Operation of the instrument via USB using Matlab®

The USB driver (*CDM21228_Setup.exe* for instruments with black USB 2.0 socket, *FTD3XXDriver_WHQLCertified_v1.3.0.4_Installer.exe* for instruments with blue USB 3.0 socket) has to be installed on your Windows system and the Novoptel instrument needs to be connected using a USB cable. Examples of Matlab communication scripts are included in Matlab_USB_Support_Files.zip, which can be downloaded from https://www.novoptel.de/Home/Downloads_en.php and https://www.novoptel.eu/Home/Downloads_en.php.

Access the USB driver

Novoptel provides precompiled MEX files for basic read and write functions. Detailed information about the driver are available at https://www.ftdichip.com/FTSupport.htm.

USB Settings

The following settings have to be applied to enable USB 2.0 communication. They will be applied automatically if you use the example communication scripts.

Baud Rate	230400 baud
Word Length	8 Bits
Stop Bits	1 Bit
Parity:	0 Bit

To speed up sequential read and write operations, we recommend to set the ***USB Latency Timer*** to 2 ms.

Example communication scripts

The following four MEX file examples allow basic communication with a Novoptel instrument:

- ft2init.mexw64
- ft2read.mexw64
- ft2write.mexw64
- ft2close.mexw64

For devices with USB 3.0 interface, the MEX files start with “ft3”.

The MEX files are accessed by the basic instrument classes, for example EPS1000.m and PM1000.m

To expand the communication to more than one instrument of the same type, the class file has to be copied and renamed. Additionally, the name of the last device file name (“LastDevEPS.mat”) and the name of the handle (“EPSHandle”) have to be changed inside the class.

Example: Copy and rename EPS1000.m to get the additional file EPS1000_2.m. In EPS1000_2.m, change the last device file name to “LastDevEPS2.mat” and rename the handle to EPS2Handle. Now, both instruments can be kept connected at the same time and read and written to individually.

The usage of the classes is described in the following:

EPS1000.init (calls *ft2init.mexw64*):
This command has to be called before the first write or read attempt. When running the first time, a list of devices is presented, from which the user can select the desired one. After that, the command will always connect to this device if it is detected. To select another device later, call the script with argument “(0)”.

EPS1000.read (calls *ft2read.mexw64*):
Reads a value from a given register.
Example: `res=EPS1000.read(addr)`

EPS1000.write (calls *ft2read.mexw64*):
Writes a given value into a given register.
Example: `ok=EPS1000.write(addr, data)`

EPS1000.close (calls *ft2close.mexw64*):
Once an EPS1000 is allocated by Matlab, it cannot be accessed by other programs, e.g. Novoptel's GUI, until it is deallocated by calling the command *EPS1000.close.*

Burst transfer example

To increase transfer speed, consecutive addresses of an internal memory can be transferred at once in burst mode.

EPS1000.readburst (calls *ft2readburst.mexw64* or *ft3readburst.mexw64*):
Example: `res = EPS1000.readburst(addr, start, stop, read)`
The parameter `addr` defines the address register to be incremented during burst transfer, `start` (`stop`) defines the minimum (maximum) address and `read` defines the register that contains the data to be sent.