

Thin Film Measurement Programming Library TF++ *lite*

For developing your own film thickness measurement applications using the tec5 spectrometer system, we provide our powerful and very easy-to-use programming library TF++*lite*.

With TF++*lite* the entire spectral data acquisition, like scanning the diode array, raw data averaging, dark current correction and the spectra normalization is fully encapsulated in just a few simple function calls. The measured interference spectra will be evaluated in real-time for either single or double layer film thickness using the same high precise Fast-Fourier Transform (FFT) algorithm as the TF*ProVis* 2000 *lite* software packages.

TF++*lite* gives you full access to all measured spectral data (including the raw data), the computed FFT spectrum and film thickness results and lets you create the so-called Spectra-Recorder files, which can be viewed and re-processed using TF*ProVis* *lite*. By this you can easily check all measurements executed with your application and TF++*lite*.

- Runtime licensed Dynamic Link Library (DLL) providing standard C calls
Compatible with common C/C++ compiler, Delphi, Visual Basic and VBA
(Excel)
- Extensive Parameter Checks and Measurement Status Verify
You hardly can do anything wrong when working with TF++*lite*
- Detailed User's Manual as compiled HTML file and PDF document
- Demo software as Windows console application, including C/C++ source code

Please see next page for a programming example!

Minimum Hardware and Software Requirements

- § PC with at least Pentium 3 processor (Pentium 4 with 1.8GHz or higher recommended)
 - § Windows 2000 or Windows XP operating system (NT not supported)
 - § C/C++ development system (MS Visual Studio recommended), Delphi, Visual Basic or VBA
 - § tec5 / tec5USA spectrometer system
 - § CD ROM drive for installation
 - § Graphics adapter with 1024 x 768 pixels (1280 x 1024 recommended)
-

Features and Specifications

Programming Example

```
// Step 1: open and initialize
spectrometer
FTMLITE_SPECHARDWARE
sSpecHardwareInfo;
FTMLite_OpenSpectrometer( FTMLITE_TRANSPEC_LITE, &sSpecHardwareInfo );

// Step 2: setup measurement
parameter:
FTMLITE_MEASPARA
sMeasPara;
sMeasPara.dIntegrationTime = 20.0; // 20 ms integration time
sMeasPara.bEnableAverage = 1; // averaging on
sMeasPara.iNumberAverage = 10; // 10 scans for averaging
FTMLite_SetMeasPara( &sMeasPara );// notify settings to spectrometer

// Step 3: perform measurement of an averaged Dark Current
FTMLite_CloseShutter(); // close shutter of connected lamp
FTMLite_RunMeasDarkCurrent(); // start
measurement FTMLITE_SPECSTATUS sSpecStatus;
FTMLite_GetSpecStatus( &sSpecStatus ); // wait until
measurement is done while ( sSpecStatus.bRunDarkCurrent )
FTMLite_GetSpecStatus( &sSpecStatus );

// Step 4: perform measurement of an averaged and Dark Current corrected Reference spectrum
FTMLite_OpenShutter(); // open shutter of connected lamp
FTMLite_RunMeasReference(); // start measurement
FTMLite_GetSpecStatus( &sSpecStatus ); // wait until
measurement is done while ( sSpecStatus.bRunReference )
FTMLite_GetSpecStatus( &sSpecStatus );

// Step 5: setup film thickness evaluation parameter
(simple example) FTMLITE_EVALPARA sEvalPara;
sEvalPara.bSpecEvalRangeFull = 1; // use entire interference spectrum for evaluation
sEvalPara.bPeakSearchRangeFull = 1;// search entire FFT spectrum for peak
sEvalPara.dRefIndex = 1.56; // refraction index of the
layer FTMLite_SetSingleLayerEvalPara( &sEvalPara ); // initialize
single layer evaluation

// Step 6: measure and evaluate an averaged and Dark Current corrected interference spectrum
FTMLite_RunMeasInterference(); // start measurement
FTMLite_GetSpecStatus( &sSpecStatus ); // wait until
measurement is done while ( sSpecStatus.bRunInterference )
FTMLite_GetSpecStatus( &sSpecStatus );
FTMLITE_RESULT sResult;
FTMLite.EvalSingleLayer( &sResult ); // evaluate interference spectrum

// Done! Aside from other information, the structure sResult now contains:
sResult.dThickness // the film thickness in microns
sResult.blPlausible // thickness seems to be plausible or not
sResult.sDateAndTime // the date and time (microsecond resolution) of the measurement
```

§