High Sensitivity ❙ 100: (optional "HighSen"-Feature)

Example:

```
:detector:sensitivity 10
```

DETECTOR:SENSITIVITY?

DET:SEN?

Get Sensitivity
<number> integer
Detector Sensitivity | 1: Low Sensitivity | 10: High Sensitivity | 100: (optional "HighSen"-Feature)

Example:

```
:detector:sensitivity?
```

TRIGGER:lEVEl<level>
TRI:lVl<level>

Set Trigger Level
<level> integer, unit: mV, range: 200 ... 5000
Trigger Level

Example:

```
:trigger:level
```

TRIGGER:lEVEl?
TRI:lVl?

Get Trigger Level
<level> integer, unit: mV, range: 200 ... 5000
Trigger Level

Example:

```
:trigger:level?
```

TRIGGER:DElAY<level>
TRI:DEl<level>

Set Trigger Delay
<level> integer, unit: us, range: 1 ... 50
Trigger Delay

Example:

```
:trigger:delay
```

TRIGGER:DElAY?
TRI:DEl?

Get Trigger Delay
<level> integer, unit: us, range: 1 ... 50
Trigger Delay

Example:

```
:trigger:delay?
```

TRIGGER:FREQUENCY?
TRI:FRQ?

Get Trigger Frequency
<level> integer, unit: Hz
Trigger Frequency

Example:

```
:trigger:frequency?
```

TRIGGER:IMPEDANCE<level>
TRI:IMP<level>

Set Trigger lmpedance
<level> integer, unit: Ohms
Trigger lmpedance

Example:

```
:trigger:impedance
```

TRIGGER:IMPEDANCE?
TRI:IMP?

Get Trigger lmpedance
<level> integer, unit: Ohms
Trigger lmpedance

Example:

```
:trigger:impedance?
```

ACF:DATA?
ACF:RAW_DATA?

Get ACF Data
<acf> array of s in block data format
ACF Data as binary Block (little-endian byte order); The returned data holds an interleaved array of Double (lEEE754) Values with the following scheme [y0,x0,y1,x1,...,yN,xN], x = Delay (ps), y = ln-tensity (a.u.). ❙ Please note: the binary Data must be unpacked before it can be used. Please also see our example codes for more info about that.

Example:

```
:acf:data?
```

ACF:DISPlAYED_ACF?
ACF:DACF?

Get ACF Data as it is displayed (averaged/filtered/..)
<acf> array of s in block data format
Displayed ACF Data as binary Block (little-endian byte order); The returned data holds an interleaved array of Double (lEEE754) Values with the following scheme [y0,x0,y1,x1,...,yN,xN], x = Delay (ps), y = lntensity (a.u.) ❙ Please note: the binary Data must be unpacked before it can be used. Please also see our example codes for more info about that.

Example:

```
:acf:displayed_acf?
```

ACF:FIT_DATA?
ACF:FIT?

Get ACF Fit Data as it is displayed
<acf> array of s in block data format
Fitted ACF Data as binary Block (little-endian byte order); The returned data holds an interleaved array of Double (lEEE754) Values with the following scheme [y0,x0,y1,x1,...,yN,xN], x = Delay (ps), y = lntensity (a.u.) ❙ Please note: the binary Data must be unpacked before it can be used. Please also see our example codes for more info about that.

Example:

```
:acf:fit?
```

ACF:MEANDATA?

Get ACF Mean Data
<mean> string
Mean Values separated by semicolons: [AVG];[Xmax];[Xmin];[Ymax];[Ymin]

Example:

```
:acf:meandata?
```

ACF:FWHM?

Get FWHM Value
<fwhm> double
FWHM Value

Example:

```
:acf:fwhm?
```

ACF:FITFWHM?

Get fitted FWHM Value
<fwhm> integer
Fitted FWHM Value. (Note: Prior to V1.0.2.255 this value does not include the correction factor. Please multiply the value with 0.71 for gaussian, 0.5 for lorentz and 0.648 for sech2 fits to get the correct value)

Example:

```
:acf:fitfwhm?
```

ACF:FIT_COEFF?
ACF:FITC?

Get fit curve parameters
<fwhm> integer
Parameters calculated by the fitting algorithm, separated by semi-colons: [Amplitude];[X-Shift];[FWHM];[Y-Shift]

Example:

```
:acf:fit_coeff?
```

SERVICE:MODE?

Get current operation mode
<mode> integer
State of Service Mode (0 = Default mode, 1 = Service mode)

Example:

```
:serivce:mode?
```

SERVICE:CONFIG?

Get content of the local configuration/calibration file (Service Mode required)
<cfg> array of s in block data format
Content of the local configuration file

Example:

```
:serivce:config?
```

SERVICE:DEV_CONFIG?

Get content of the device configuration/calibration file (Service Mode required)

<cfg> array of s in block data format
Content of the device configuration file

Example:

```
:serivce:dev_config?
```

C. Declaration of Conformity

We declare that the accompanying product, identified with the CE mark, complies with requirements of the Electromagnetic Compatibility Directive, 2004/108/EC dated December 15, 2004 and the Low Voltage Directive 2006/95/EC dated December 12, 2006.

Product name: **Mini TPA**
Product options: **all options**

CE mark affixed: Berlin, December 3, 2014

Type of Equipment: Electrical equipment for measurement, control and laboratory use in industrial locations.

Manufacturer: A·P·E Angewandte Physik & Elektronik GmbH Berlin
Plauener Straße 163-165
13053 Berlin, Germany

Standards Applied:

Compliance was demonstrated to the following standards to the extent applicable: BS EN 61010-1:2010, "Safety requirements for electrical equipment for measurement, control and laboratory use"

EN 55011 radio interference voltage class A
DIN EN 61000-4-2:2009
DIN EN 61000-4-3:2011
DIN EN 61000-4-4:2010
DIN EN 61000-4-5:2007
DIN EN 61000-4-6:2009
DIN EN 61000-4-8:2010
DIN EN 61000-4-11:2005

Name (printed): Dr. Bodo Richter
Title: CEO

Telephone: +49 30 98601130
Email: ape@ape-berlin.de

D. Declaration of Conformity to EU RoHS

A·P·E Angewandte Physik & Elektronik GmbH
Plauener Str. 163 - 165 I Haus N
13053 Berlin
Germany

Declaration of Conformity to EU RoHS

Products listed below that are manufactured by A·P·E Angewandte Physik & Elektronik GmbH are in compliance with Directive 2011/65/EU of the European Parliament and of the Council of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment (also known as "RoHS Recast"). ln addition, this declaration of conformity is issued under the sole responsibility of A·P·E Angewandte Physik & Elektronik GmbH. Specifically, products manufactured do not contain the substances listed in the table below in concentrations greater than the listed maximum value.

Substance	Maximum limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000

Product ldentification:
Product

Autocorrelator **Mini TPA** PD-UV
A·P·E ld: 150656

Signature:

Name (printed): Dr. Bodo Richter
Title: CEO

Telephone: +49 30 98601130
Email: ape@ape-berlin.de