AVANTES
MEMBER OF THE NYNOMIC GROUP

FIBER OPTIC MULTIPLEXER DRIVER

ACCESSORIES

Operation and Installation Manual

NEED TECHNICAL SUPPORT?

Scan the **QR-code** or visit **www.avantes.com/support**

We are happy to help you!

ID: 032386

FiberOpticMultiplexer

Interface Package for Windows Applications

Version 2.0

Avantes bv

Oude Apeldoornseweg 28
NL-7333 NS APELDOORN
The Netherlands
Tel: +31-313-670170
Fax: +31-313-670179

Web: www.avantes.com
Email: info@avantes.com

ID: 032386

Microsoft, Visual C++, Visual Basic, Visual C# , Windows and Microsoft Office are registered trademarks of the Microsoft Corporation. Windows Vista, Windows 7, Windows 8 and Windows 10 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Delphi and C++Builder are trademarks of CodeGear, a subsidiary of Embarcadero Technologies.
LabVIEW is a trademark of the National Instruments Corporation.
Qt is a trademark of the Qt Company Ltd in Finland and/or other countries worldwide.

Every effort has been made to make this manual as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. Avantes bv shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this manual.

ID: 032386

Software License

THE INFORMATION AND CODE PROVIDED HEREUNDER (COLLECTIVELY REFERRED TO AS “SOFTWARE”) IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL AVANTES BV OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS OF BUSINESS PROFITS OR SPECIAL DAMAGES, EVEN IF AVANTES BV OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES SO THE FOREGOING LIMITATION MAY NOT APPLY.

This Software gives you the ability to write applications to acquire and process data from Avantes equipment. You have the right to redistribute the libraries contained in the Software, subject to the following conditions:

1. You are developing applications to control Avantes equipment. If you use the code contained herein to develop applications for other purposes, you MUST obtain a separate software license .
2. You distribute only the drivers necessary to support your application.
3. You place all copyright notices and other protective disclaimers and notices contained on the Software on all copies of the Software and your software product.
4. You or your company provides technical support to the users of your application. Avantes bv will not provide software support to these customers.
5. You agree to indemnify, hold harmless, and defend Avantes bv and its suppliers from and against any claims or lawsuits, including attorneys’ fees that arise or result from the use or distribution of your software product and any modifications to the Software.

1. Table of Content

2. Overview

The FOM Interface Package for Windows Applications is a software driver package for Windows 7 / 8 / 8.1 / 10 that allows you to easily write custom software solutions for the FiberOptic Multiplexer.

The package supports all necessary calls to the FOM. The driver uses Windows messages to signal the occurrence of a power failure, thereby allowing the program to prevent damage to your equipment.

3. Installation

Please run the installer that is provided.
The driver does not use the Windows registry or specific files that must be copied to the Windows directory tree. Just copy all necessary files to your working directory. In principle, all you need are the files MUXUSB.DLL and PHIDGET22.DLL. Both 32 and 64-bit versions of these DLLs are provided in the “Binaries” subdirectory.
Delphi users also need the interface file MUXUSB.PAS, C++ users need MUXUSB.H and MUXUSB.LIB etc. A custom version of the link library is supplied for Borland C++ Builder.

4. Version History

New in version 2.0

- Different stepper controller, with new DLL version PHIDGET22.DLL.
- The MPX_Status call now also returns the FOM-type.
- The MPX_Goto, MPX_GotoStep and MPX_GotoReferenceCustom calls are simplified.

New in version 1.3.1

- Updated VC version used to VC2017, updated Qt version to 5.12.3
- Added 64-bit version and sample application.

- Added MPX_GotoReferenceCustom function.

New in version 1.3

- Major change to different stepper controller with USB interface. New controller supports 3200 steps, where the older Elstep controller supported 400 steps.
- PHIDGET21.DLL supplied instead of serial library SUPERCOM.DLL. Samples updated with .net languages, VC++ sample changed from MFC to Qt4 library.
- When updating your program, make sure you use a value of 1 for port in Myxinid and update the values for F en L in MPX_Goto and MPX_GotoStep to the new default values of 5500/45.

New in version 1.2

- Added MPX_GotoStep function to allow positioning at other than predefined positions

5. Basics

You must have the following other items before you can develop your own software for the FiberOptic Multiplexer.

- The multiplexer
- A programming environment that supports calling Dynamic Link Libraries (DLLs). This includes most popular programming environments (e.g. Microsoft Visual Basic .net, Visual C#, Visual C++, Embarcadero Delphi, Embarcadero C++ Builder).
  National Instruments LabVIEW does not directly support responding to Windows messages. Therefore, the power detection feature cannot be used with this programming environment. The multiplexer can, however, be operated normally.

If you are using a 64-bit version of Windows, please note that the default Solution Platform of Visual Studio is “Any CPU” (at least for .net languages), which could yield an error, as your program will be compiled in 64-bit, and will then not be able to load the 32-bit DLL. Please select the “x86” Solution Platform to force the compiler to output a 32-bit compatible program. Select the “x64” Solution Platform to force the compiler to output a 64-bit compatible program.
The only 64-bit program binaries that are provided are for C++/Qt5. All other samples are providing with 32-bit program binaries only. If you recompile these for 64 bits, make sure you copy the 64-bit versions of muxusb.dll and phidget22.dll into the correct output directory.

The driver was developed in Microsoft Visual C++ 2017.

Sample programs in Borland Delphi 2009, Borland C++ Builder 2009, Microsoft Visual Basic 2017, Microsoft Visual C# 2017, Microsoft Visual C++ 2017/Qt5 and National Instruments LabVIEW 2009 are provided. These sample programs are not commercial grade programs. They are not hardened against user error and provide very limited error checking. The purpose of these programs is to illustrate how to call the various driver functions.

As there is no continuous feedback on the current position of the multiplexer, the computer does not know where the multiplexer is positioned in case the power or the data connection is (temporarily) lost. To prevent problems, the power can be monitored. If there is a problem, a windows message will be sent to the host application. Your program can then take appropriate action, such as moving the multiplexer to the reference position or stopping altogether.

ID: 032386

6. Data Structures

Structures used:

The multiplexer’s status is transmitted as a structure:

C style:

```
typedef struct
{
    unsigned char MR;
    unsigned char PR;
    unsigned char PWR;
} StatusType;
```

Pascal style:

```
PStatus=^Status;
Status=record
  MR:byte;
  PR:byte;
  PWR:byte;
end;
```

Abbreviations used:

MR: Motor Running
PR: Position Reached
PWR: Power

7. Description of Functions

```
function MPX_Init(h:THANDLE;
                  port:DWord;
                  PowerDetection:byte):integer;stdcall;

Initialises the multiplexer control.

Input  : h               -> Windows Handle of host application
         port            -> use 1 for a USB connection
      PowerDetection ->  If true=1, power failure is signaled to the host
application.
Return : 0               -> success
         -1              -> failure
```

```
function MPX_Close:integer;stdcall;

Closes the connection to the multiplexer.

Return : 0               -> success
         -1              -> failure
```

```
function MPX_Goto(position:byte;F,L,U:word):integer;stdcall;

Positions the multiplexer at the desired position.

Input : position         -> Multiplexer position (1-16)
        F                -> unused, for compatibility only
        L                -> unused, for compatibility only
        U                -> unused, for compatibility only

Return : 0               -> success
         -1              -> failure
```

function MPX_GotoStep(position,F,L,U:word):integer;stdcall;

```
Positions the multiplexer at the desired steppermotor position.

Input : position        -> Steppermotor position (1-3199)
        F               -> unused, for compatibility only
        L               -> unused, for compatibility only
        U               -> unused, for compatibility only
```

Comparing both goto functions, the steppermotor positions are calculated from the multiplexer positions with the following formula:

Y = (X - 1) * 200 + 50

where Y is the steppermotor position, and X is the multiplexer position.

```
Return : 0              -> success
         -1             -> failure
```

function MPX_Stop:integer;stdcall;

```
Immediate software stop.

Return : always returns 0.
```

function MPX_GotoReference:integer;stdcall;

```
Positions the multiplexer at the reference position.

Return : always returns 0.
```

function MPX_GotoReferenceCustom(VelLim,Acc:word):integer;stdcall;

```
Maintained for compatibility only
Positions the multiplexer at the reference position.

Return : always returns 0.
```

function MPX_Status(stat:PStatus):integer;stdcall;

```
Queries the three hardware signals that are monitored: Motor Running,
Position Reached and Power.

Input/Output : stat     -> Pointer to the status record
Return : FOM-type
         1 = older model FOM with 1063 board
         2 = 4x4 FOM
         3 = 2x8 FOM
         5 = 1x16 FOM
```

```
function MPX_MotorStatus:integer;stdcall;

Queries the status of the stepper motor.

Returns:
-1 Limit switch approached in positive direction.
-2 Limit switch approached in negative direction.
-3 Low level at stop input.
-4 Position greater than setpoint range (software limit switch).
-5 Position less than setpoint range (software limit switch).
-6 Excess temperature.
-10 Range of the parameter exceeded.
-11 Start command while the motor is running.
-12 Command unknown.

>= 0 is the position of the motor in steps.
```

```
function MPX_EnableLogging(enabled:boolean):integer;stdcall;

Will write messages to a "muxusb.log" file when enabled.

Input : enabled       -> use true=1 to enable, false=0 to disable
Return : always returns 0.
```

8. muxusb.ini file

The muxusb DLL can write messages to a logfile when the MPX_EnableLogging function is used. As an alternative, a ini file could be used.
A file called “muxusb.ini” can be placed in the same folder where the DLL itself is located. The ini file should look like this:

```
[logging]
force=1
```