Novoptel

Application note 2
Automated PDL Measurement with EPS1000

Revision history

Version	Date	Remarks	Author
0.9.1	12.09.2013	Draft version	B. Koch
1.0	26.06.2015	Updated due to firmware rev. 1.0.6.0 (release 15.06.2015)	B. Koch

Contents

Summary

This Application Note describes the automated PDL (Polarization Dependent Loss) measurement of a passive optical DUT (Device Under Test) with direct EPS1000 register access via USB or SPI. Two measurement techniques are covered that do not need an external polarimeter. Unless the EPS1000 scrambler is equipped with the optional internal power meter, an external power meter is required.

Novoptel GmbH
EIM-E

Available PDL measurement techniques

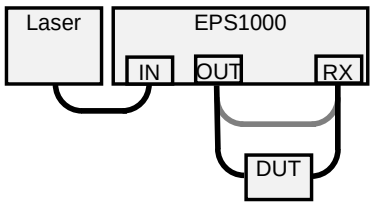

Fig. 1: Setup for PDL measurement with EPS1000 scrambler. The optional power meter input is shown as RX port. The scrambler output is directly connected to a power meter for a subsequent reference measurement.

Extinction method:
A power meter behind a device under test (DUT) can be utilized to drive a polarization scrambler/ transformer in front of the DUT into the states where minimum and maximum transmissions are reached. From minimum and maximum transmission, polarization dependent loss (PDL) can be calculated. This method is referred to as the *extinction method*.

Scrambling method:
With a Novoptel EPS1000 polarization scrambler/transformer, one can generate polarization states that are equidistributed on the Poincaré sphere. This *scrambling method* as described in [1] allows measuring especially small PDL with higher accuracy. Because the power samples have to be strictly synchronous to the polarization states alignment, this method requires an EPS1000 model with optical, low-PDL receiver input, e.g. part number EPS1000-10M-XL-S-AA-**O**-D.

Removing scrambler PDL:
PDL of the scrambler will be superimposed to the PDL of the DUT. To get more precise results, we propose to repeat the measurement while output and RX port of the EPS1000 are connected with a patch cord. The samples in the reference measurement should be taken at the same SOPs (States of Polarization) behind the scrambler as in the DUT measurement because this allows removing the scrambler PDL from the DUT PDL mathematically. So, the EPS1000 should be configured to provide the same polarization transformations in both measurements, and the input polarization of the EPS1000 should be fixed. This can be done with both the extinction and the scrambling method.

With the optional optical receiver input, both the extinction and the scrambling method including the reference measurements are supported by the Novoptel GUI as described in the EPS1000 User Guide. The following descriptions shall improve understanding of the involved EPS1000 control registers to enable automated PDL measurement from any platform. Suitable Matlab code is provided. It can be treated as pseudo code and translated into other languages, e.g. Python or Labview.

Dark Current Measurement

Both methods require knowledge of the dark current, i.e. the ADC value at zero optical input power. To get a most accurate measure of the dark current, it is useful to increase the averaging time for this ADC sample. The averaging time is T = 80 ns • 2^{ATE}, where ATE is the Averaging Time Exponent (ATE) as described in the EPS1000 User Guide.

Two basic communication functions are used for USB transfer to and from a connected EPS1000: `writeusb(x, y)` will write a value y into register x, and `readusb(z)` returns the data contained in register z. For details of this functions please refer to the EPC1000 or EPS1000 User Guide. The code can be transferred into any language that allows data transfer using the FTDI USB driver.The new ATE value is set in Matlab by

```
writeusb(129, ATE);
```

Sometimes it is useful to read not only the current intensity sample but also the fractional part of the averaged value. To get both, the Matlab code is:

```
dc = readusb(128) + readusb(133)/2^16;
```

Any read of register 128 will freeze the fractional part in register 133. Hence it is important to read register 128 before register 133 to get integer and correct fractional part of the same sample.

Extinction method

The extinction method algorithm has two basic elements: One is to read the current optical power. This can be done as described in the previous section. The other is the modification of the scrambler's polarization transformation. For this purpose, all 16 electrode voltages of the polarization transformer can be accessed through the USB/SPI registers. The command

```
writeusb(50+isect-1, 8192+dacvalue);
```

writes the value *dacvalue* plus an offset of 8192 in the DAC of electrode number *isect*. The value *dacvalue* is limited to ±6000 for all electrodes, which corresponds to ±45 Volt. The algorithm must modulate the electrode voltages, read the optical power and adjust the current working point until the global transmission minimum or maximum is reached.

For the reference measurement, the voltage sets for minimum and maximum transmission should be saved. With a patch cord instead of the DUT, the same two voltage sets can be applied to measure the scrambler PDL influence on the measured DUT PDL. Between and during DUT and reference measurement, the input polarization to the EPS1000 must be stable.

The PDL value in dB is calculated by

```
PDL_dB = 10*log10(I_max / I_min)
```

where *I_max* and *I_min* are the (averaged) ADC samples at maximum and minimum transmission. From both values, the dark current value has been subtracted previously.

Scrambling method

Synchronous / Triggered Waveplate Rotations

For this method, the goal is to generate a reproducible set of polarization states that are equally distributed over the Poincaré sphere. During polarization scrambling, this is obtained by endlessly rotating waveplates. In contrast to the normal scrambling mode, we use a trigger here to keep the electrode voltages fixed during the sampling time. This is what we call a *synchronous* or *triggered waveplate rotation*. In this mode, the generated polarization states are reproducible if

a) the waveplates rotate with the same speeds and start at the same predefined orientations,

b) the same internal (synchronous) trigger is used and

c) the scrambler's input polarization does not change between two measurements.

This technique does not require predefined execution tables, thus allowing to use a maximum of 2^{16} measurement positions when the internal sampling memory is used.

Configuration from Matlab

When controlling the EPS1000 from Matlab, the synchronous / triggered waveplate rotations can be configured by the following code.

```
writeusb(126, 0); % Use only Photodetector 1
writeusb(229, 0); % Disable Synchronous Table Execution
writeusb(224, 0); % Disable Ext Trigger
writeusb(220, 0); % Disable Continuous Table Execution
writeusb(225, 0); % Disable ATE Trigger; Disable INT Trigger
writeusb(132, 1); % Enable Timed Output
writeusb(129, ATE); % ATE=11: averaging over 2^11 samples * 80 ns = 163.84 us
```

```
writeusb(137, ATE+1); % Sample recording trigger is 2^(11+1) * 80 ns= 327.68 us
writeusb(134, 2^15-1); % max. memory address: 2^16-1
writeusb(136, 0); % any delay in multiples of 20 ns
writeusb(140, 0); % any delay in multiples of the waveplate trigger period
writeusb(141, 0); % 0/1/2/3: 1/2/4/8 samples stored per waveplate position
```

If the DUT induces a long signal delay, this delay can be taken into account by using the delay registers 136 (multiples of 20 ns) and 140 (multiples of the sample recording trigger period). If the DUT delay is unknown but less than a trigger period, 2, 4 or 8 samples can be stored per waveplate position by writing 1,2 or 3 to the oversampling register 141. The collected data can then be split to 2,4 or 8 measurements with different delays each. Since a wrong delay leads to a lower calculated PDL value, the measurement with the highest PDL value has to be chosen.

During the measurement process, the scrambler will fill the sampling memory from address 0 up to the address defined in the memory address register 134. The maximum possible value for this register is 2^{16}-1.

Register 129 defines the averaging time exponent (ATE) of the ADC sampling. The averaging time is 2^{ATE} * 80 ns, which leads to an averaging time of 163.84 µs at ATE=11. With the delay register 136 set to 0, the setting of a new waveplate position and the writing of an averaged ADC sample occur simultaneously. Thereby, the averaged ADC value for the old waveplate position is stored. The waveplate trigger period register (137) is programmed the same way as the ATE register. Normally, the waveplate trigger period should be twice the averaging time. This way, only samples in the second half of the waveplate trigger period contribute to the averaged value, which increases accuracy. This is achieved by writing “ATE+1” to register 137.

Waveplate	Turns	Steps	Start at
QWP0	2^2	2^{13}	1/48
QWP1	2^6	2^9	3/48
QWP2	2^{10}	2^5	5/48
HWP	2^{12}	2^3	0
QWP3	2^8	2^7	7/48
QWP4	2^4	2^{11}	9/48
QWP5	2^0	2^{15}	11/48

Table. 1: Electrooptic waveplate turns and steps in each measurement with 2^{15} samples, and exemplary starting positions. The inverse of the value “Steps” is used for waveplate speed setting.

The initial positions of the waveplates may help to randomize polarizations. They are defined and applied to the EPS1000:

```
QWP0position = round( 1/48 * 2^16);
QWP1position = round( 3/48 * 2^16);
QWP2position = round( 5/48 * 2^16);
HWPposition  = round( 0/48 * 2^16);
QWP3position = round( 7/48 * 2^16);
QWP4position = round( 9/48 * 2^16);
QWP5position = round(11/48 * 2^16);

writeusb(40, HWPposition);
writeusb(41, QWP0position);
writeusb(42, QWP1position);
writeusb(43, QWP2position);
writeusb(44, QWP3position);
writeusb(45, QWP4position);
writeusb(46, QWP5position);
```

The waveplate rotation speeds have to be set individually. We project a measurement of 2^15 samples and request that all waveplates perform integer numbers of rotations during the measurement, while the generated polarizations are as randomly distributed on the Poincaré sphere as possible. Starting from different QWP and HWP orientations (see above) all combinations of 4 equispaced QWP

orientations and 8 equispaced HWP orientations ($4^3 \cdot 8 \cdot 4^3 = 32768$) are applied in our example. See also Table 1.

```
QWP0speed = round(  2*pi / 2^13/ (2*2^ATE*80e-9) * 100);
QWP1speed = round(  2*pi / 2^9 / (2*2^ATE*80e-9) * 100);
QWP2speed = round(  2*pi / 2^5 / (2*2^ATE*80e-9) * 100);
HWPspeed  = round(2*2*pi / 2^3 / (2*2^ATE*80e-9) * 100 / 1000);
QWP3speed = round(  2*pi / 2^7 / (2*2^ATE*80e-9) * 100);
QWP4speed = round(  2*pi / 2^11/ (2*2^ATE*80e-9) * 100);
QWP5speed = round(  2*pi / 2^15/ (2*2^ATE*80e-9) * 100);

writeusb(9, mod(HWPspeed, 2^16));
writeusb(10, bitshift(HWPspeed, -16));
writeusb(11, mod(QWP0speed, 2^16));
writeusb(12, bitshift(QWP0speed, -16));
writeusb(13, mod(QWP1speed, 2^16));
writeusb(14, bitshift(QWP1speed, -16));
writeusb(15, mod(QWP2speed, 2^16));
writeusb(16, bitshift(QWP2speed, -16));
writeusb(17, mod(QWP3speed, 2^16));
writeusb(18, bitshift(QWP3speed, -16));
writeusb(19, mod(QWP4speed, 2^16));
writeusb(20, bitshift(QWP4speed, -16));
writeusb(21, mod(QWP5speed, 2^16));
writeusb(22, bitshift(QWP5speed, -16));
```

If the scrambler has a firmware 1.0.6.0 or later, the rotation speeds can alternatively be given by rotations per 10.7 s instead of rad/s, which is more accurate. 10.7 s is the measurement time when recording 2^{15} samples at 327.68 µs per sample. If a smaller memory size or a lower ATE is selected, the following values would have to be increased to ensure integer numbers of rotations within the measurement time.

```
writeusb(150, 1); % define rotations per 10.7 s instead of rad/s
writeusb(151, 4096);
writeusb(152,    4);
writeusb(153,   64);
writeusb(154, 1024);
writeusb(155,  256);
writeusb(156,   16);
writeusb(157,    1);
```

The rotations have to be configured. We choose the forward direction:

```
%enable and set forward direction
  writeusb(0, 1); % HWP:
  writeusb(1, 1); % QWP0:
  writeusb(2, 1); % QWP1:
  writeusb(3, 1); % QWP2:
  writeusb(4, 1); % QWP3:
  writeusb(5, 1); % QWP4:
  writeusb(6, 1); % QWP5:
```

After the configuration described above, a measurement can be started by the command

```
writeusb(225, 2); % Enable ATE Trigger; Disable INT Trigger
```

Register 135 contains the current memory address (Bits 0 to 15). It can be read to observe the measurement process. If the maximum address 2^{16}-1 is used, also Bit 16 of the memory address (Bit 0 of register 139) has to be observed because after writing to address 2^{16}-1, the counter will stop at address 2^{16}. Once the address has reached the defined maximum address plus 1, the trigger can be disabled by writing “0” to register 225. Disabling the trigger will reset the address counter.

Synchronous / Triggered Table Execution

The requirement of equidistribution on the Poincaré sphere can not only be met with rotating waveplates, but also with a limited number of predefined polarization state sequences. Examples for such sequences are the corners of diamonds, cubes and other polyhedrons, which can be stored in and executed from a table. However, a polarimeter is needed to find these polarization states by

suitable voltage settings of the EPS1000. The synchronous / triggered table execution is also useful for various other applications such as recirculating-loop experiments.
The internal execution table (= Table) stores all 7 waveplate positions and all 16 electrode voltages for each of up to 1024 settings (= Current Table Position). Electrode voltage settings are modified appropriately upon waveplate position setting by the user, but not the other way round. During table execution only the electrode voltage settings matter and the (also stored) waveplate positions are irrelevant. This makes sense, given that electrode voltage settings generally produce waveplates other than quarter- or halfwave ones, i.e. with other retardations. Waveplate position setting is still useful for initial table generation. But a Table can be generated just as well by electrode voltage settings. Nevertheless, the waveplate positions will always be kept as a part of the Table to ease a later interpretation of the polarization transformations.

Samples of the optical power can be taken at the end (= after application) of each table step. This can be done externally, preferably timed with the trigger input or output. When an optical receiver input is provided by the EPS1000 model, samples of the detector are taken and stored directly before the next table step is executed. The averaging time for these measurements is given by $T = 80\ ns \bullet 2^{ATE}$, where ATE is the Averaging Time Exponent (ATE) that can be set in the GUI or by writing to register 129. The internal memory for measurement data can store all 1024 samples, e.g. for a complete table execution.

Creating tables using text files and GUI

Each row of the text file corresponds to one polarization transformation setting. Rows are strings containing 24 integers, separated by commas. The first 7 integers are the waveplate positions in sequence QWP0, QWP1, QWP2, HWP, QWP3, QWP4 and QWP5. The positions are given as a rotation variable in the unit interval (≥0, <1), where 1 would mean a full waveplate rotation (by 360° electronically or 180° for an equivalent physical waveplate), multiplied by 2^{16} and rounded to the nearest integer.

The next 16 integers in the row define the electrode voltages. An offset of 8192 corresponds to 0 Volt, the upper and lower limitations of 8192±6000 correspond to ±45 Volt. The first two elements correspond to electrodes 1 and 2 of the first of eight sections of the $LiNbO_3$ polarization transformer. Elements 3 and 4 correspond to electrodes 1 and 2 of the second section and so on.

The last integer in the row defines the dwell time (= subsequent delay) in nanoseconds. The dwell time must be a multiple of 40 ns. Minimum dwell time is 200 ns. Note that dwell times must be longer than the averaging time of the measurement!

Creating Tables using Matlab

The following Matlab code generates an execution table and stores it in the EPS1000.

In the example, 1000 times 7 waveplate positions are generated and sent to registers 40 to 46. Because we want to read the applied electrode voltages afterwards, the EPS1000 table execution mode (register 229) has to be disabled. This allows the current waveplate positions to be defined by registers 40 to 46, not by the table memory. The waveplate positions are additionally written to the execution table memory registers 230 to 236, and the step address 0 to 999 (for the 1000 positions in our example) is written to register 219. The current electrode voltages are read from registers 50 to 65 and written into the table memory registers 250 to 265. A short pause before the read operation ensures that the registers are set to the desired waveplate position. The lower 16 bits of the execution time are written into register 237, the upper 16 bits into register 238. Upon a memory write trigger (“1” and “0” to register 221), the table entry is stored at the given address. The actual table length is defined in register 228.

Matlab Code:

```
writeusb(229, 0); % Disable Table Execution Mode
```

```
for ii=1:1000; % number of WP positions,

   % calculate new WP positions
   QWP0Position=round(mod(ii/1024*  11.34, 1)*2^16);
   QWP1Position=round(mod(ii/1024*   2.73, 1)*2^16);
   QWP2Position=round(mod(ii/1024*   5.19, 1)*2^16);
   HWPPosition =round(mod(ii/1024* 37.247, 1)*2^16);
   QWP3Position=round(mod(ii/1024*  13.11, 1)*2^16);
   QWP4Position=round(mod(ii/1024*   7.28, 1)*2^16);
   QWP5Position=round(mod(ii/1024*   4.14, 1)*2^16);

   % set all WPs to desired position
   writeusb(41, QWP0Position);
   writeusb(42, QWP1Position);
   writeusb(43, QWP2Position);
   writeusb(44, QWP3Position);
   writeusb(45, QWP4Position);
   writeusb(46, QWP5Position);
   writeusb(40, HWPPosition);

   if 1 % store current position in EPS table

      writeusb(219, ii-1);
      writeusb(230, QWP0Position);
      writeusb(231, QWP1Position);
      writeusb(232, QWP2Position);
      writeusb(233, HWPPosition);
      writeusb(234, QWP3Position);
      writeusb(235, QWP4Position);
      writeusb(236, QWP5Position);

      pause(.1);

      for iii=0:15 % store current electrode voltages in table
         writeusb(250+iii, readusb(50+iii));
      end;

      % Delay time between positions:
      % 2^17 * 20 ns = 5.24 ms
      Execution_Time=2^17;
      writeusb(237, mod(Execution_Time, 2^16)); % lower 16 Bits
      writeusb(238, bitshift(Execution_Time, -16)); % upper 16 Bits

      % Table write trigger
      writeusb(221, 1);
      writeusb(221, 0);

      % set new table length
      writeusb(228, ii);
   end;

end;
```

Table execution

The table stored in the EPS1000 can be executed by a USB or SPI command (writing “0” to register 227), by an external trigger signal or by the Windows GUI.

Calculating the PDL

After the measurement process, the collected data can be loaded from the EPS1000. For that purpose, each memory address has to be written into register 130, and the value stored at that address can be read from register 131. The EPS1000 supports a burst transfer mode, in which data from subsequent memory addresses can be transferred at once. The burst transfer Matlab program (available from Novoptel) can be used to transfer the whole sample memory between 0 and *maxaddr* (i.e., maxaddr+1 samples) with a single program call:

```
Iref = readburstusb(130, 0, maxaddr, 131);
```

From the measurement data, the PDL and mean and minimum loss in dB are calculated by

```
I = Imeas./Iref;
meanloss_dB = 10*log10(mean(I));
minloss_dB = 10*log10( (mean(Imeas) + sqrt(3)*std(Imeas) ) / mean(Iref) );

th = sqrt(3)*std(I/mean(I));
th = min(th,0.999999999999); % limit, to avoid numerical overflow of atanh and log10
PDL_dB = 10*log10((1+th)/(1-th)); % or alternatively: PDL_dB=log10(exp(1))*20*atanh(th);
```

where *Imeas* and *Iref* are vectors that contain the measurements samples with and without DUT. From all samples in each measurement, the dark current offset has been subtracted previously.

Literature

1. B. Koch, R. Noé, V. Mirvoda, D. Sandel, M. F. Panhwar, "Simple Polarization-Dependent Loss Measurement Based on Polarization Scrambling", Proc. OECC-ACOFT 2014, Paper TU3D-3, 6. –10. July 2014, Melbourne, Australia