Leader in Photon Counting

Manual For The LabVIEW™ GVD .dll Wrappers

Written by

Frank Quadt,M.sc.

February 11, 2021

Contents

Preface

Thank you for working with Becker&Hickl's GVD and DCS modules.
You are already familiar with B&H's own control software Spcm.exe. There you have the full control over all different module types and their operation modes.
Nevertheless, you feel that there is an important feature missing to bring your application to its expected heights.
B&H honours that and is offering you software drivers capable to control your GVD and DCS module down to the lowest level.
These drivers are written in National Instruments LabWindows / CVI and available on your computer at C:\Program Files (x86)\BH\GVD\DLL\gvdXX.dll (XX standing for your operating system's bitness).
For programmer wanting to use National Instruments LabVIEW to control their GVD and DCS modules, B&H provides LV wrappers in the Becker&Hickl GVD Modules folder to call the above mentioned dll.
Also included are some higher-level convenience VIs as well as some simple examples to provide you with a starting point to your own LV-project.

This manual is about the use of these wrappers and examples, aimed at users with a solid understanding of programming technics in LabVIEW.
For training in LabVIEW itself we recommend to contact National Instruments directly at www.ni.com.

Attention: Notice that the LabVIEW Plug and Play Drivers are just usable with the corresponding DLL. Otherwise these Drivers will fail to load! Further, a valid license key is necessary to use these DLLs.

1 Installation of the LabVIEW™ GVD .dll Wrappers

The Becker&Hickl GVD Modules LabVIEW project comes as collection of files in a folders hierarchy installed together with your Spcm.exe software at C:\Program Files (x86)\BH\GVD\BH-LabVIEW-GVD.
The VIs and the project are written in the LabVIEW2017 development environment. Upgrading to a newer development environment on your PC should not be a problem. For downgrading to older versions use the File>Save for Previous Version… option in the LV project window or get in contact with Becker&Hickl.

All LV-wrapper VIs in the \GVD folder depend on the gvdXX.dll located at C:\Program Files (x86)\BH\GVD\DLL on your computer. There are 3 versions available. You will need to use gvd32.dll on Windows32 bit systems. For Windows64 bit systems you will use the gvd32x64.dll when programming in the LabVIEW 32bit environment or gvd64.dll when running LabVIEW 64bit.

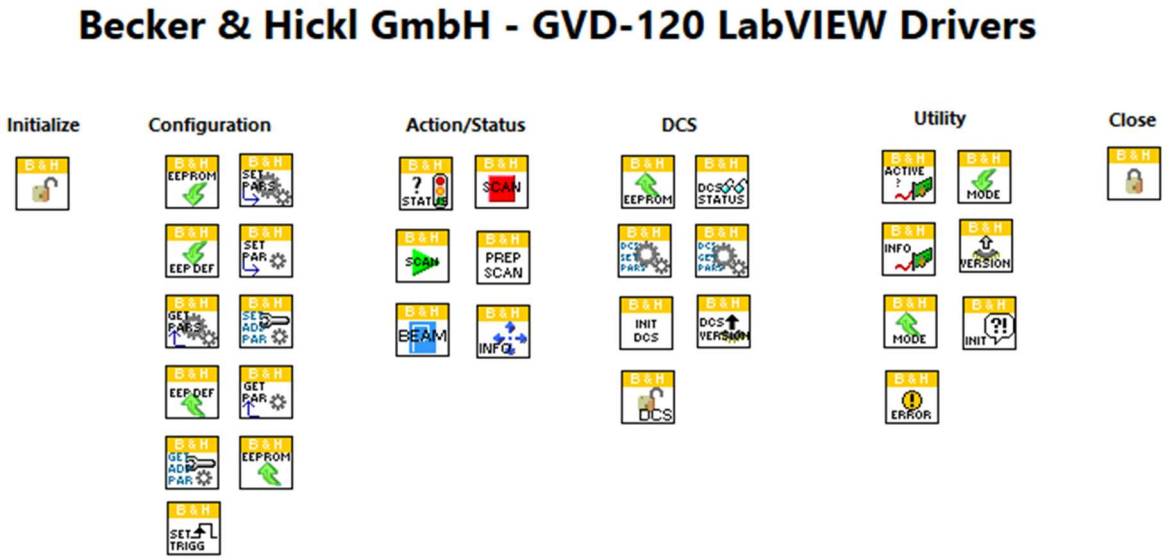
We recommend to always upgrade your Spcm.exe to the newest version which will include new features and bug fixes.
With every new release there will be new versions of the gvdXX.dll. Since B&H reserves the right to change (with release notes) functions and their parameter sets at any time, we also recommend not to edit the wrapper VIs directly but keep them in their original form at their original location. Like this an upgrade of Spcm can overwrite the wrapper VIs without changing your code.

To start your own LabVIEW project, you should create a folder on your User\My Documents path, create a New Project on the LabVIEW start-up window. In the project window right click on My computer and choose Add>Folder (Auto-populating)… In the file select box navigate to C:\Program Files (x86)\BH\GVD\BH-LabVIEW-GVD\ and click Select Folder.
You could do the same for the Examples folder, but we recommend to create a copy of this folder on your development path. You most likely will want to do changes on these VIs which you might not be allowed to on the protected path of C:\Program File (x86). They also might be overwritten with newer Spcm.exe version distribution.

When attempting to write a GVD control application we suggest to start off with one of our examples located in the \Examples folder.

2 The GVD Module Functions

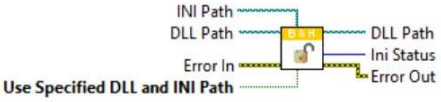
The GVD module functions provide a full range of procedures for controlling the module(s) totally. Besides the possibility to initialize and close the module properly different functions for controlling the three different outputs are available.
In Fig.15 the main menu can be observed.



Attention: Take care that you fully understand the specific functions and their influence on the GVD module. Other behaviour can damage the GVD module and also other connected devices.

2.1 The Initialization Function

This function establishes the communication with the cards mounted in the computer system or with an external device which includes GVD cards. The DLL can handle up to four GVD modules in total.



Further, it prepares the cards for further driver operations. Therefore, call this VI before calling other driver VIs for this device. Generally, you need to call the **Initialize** VI only once at the beginning of an application. Moreover, this function checks the Becker & Hickl default installation path for the right DLL and the necessary INI file. If the user wants to use a DLL and a INI file located in another directory than the standard installation path please wire a true constant to “Use specified DLL and INI path” input parameter. Furthermore, if the user wants to use more than one GVD module please notice that the INI file has to be updated. For that, open this INI file, normally located in “C:\Program Files (x86)\BH\GVD\DLL” with an appropriate text editor like notepad++ or notepad and change inside the entry “active =0” to “active=1” in the downer part of the file. Notice, that the INI file name are not allowed to change!

Inside this function other functions like “GVD init” and “GVD GetDLLPath.vi” will be called.

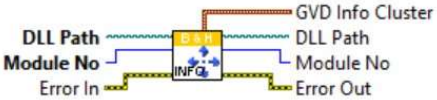
2.2 The Close Function

The close function closes all open references and closes the DLL reference in a proper way and brings the internal DLL structure in a state before the initialising function was called.

2.3 Action–Status Functions

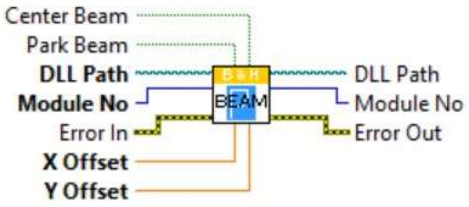
2.3.1 The Get Scan Info Function

The "Get Scan Info" function contains the most important information about the current scan curve prepared for the GVD module specified by "module no". The obtained cluster contains parameters like whether the scan was valid, the current values for the line time, tick time, pixel time as well as the flyback times in both directions, x and y respectively. If the value of "Scan Valid=0" the function "Prepare Scan Curve" must be called to prepare and load the scan data correctly for first use.



2.3.2 The Park Beam Function

The procedure is used to park beam in the specified position of the current scanned area for the GVD module specified by the user.
Further, an offset can be defined which has the origin at the upper left corner of the specified scan area.
Notice, that the specified offset will be neglected if the "Center Beam" parameter was set to true.



2.3.3 The Prepare Scan Curve Function

This function must be called before scanning. Here the scan curve will be calculated and stored in GVD internal memory. After setting most of the GVD parameters the scan curve need to be recalculated. Therefore, this function should be called right before the next scanning.

2.3.4 The Start Scan Function

The "Start Scan Function" starts the scanning procedure of the GVD, that means the GVD starts to send the specified sequence of the scan control signals. The sequence runs continuously or with the number of frames defined by "frame counter". In the case that the scan curve is not valid, the function prepares a new scan curve.

Further, if a trigger condition was set the function will wait for the trigger pulse before the start of the scan sequence (feature available on modules with FPGA version B1 or newer).

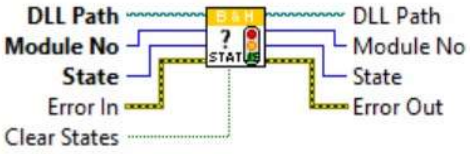
2.3.5 The Stop Scan Function

The "Stop Scan Function" is used to stop the running scan sequence by a software command on the GVD module specified by "Module No".

Further, this function must be called after the finished scan sequence to set the GVD to its initial state.

2.3.6 The Test State Function

The "Test State Function" is used to check in which status the GVD module currently is. For that, the delivered status bit information will be evaluated and displayed for the user.



The different status can be:

Scan Running If this is active in the provided cluster the scan is currently running

Scan Finish If this is active in the provided cluster the started scanning procedure with the previous defined scan curve is finish.

Beam Parked After using the park beam function with park beam enabled this LED flashes if the park beam position was reached.

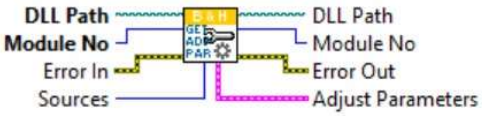
Wait For Trigger After set the trigger condition and after calling the start scan function this LED indicates that the module waits for a trigger

Scan Signals If this is active the GVD specified by "Module No" sends scan signals. This status is obligatory for using the FIFO image mode.

2.4 Configure Functions

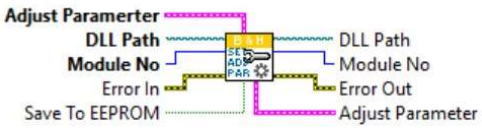
2.4.1 The Get/Set Adjust Parameters Function

The "Adjust Parameter Functions" are low level functions and are used to set values which have an impact of the determination of the scan curve.

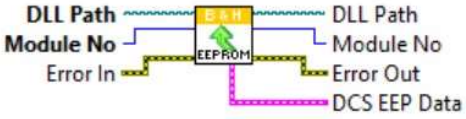


Normally, the adjust parameters need not be read explicitly because the EEPROM is read during the initialising process and the values are loaded directly from the EEPROM.

Notice, that it is not recommended to change these values. Becker and Hickl GmbH strongly discourage to use modified adjust parameters without consulting Becker and Hickl GmbH due to the fact that they have a big influence on the generated scan curve and the module function can be seriously corrupted.

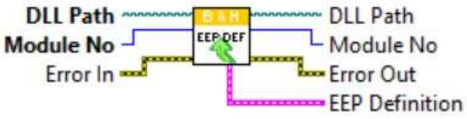


2.4.2 The Get EEPROM Data Function



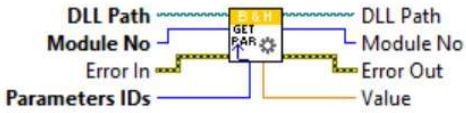
The "Get EEPROM Data" function reads the currently stored data from the EEPROM. The EEPROM contains information about the adjustment parameters which can be read using "Get Adjust Parameter Function" but also the correct module type, the serial number as well as the production date.

2.4.3 The Get/Set EEPROM Defines Function

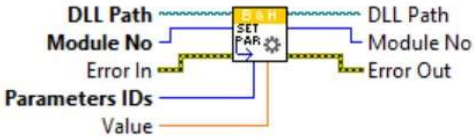


The "EEPROM Defines Function" contains information about the last used LASER names. These names can be overwritten with the "GVD Set EEPROM Defines" function. Up to two LASER names can be stored.

2.4.4 The Get/Set Single Parameter Function

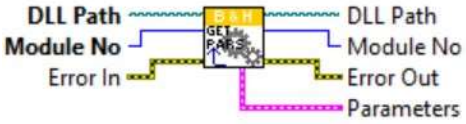


The functions "Get Single Parameter" and "Set Single Parameter" load/ overwrite just the current value of the requested parameter from the DLL internal data structures of the GVD module specified by "Module No". Further, the "Set Single Parameter" function sets the specified hardware parameter and the new parameter value is recalculated according to the parameter limits and hardware restrictions.



Furthermore, cross dependencies between different parameters are taken into account to ensure the correct hardware operation.
Moreover, it is recommended to read back the parameters after setting to get their real values after recalculation.
Notice, that **just the specified parameter** will be read/ changed. Other parameters are not affected.

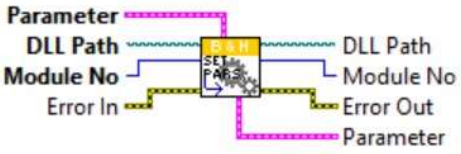
2.4.5 The Get/Set Multi Parameters Function



The "Get Multi Parameters" and "Set Multi Parameters" function loads/overwrites all of the current values of all parameters from the DLL internal data structures of the GVD module specified by "Module No".
Further, the "Set Multi Parameters" function sets the specified hardware parameters and the new values will be recalculated according to the parameter limits and

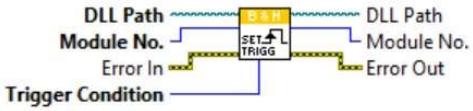
hardware restrictions. Furthermore, cross dependencies between different parameters are taken into account to ensure the correct hardware operation.

Moreover, it is recommended to read back the parameters after setting to get their real values after recalculation.



Notice, that **all specified parameters** will be read/ changed at the same time.

2.4.6 The Set Trigger Function

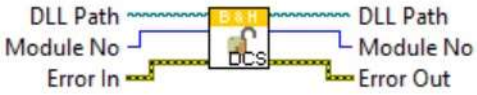


This function provides an easy way to set the trigger condition in a correct way. Possible values are “None”, that means no trigger will be used and the software will not be waiting for the trigger. Further, “Rising Edge” as well as “Falling Edge” are chooseable. Here the software will wait for the corresponding trigger before the measurement will be started.

2.5 DCS–Box Specific Functions

A DCS–Box can be connected to a GVD module to simplify the cabling of the scanning system. It contains its own FPGA and EEPROM.

2.5.1 The Init Function



The “Init Function” must be the first function called if using a DCS– Box and after calling the “Initialize.vi”.

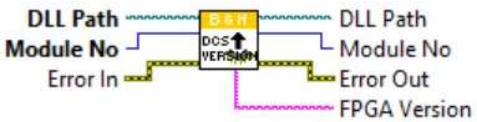
The “DCS Init Function” tests the communication between GVD module specified by “Module No.” and the DCS–Box connected to it.

If a DCS–Box is connected the function:

– performs an EEPROM checksum test of the DCS–Box

– writes control parameters to the DCS–Box FPGA (for that the INI file will be used)

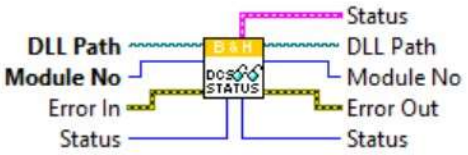
2.5.2 Get Version Function



This procedure loads the version variable with the currently used FPGA version of the DCS–Box connected to the GVD module specified by “Module No.”. This is a low level procedure and not intend to use in normally.

2.5.3 Get Init Status

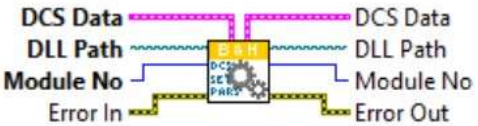
The “Get Init Status” function loads the ini status variable with the initialisation result code set by the function “DCS init” for the DCS–Box connected to the GVD module specified by the “Module No.”.

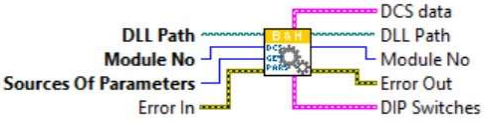


Possible values are:
±0 Initialisation OK
−1 Wrong DCS EEPROM checksum
−2 Initialisation not done
−3 Initialisation not done, wrong FPGA version of GVD
−4 Cannot access DCS resources – data line hangs on 0
−5 Cannot access DCS resources – data line hangs on 1
−6 Error when rd/wr DCS EEPROM
−7 Error when rd/wr DCS FPGA

2.5.4 Get/ Set Parameters Function

After calling the “DCS Init” function the control parameters from the initialization file are sent to the DCS–Box and in the internal data structure of the DLL. The “Get Parameters” function provides the possibility to get access to these control parameters. This function transfers the parameter value of the DCS–Box which is connected to the GVD module. Depending on the parameter “Source Of Parameters” the parameters are taken from the DCS–Box, FPGA, DIP switches or EEPROM. The resulting value is bitwise encrypted. The encryption is as follows:





Ctrl Low	bits 0 – 2	SPC–A total no. of routing bits
	bit 3	SPC–A laser routing enabled
	bits 4 – 6	SPC–B total no. of routing bits
	bit 7	SPC–B laser routing enabled
Ctrl High	bit 0	Red LED state if bit 1 = 1, 0 = off, 1 = on
	bit 1	Red LED of Scanner shows filter switch state
	bits 2	SPC–A MARK 3 enable
	bits 3	SPC–B MARK 3 enable
	bits 4	SPC–A DCS–Switch signals OVLD to DCC
	bits 5	SPC–B DCS–Switch signals OVLD to DCC

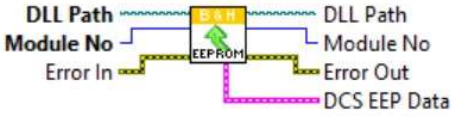
	bits 7	always 1 when DCS is controlled by software

2.5.5 Read Status Function

The "Read Status Function" is a low level VI and returns the current status of the DCS–Box connected to the GVD module specified by "Module No.".
The status bits delivered by the function are listed below:

State	Status No.
Switch 0 state	0x1
Switch 1 state	0x2
A SPC module is not SPC–830	0x3
B SPC module is not SPC–830	0x8
Scanning in Progress	0x10

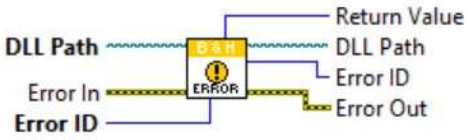
2.5.6 Get EEPROM Data



The structure "EEP Data" is filled with the contents of the EEPROM of DCS–Box connected to the GVD module specified by "Module No.".
The EEPROM contains information about the production date and the adjust parameters. Further, the serial number as well as the correct module type is stored inside the EEPROM.

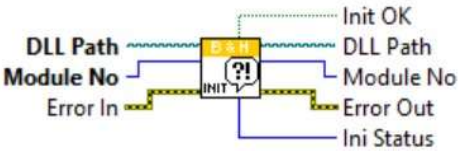
2.6 Utility Functions

2.6.1 The Get Error String Function



The procedure returns an error string which contains the explanation of the GVD DLL error. Additionally, it contains the ID equal to the Error ID. Possible error values are listed in the "gvd def.h" file.

2.6.2 The Get Init Status Function

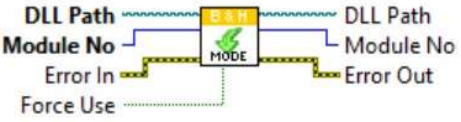


The procedure loads the ini status variable with the initialisation result code set by the function "GVD init" for the specified module by "Module No.".
Possible values are listed below:

±0 Initialisation OK
−1 Active = 0, Initializing Not Done
−2 Wrong EEPROM Checksum
−4 Hardware Test Failed
−5 Cannot Open PCI Card

−6 Module Already in Use
−7 Corrupted License Key
−8 License Key Not Read From Registry
−9 License Is Not Valid For GVD DLL
−10 License Date Expired

2.6.3 The Get/Set Mode Function



The “Get Mode Function” returns the current mode of the DLL operation. Two modes are possible: hardware or simulation mode. The “Set Mode Function” is used to change the mode of the DLL operation between the hardware mode and the simulation mode. It is a low level VI and not intend to normal use. It can be used to switch the DLL to the simulation mode if hardware error occurs during the initialization.

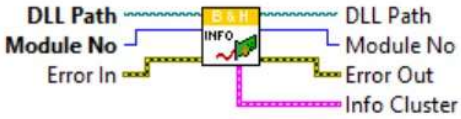
2.6.4 The Get Version Function

This function returns the currently used FPGA version of the GVD module specified by “Module No.”. This is a low-level procedure, not needed normally.

2.6.5 The Test If Active Function

This function returns information whether the GVD specified by “Module No.” is active or not. As a result of a wrong initialisation a module can be deactivated. For further information for the reason of deactivating the module, please run the “GVD Get Init Status” function.

2.6.6 The Get Module Information Function



After calling the “GVD Init Function” the internal structures for all active modules will be filled with the module information. These structures contain information about i.e. the current used module type, the bus number of the card and the slot number.

2.7 Example VIs

Becker and Hickl GmbH provides additionally to the LabVIEW™ gvd.dll wrappers some example VIs. These VIs show how to use the corresponding single functions to build up your own application. One of the first examples which should be started is the ”JustOpenClose.vi”. This very simple example calls the GVD module in your system. After that it will display several information like module type, bus number and serial

number. Further, this is a quick check for the GVD LabVIEW™ license. In Fig.16 the front panel of the ”JustOpenClose.vi” is shown.

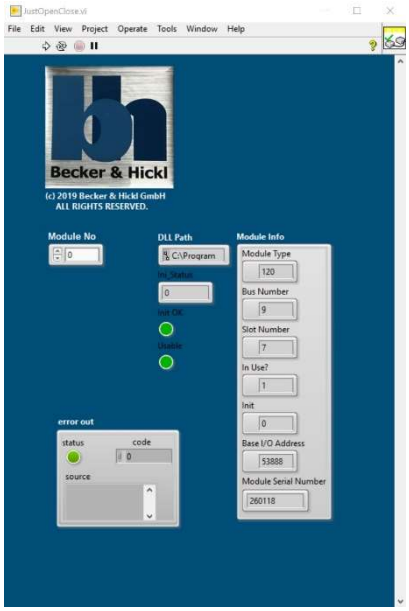


Figure 16: Front panel of the “JustOpenClose” application. Here one can check the connection of the used GVD

Moreover, the example folder contains also an example to have access to a single GVD module which is connected to a DCS–Box. This application gives the user the possibility to obtain all information from the GVD module like module type, bus number, serial number but also the current status of the DIP switches of the used DCS–Box. Please notice, that this application is just to obtain information from the GVD as well as DCS module. In Fig.17 the front panel of this application is shown.

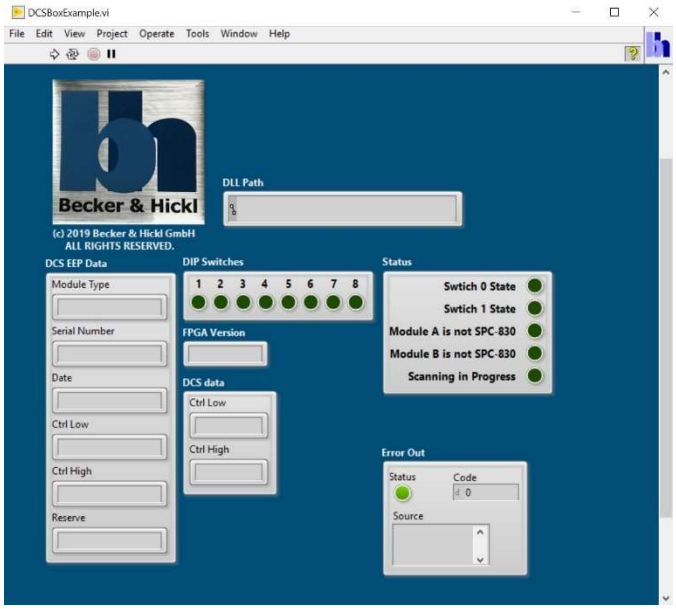


Figure 17: Front panel of the "DCSBoxExample" application. Notice, that this example just shows how to read specific information from the DCS-Box.

The most elaborate example in this GVD-LV-project is the SimpleControl.vi. It shows how a complete control GUI for a GVD-module can be implemented.

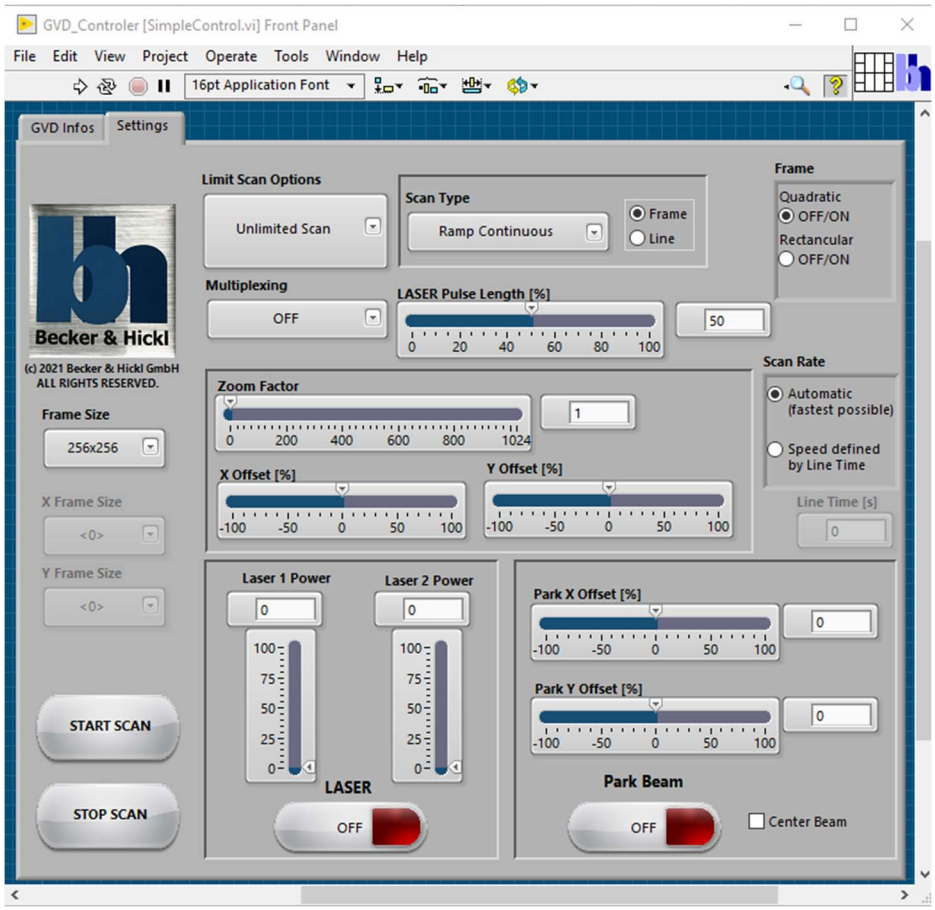


Figure 18: Front panel of the "SimpleControl" application.

Known Issues

In the following section frequently ask questions as well as known issues will be listed. If you here do not find a solution to your specific problem please feel free to get in touch with Becker & Hickl GmbH.

LabVIEW do not find the previous copied drivers If LabVIEW asks where to find the different VIs for your specific measurement card please make sure that you had started the correct LabVIEW version. Maybe you started a 32–bit version instead of the 64–bit version.

LabVIEW crashes during the use of the LabVIEW Plug & Play Drivers
This issues can happen, if the corresponding used DLL has a wrong version. There exists a minimum DLL version for each LabVIEW Plug and Play driver. In the following the minimum DLL version number for each Becker & Hickl LabVIEW Plug and Play driver will be provided.

Package	Version
DCC	2.5
DDG	3.0
GVD	1.9
PMS	3.4
SPC	4.6

Contact Information

Becker & Hickl GmbH provide a huge amount of possibilities to establish your experimental setup. You need help, support or you want to discuss some specific topics or issues so please get in touch with us.

Becker & Hickl GmbH
Nunsdorfer Ring 7 – 9
12277 Berlin Germany

Phone: +49(30)2128002 – 0
Fax: +49(30)2128002 – 13
Email: info@becker–hickl.com

If you have some specific questions, annotations or criticism referring the programming of the LabVIEW™ GVD .dll wrappers, please get directly in touch with the programmer:

Software Graphic & Design:	Dr.C. Junghans and F. Quadt
Text/Layout:	Dr.C. Junghans and F. Quadt
Developer:	F. Quadt