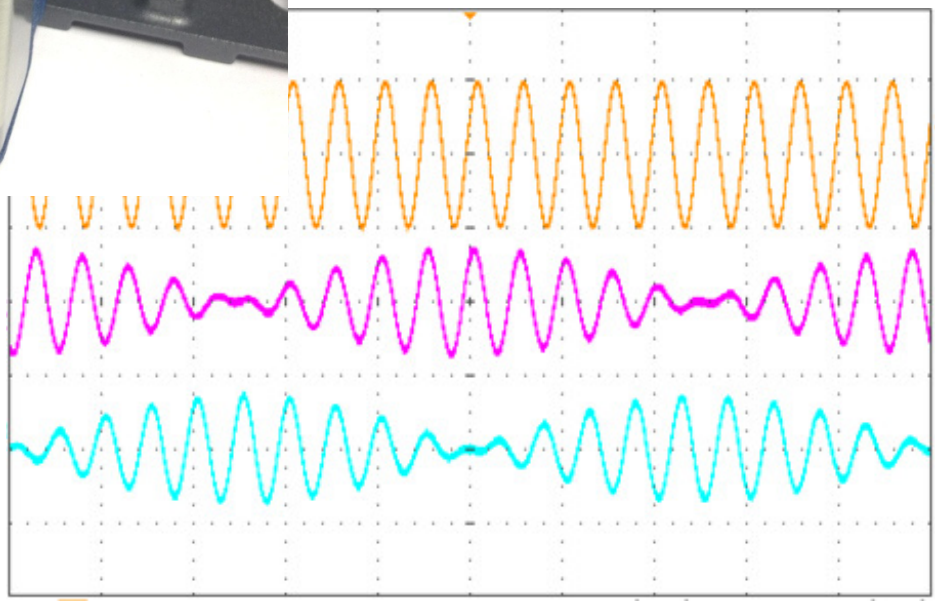
IRS cRIO-Resolver-Simulation

Manual



Content

General

1.1 Revision history

Date	Version	Changes
21.11.2012	1.0	Initial release
03.12.2012	1.1	- added: screenshots - added: block diagram - changed: software interface
07.12.2012	1.2	Adaption to a more generic documentation
29.01.2013	1.3	Adaption of software part to control by DLL, FPGA and LabVIEW
28.03.2013	2.0	Added extension for HW version 2.0 including programmable amplitude.
09.04.2015	2.1	Updated Software for LV2013, added Executable GUI
27.05.2015	2.2	Added Speed ramp functionality

Table 1: Revision history

1.2 Abbreviations

Abbreviation	Description
cRIO	Compact reconfigurable input output module
FIFO	First in first out
FPGA	Field programmable gate array
LUT	Look-up table
MAX	Measurement & Automation Explorer
PGA	Programmable gain amplifier
RPM	Revolutions per minute
RT	Real time
SPI	Serial peripheral interface
VI	Virtual instrument

Table 2: Abbreviations

1.3 Purpose

This document describes the hardware functionality and the usage of the software interface for IRS resolver simulation module.

1.4 Annexes

Nr.	Document	Date	Remark
1	cRIO_Resolver_Simulator_V1.2.pdf	07.01.2013	Schematic

Table 3: Annexes

1.5 List of tables

1.6 List of figures

2. Introduction

IRS cRIO resolver simulator can be used in a National Instruments Compact-RIO chassis to simulate the signal of the resolver sensor of electric motors.

2.1 Resolver sensor

Resolver sensors are often used in the power-train of electrical or hybrid vehicles. Following figure shows the principle of a typical resolver, where the Excitation winding "R" is rotating. The two stator windings receive the energy from the excitation and generate a signal with an amplitude, depending on the rotor position.

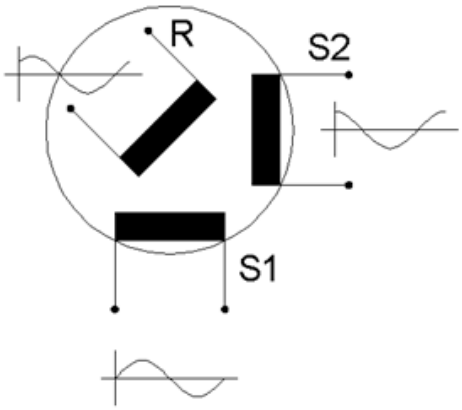


Figure 1: Resolver principle

When the Excitation winding is rotating, the two stator windings will show a signal similar to the following figure on the right.

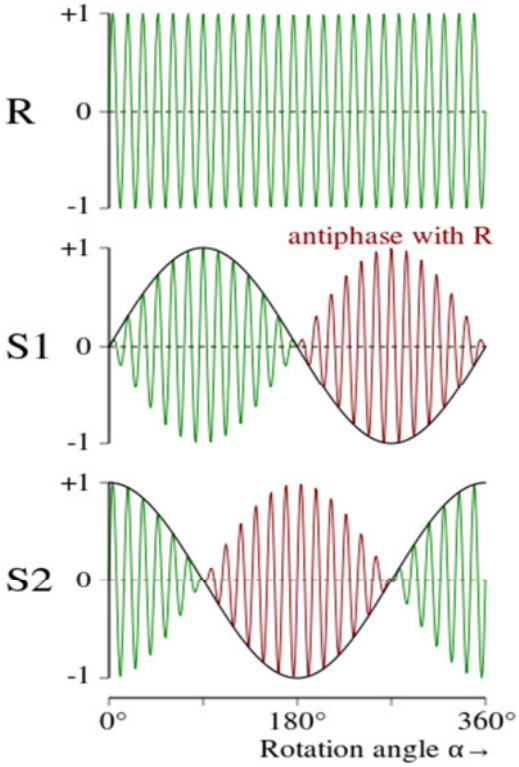


Figure 2: Resolver signal

2.2 Resolver simulation

For component testing of the power stages of the motor driver, it is often not desired to include an original motor with its sensors in the tester. On the other hand the resolver signal should be generated for both functional test, End-Of-Line test or for lifetime tests during design validation.

Especially for lifetime testing IRS provides test systems which simulate the original motor by means of passive inductive loads. With this setup high phase currents of 400A_{rms} can be generated, while the power loss is very low, since the energy is stored as reactive power in the load coils.

The inverter, which is tested and generates the phase currents through the load coils, must “see” the same conditions as it would see with a real electric motor in the car. Thus, the sensor signal of the electric machine must be emulated. At this point the Resolver simulation comes into play.

The resolver simulation can simulate the sensor signal of a rotating machine or may emulate specific positions of the electric motor. The module can be controlled by software to stop at certain positions or perform continuous modulation, like a rotating electric motor does. Following figure shows a typical waveform of the resolver simulator:

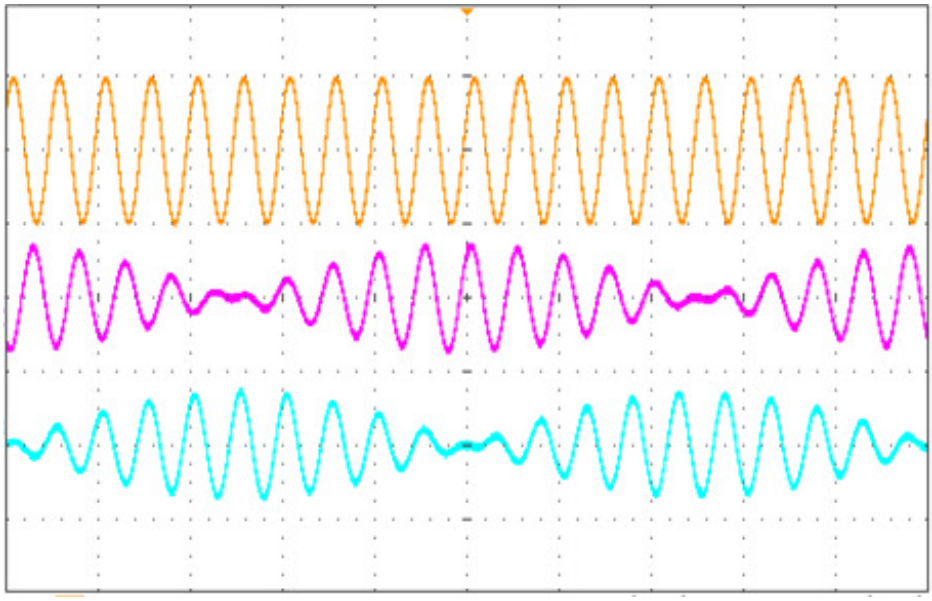


Figure 3: typical simulation waveform

3. Hardware

The following chapter highlights the hardware of the module.

3.1 Block diagram

The following figure shows the block diagram of the module. The signals on the right are accessible to the user. The signals on the left represent the Compact-RIO interface and are not described in detail in this manual.

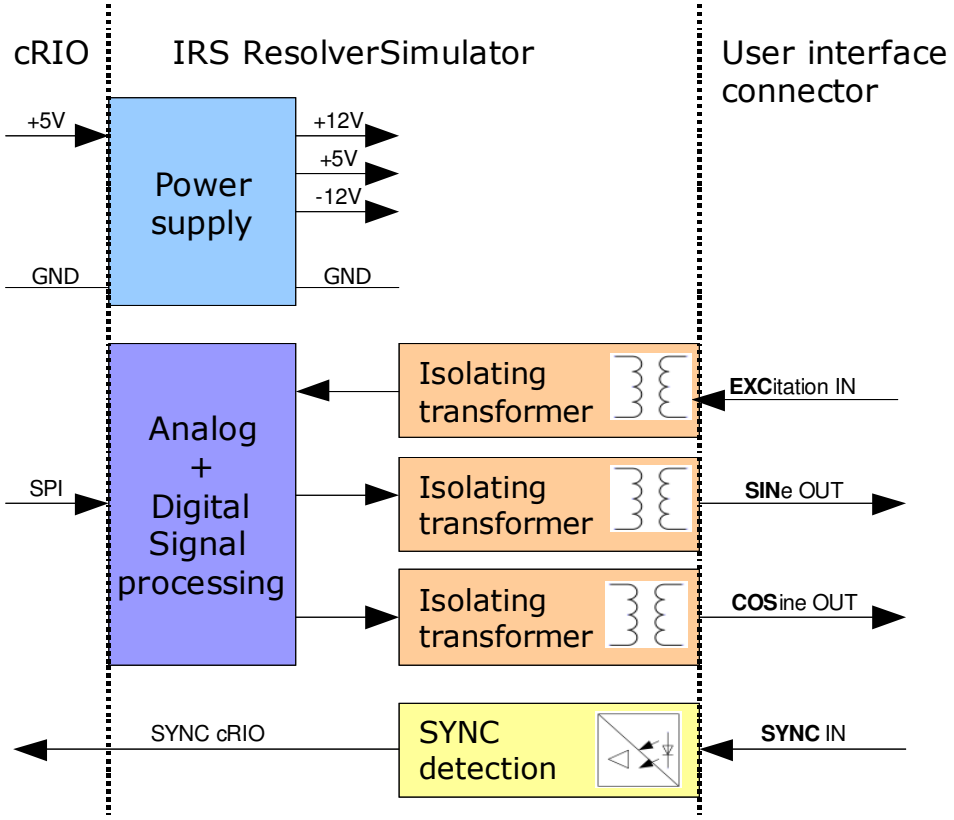


Figure 4: Block diagram

All input and output signals are isolated from the Compact-RIO. The Excitation reference input, sine and cosine outputs are isolated by means of transformers. The SYNC input is digitally isolated by means of optocoupler.

3.2 Interfaces

The user interface Signals have the following functionality. Every signal has a positive and negative terminal.

- **Excitation reference In**:
  - Input signal from the power converter to be tested
  - Typical sine wave signal of about $10V_{pp}$ is applied
- **Sine / Cosine Out**:
  - Output signals from the simulation, which is an Amplitude-modulated representation of the excitation signal. Modulation is performed by software.
- **SYNC input**:
  - This signal can be used to synchronize the modulation frequency of sine and cosine outputs to a speed signal.
  - I.e. RPM of the motor can be applied here.

3.3 Technical data

Signal	Item	Min	Typical	Max	Unit
Supply voltage	Supply voltage (provided by cRIO chassis, no external voltage required)	4,5		5,5	V
	Power consumption (provided by cRIO chassis)		50		mA
Excitation reference input	Excitation reference voltage range			20	V_{pp}
	Excitation reference input resistance (@10 kHz, inductive)		+j 2000		Ω
	Excitation reference input frequency	2	10	20	kHz
SYNC input	SYNC input voltage range (peak voltage)	5		50	V_{peak}
	SYNC input detection threshold		3		V
	SYNC input frequency	1		200	Hz
	SYNC input current (U > 3V)	2		7	mA
Sine / Cosine output	Sine/ Cosine Output level (depends on software settings)	0		20	V_{pp}
	Output resistance (@2,5 … 10 kHz)	5	7	20	Ω

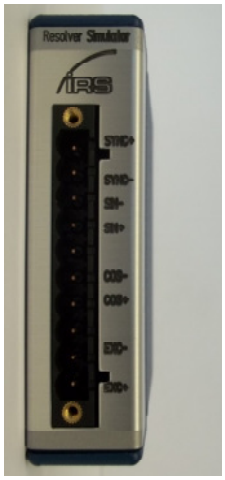
Table 4: Technical data

3.4 Pin out

The following table shows the pin out of the front connector.

Pin	Description
0	SYNC +
1	SYNC -
2	Sine OUT -
3	Sine OUT +
4	-
5	Cosine OUT -
6	Cosine OUT +
7	-
8	Excitation reference IN -
9	Excitation reference IN +

Table 5: Pin out front connector



An input sine wave will be will be modulated depending on the requested sense of rotation and motor speed. The modulated signals will be output as a damped sine wave and a damped (and phase shifted) cosine wave. The attenuation can be configured by software.

It is also possible to set a static rotor angle.

Furthermore a synchronization input is available which can be used to measure an external frequency for example.

3.5 Example Waveforms

The next screenshots illustrate the functionality (channel 1: excitation, channel 2: sine output, channel 3: cosine output).

Figure 5: Input and output signals at 10000 rpm

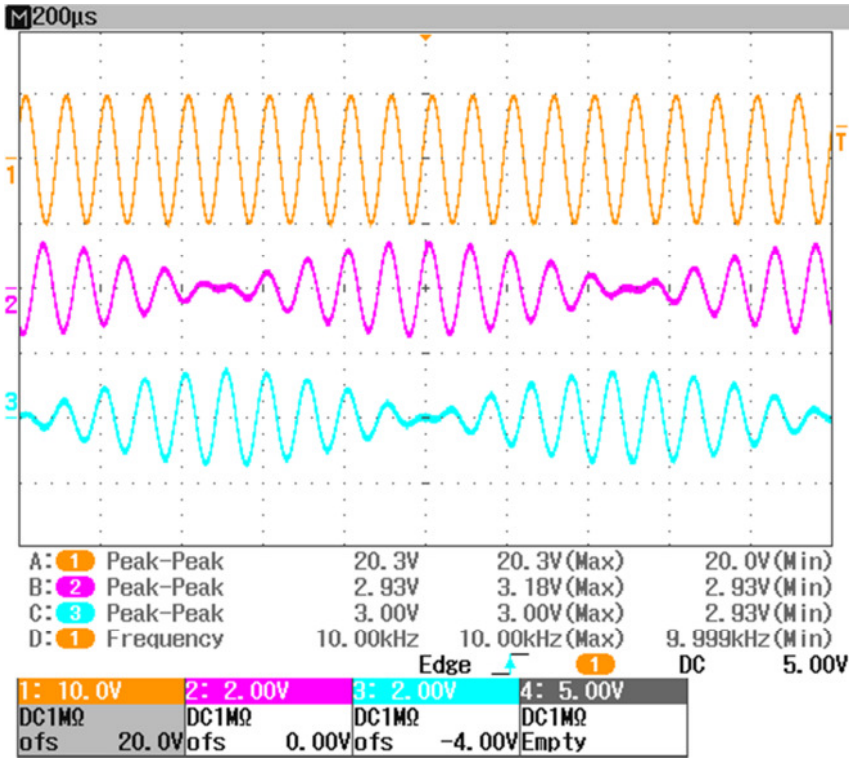


Figure 6: Input and output signals at 4000 rpm

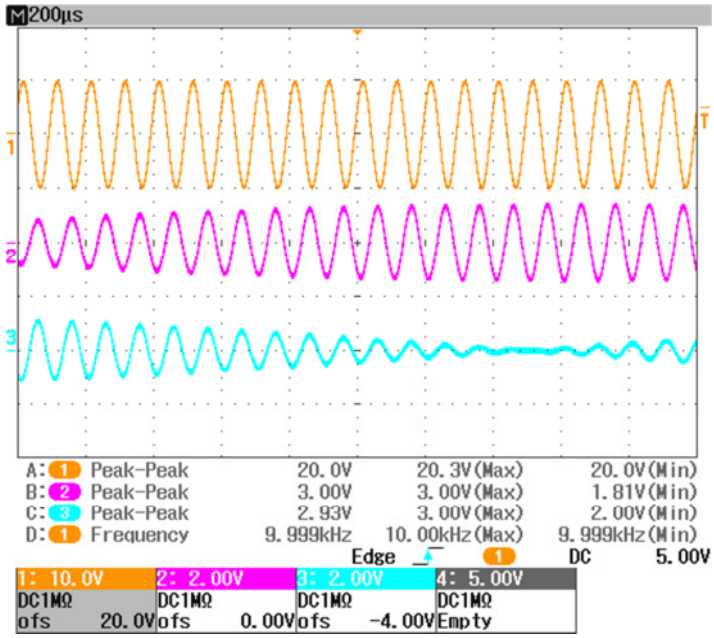


Figure 7: Input and output signals at 1000 rpm

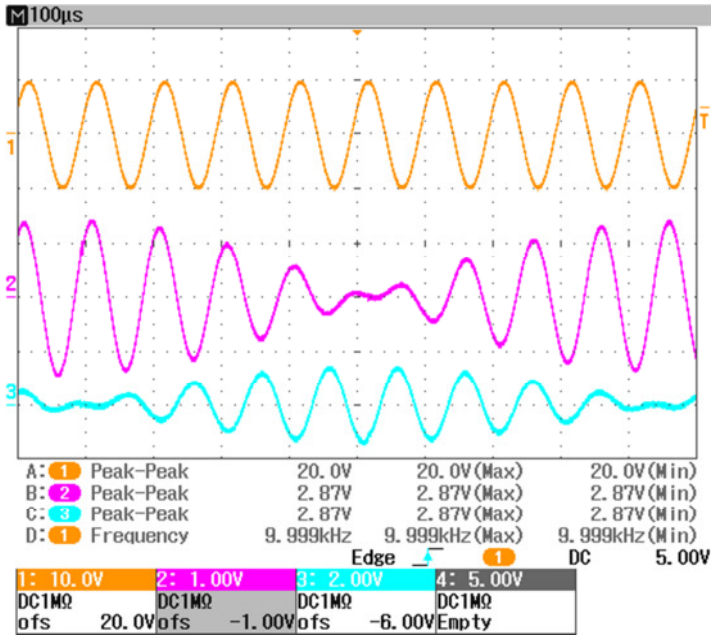


Figure 8: Details on switching point

4. Software

This chapter details the software part. The software of the simulation is mainly executed on a cRIO FPGA and can run on any Compact-RIO chassis. It is already compiled and tested for a NI-9074 and NI-9075 with the module at slot 1. The source code of the software is included in the module's documentation and can also be included into a user specific LabVIEW project.

In the LabVIEW project, there is also an example code included, how to control the module from the PC with LabVIEW. This example code for execution on a computer may be easily ported to real-time execution on the Compact-RIO real-time controller.

Furthermore a DLL is available to control the module from applications, written in C-code. An example software is included in the documentation CD, which describes how to use the DLL. Please note, that the DLL will work only with the respective FPGA-bitfile, which is already compiled for a cRIO-9075 chassis including a single module on slot 1.

Thus, the following description covers the chapters:

- General control
- Module control by DLL
- FPGA:
  - Interfaces
  - Including the code into a user defined FPGA-application
- Control of the module with LabVIEW code running on PC or LabVIEW-RT
- Control by executable GUI for stand-alone cRIO real-time system.

4.1 General Control

In the following chapter the general control parameters are described. Their names will be consequently used on different software layers, like FPGA top level design, PC control demo software or the DLL.

The amplitude control with programmable full scale transformation ratio is only available for HW version 2.0 or higher.

4.1.1 Mode and Speed control

The following parameters can be used to control the status and speed of the module by software

Parameter	Type		Description
Mode	Enum-eration	FixedPosition	Set to **fixed position**. When this mode is set, the module changes the actual position until it reaches the fixed position and holds this fixed position.
		Rotate_Right	**Rotate right** with speed, specified by **RPM_Set** value
		Rotate_Left	**Rotate left** with speed, specified by **RPM_Set** value
		Rotate_Right_Sync	**Rotate right** with speed, specified by **SYNC** input
		Rotate_Left_Sync	**Rotate left** with speed, specified by **SYNC** input
RPM_Set	Numeric Value to Set		**Speed** of rotation in RPM (rounds per minute) (*only in Mode Rotate_xxx*)
Position_Set [degrees]	Numeric Value to Set		constant position of the simulation in degree in the range 0…360° (*only in Mode FixedPosition*)
SYNC_Phase [degrees]	Numeric Value to Set		constant phase of the rotation relative to Sync input in the range 0…360° (*only in Mode Rotate_xxx_Sync*)
Speed_Gradient [RPMps]	Numeric Value to Set		Speed change rate in RPM per second, take new speed immediately, when 0
RPM_Actual	Numeric Value to Read		Actual **Speed** of rotation in RPM (rounds per minute)
Position_Act [degrees]	Numeric Value to Read		Actual position of the simulation in degree in the range 0…360°
RPM_SYNC	Numeric Value to Read		Actual **Speed** of Sync input in RPM (rounds per minute)

Table 6: Control Parameter description

The parameter RPM_Set has different meanings depending on the simulation state, as shown in the following table:

RPM_Set Value	Mode	RPM_Set meaning
> 0	FixedPosition	Defines the speed, how fast the module changes to the fixed position.
	Rotate_Right	Rotation speed
	Rotate_Left	
	Rotate_Right_Sync	Minimum rotation speed, when no SYNC signal is applied
	Rotate_Left_Sync	
= 0	Any	Set position to fixed position immediately.

Table 7: RPM meaning depending on current mode

4.1.2 Amplitude control

The following interfaces can be used to control the transformation ratio of the module. Please note, that **PGA_Set is only functional available for resolver simulation hardware version 2.0** or higher.

Parameter	Type		Description	
PGA_Set	Numeric value to set	0	0.078125	Maximum transformation ratio when sine LUT is defined to full scale (Amplitude 100%) Set this value to "2" for hardware version 1.3!
		1	0.15625	
		2	0.3125	
		3	0.625	
		4	1.25	
		5	2.5	
		6	5	
		7	10	
Write_LUT	Binary switch to initiate DMA transfer (see also DMA_FIFO)		Set "Write_LUT" to True, fill the DMA_FIFO. Reset "Write_LUT" to False after DMA has completed.	
DMA_FIFO	DMA Buffer to write a sine wave lookup table		Write a sine wave amplitude waveform with 1024 elements as 14-Bit integer values as shown in the following figure	

Table 8: Amplitude Control description

With PGA_Set the transformation ratio range may be specified. When this value is set, the ratio may be additionally fine tuned by writing the sine wave LUT via the DMA_FIFO. The sine wave must be represented by **1024 integer values (i16)** according to the following figure. This figure shows the sine wave for full scale - where the **maximum value is 16383** for full amplitude.

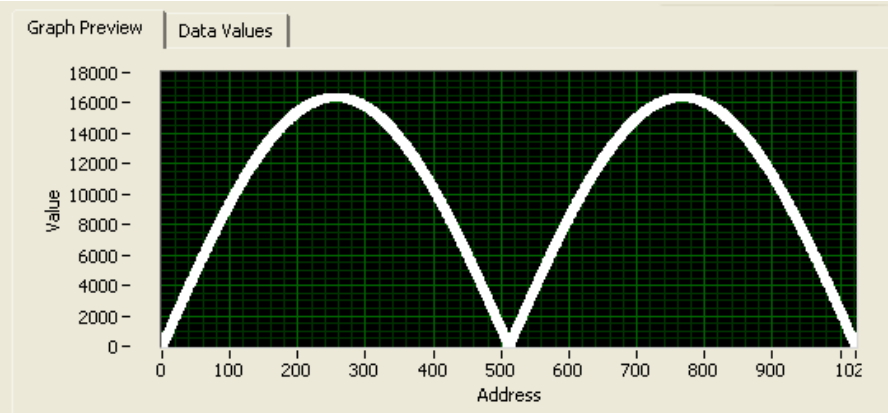


Figure 9: sine wave LUT

Use the LabVIEW-VI "LUT_Init" to create a standard sine wave for the DMA_Buffer with an adjustable Amplitude-factor of 0…100% as input parameter.

4.2 Control by DLL

The following functions are available to control the module with a DLL. The DLL requires that a Compact-RIO chassis is connected via Ethernet to the PC, where the DLL is running. The DLL has to be re-compiled using the Bitfile for the actual used chassis and module. The current DLL is available for cRIO-9075 with one module at slot 1.

Please note, that control by DLL is not implemented for all versions

To access the DLL by an application, the following files have to be included in the application:

- cRIO_ResolverSimulation.dll
- cRIO_ResolverSimulation.h
- cRIO_ResolverSimulation.lib

4.2.1 Communication control

4.2.1.1 Open connection

Following function opens the connection to the module:

```
int __stdcall cRIO_ResolverSim_Open  (char* sRIO_Device_IP_Address);
```

The parameters have the following meaning:

- `sRIO_Device_IP_Address`:
  - Zero-Terminated String, which contains the IP address of the cRIO-chassis's FPGA:
  - A typical address string is as follows: "192.168.101.88"
  - The IP-address of the Compact-RIO chassis may be changed using the NI-MAX "Measurement and Automation Explorer"
- `Return Value:`
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

4.2.1.2 Read Version

This function returns the compilation date of the FPGA design.

```
int __stdcall cRIO_ResolverSim_GetVersion    (unsigned int*    nVersion_Nr);
```

- nVersion_Nr:
  - compilation date of the FPGA code. Readable in hexadecimal representation 0xYYYYMMDD, e. g. 0x20130128
- Return Value:
  - The function returns an error code

4.2.1.3 Close connection

Following function closes the connection to the simulation module:

```
int __stdcall cRIO_ResolverSim_Close    (void);
```

- Return Value:
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

4.2.2 Get Error description

Following function converts the error code, returned by the DLL functions to a readable text.

```
int __stdcall cRIO_ResolverSim_GetErrorText  (int nError_Code, char* sError_String);
```

The parameters have the following meaning:

- nError_Code:
  - in this parameter the application may pass the error code returned by a DLL function
- sError_String:
  - in this parameter the description of the error is returned by copying data to this buffer
  - The application must provide enough space in the buffer (>1024 elements)
- Return Value:
  - The function always returns 0

4.2.3 Set simulation parameters

Following function sets the actual mode of the simulation

```
int __stdcall cRIO_ResolverSim_SetStatus  (
                                cRIO_ResolverSim_Mode   eMode,
                                double                  fPosition_degrees_Set,
                                double                  fRPM_Set,
                                double                  fSync_Phase_Set,
                                double                  fTransformationRatio);
```

The parameters have the following meaning:

- `eMode:`
  - This enumeration parameter defines the current simulation state according to table Table 6: Control Parameter description

```
typedef enum
{
   cRIO_ResolverSim_Mode_FixedPosition      = 0,
   cRIO_ResolverSim_Mode_Rotate_Right       = 1,
   cRIO_ResolverSim_Mode_Rotate_Left        = 2,
   cRIO_ResolverSim_Mode_Rotate_Right_sync  = 3,
   cRIO_ResolverSim_Mode_Rotate_Left_sync   = 4,
} cRIO_ResolverSim_Mode;
```

- `fPosition_degrees_Set:`
  - This is the position to set in °. The range may be 0…360°
  - This value is only valid if the current simulation mode is “FixedPosition” (see also chapter 4.1.1)
- `fRPM_Set:`
  - This rotating speed in rounds per minute (RPM) the meaning is depending on the actual mode as shown in the following table
  - The meaning of this value is dependent on the different modes (see 4.1.1).
- `fSync_Phase_Set:`
  - This function sets the phase of the rotation signal relative to the positive edge of the SYNC signal in°.
  - The range may be 0…360°
  - This value is only valid if the current simulation mode is “Rotate_Right_Sync” or in mode “Rotate_Left_Sync”. (see also chapter 4.1.1)
- `fTransformationRatio:`
  - This function sets the transformation ratio of the simulation.
  - Transformation ratio is converted in the two parameters of chapter 4.1.2
    - PGA_Setting
    - Amplitude setting of the LUT-Values
- `Return Value:`
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

See following table for the meaning of the speed parameter depending on the current mode:

fRPM_SetValue	Mode	fRPM_Set meaning
> 0	FixedPosition	Defines the speed, how fast the module changes to the fixed position.
	Rotate_Right	Rotation speed
	Rotate_Left	
	Rotate_Right_Sync	Minimum rotation speed, when no SYNC signal is applied
	Rotate_Left_Sync	
= 0	any	Set position to fixed position immediately.

Table 9: PRM Parameter meaning

4.2.4 Set Speed Ramp

Following function sets the rate for speed change in RPM per second. If the rate is 0, new speed settings are applied immediately. Otherwise, the speed will change at the specified rate.

```
int __stdcall cRIO_ResolverSim_SetRampGrad   (  double         fRPM_Gradient_Set )
```

The parameters have the following meaning:

- `fRPM_Gradient_Set:`
  - sets the actual ramp gradient in RPM per second
- `Return Value:`
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

4.2.5 Get simulation status

Following function reads the current status of the simulation. The status is described in the current position, the current rotation speed and current SYNC input speed.

```
int __stdcall cRIO_ResolverSim_GetStatus  (  double*     fSync_Frequency_RPM,
                                             double*     fActual_Frequency_RPM,
                                             double*     fPosition_degrees);
```

The parameters have the following meaning:

- `fSync_Frequency_RPM:`
  - This is the current speed of the synchronisation input in rounds per minute
- `fActual_Frequency_RPM:`
  - This is the current speed of the simulation in rounds per minute
- `fPosition_degrees:`
  - This is the current position in °. The range may be 0…360°
- `Return Value:`
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

4.2.6 Save settings

Following function saves the current settings in the cRIO system. After finishing the boot process after power-up the settings are restored automatically.

```
int __stdcall cRIO_ResolverSim_SaveSettings  (  void  );
```

The parameters have the following meaning:

- `Return Value:`
  - The function returns an error code
    - Zero => no error
    - Negative => error
    - Positive => Warning

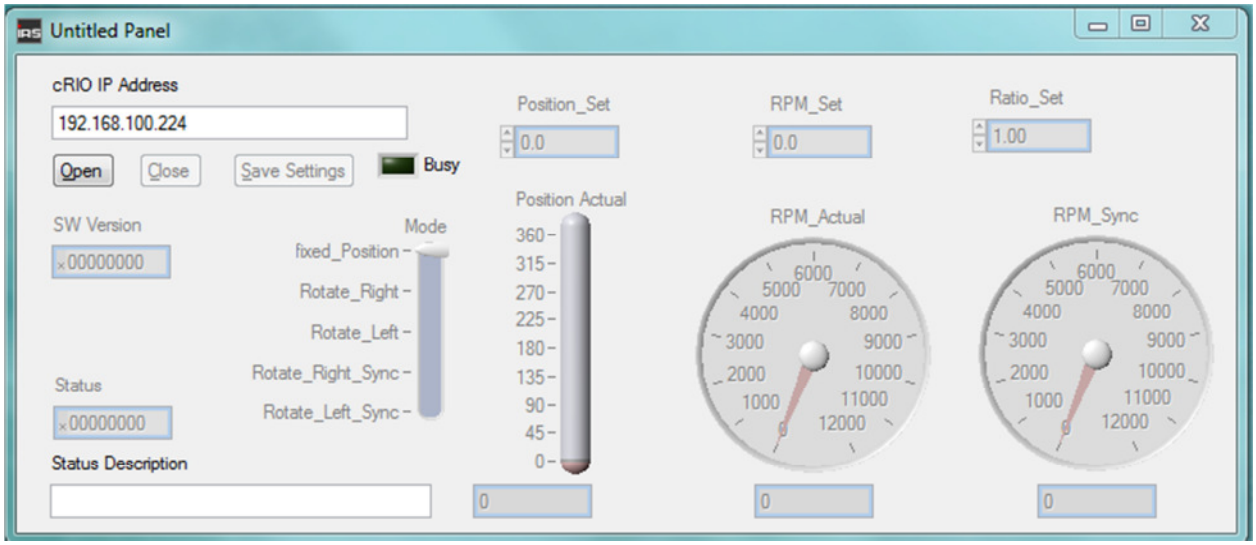
4.2.7 DLL Test software

A debug panel is included to test the access to resolver simulation by DLL. This tool is included as an installer in the folder

- CRIO_ResolverSim_DLL_Test_Install

In the bin folder on the documentation CD.

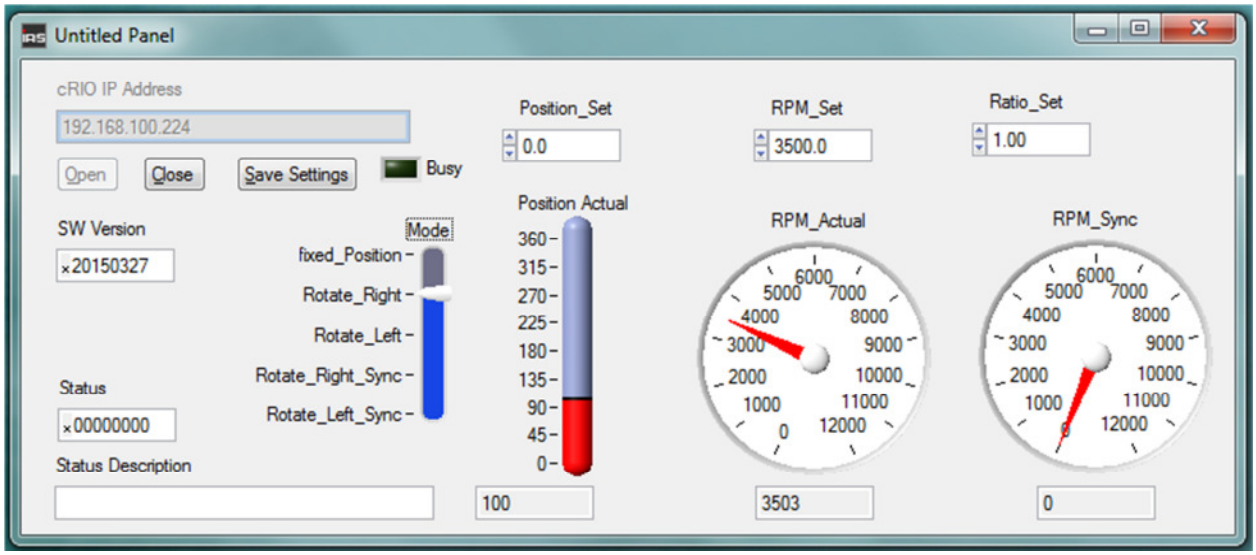
The following window appears after installation has been finished and software has been started:



After

- the correct IP address has been entered,
- a resolver simulation system is present in the network
- and has finished boot process (approx. 1 minute),

the connection may be established by pressing the “Open” button. If the connection is established, the window will appear as follows. The user may access the simulation by the controls.



4.3 FPGA

The cRIO module's software mainly runs in the FPGA of a Compact RIO system. It can run as a stand-alone application, ready compiled for NI-9075 and NI-9074 chassis. The FPGA code may be included into a user defined LabVIEW-FPGA application.

Following controls and indicators are available at the FPGA Top level design

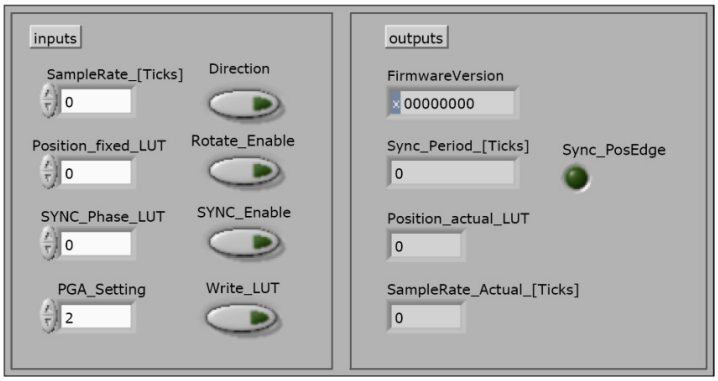


Figure 10: FPGA Top level design interfaces

4.3.1 Interfaces for speed and position control

Parameter	Direction	Data type	Description
Sample_Rate [Ticks]	IN	U32	Update rate of the output samples to define the rotating speed. When **RPM** is the desired speed in rounds per minute: Sample_Rate [Ticks] = (40e6 * 60 / (**RPM***1024)
Position_fixed_LUT	IN	U16	Sets a constant position of the simulation The value is in the range 0…1023, representing an angle of 0…360° Position_fixed_LUT = Angle * 1024/360
Direction	IN	BOOL	Sets sense of rotation FALSE -> Left TRUE -> Right
Rotate_Enable	IN	BOOL	TRUE -> a rotating machine is simulated. FALSE -> a fixed position is simulated
SYNC_Enable	IN	BOOL	TRUE: rotating speed derived from the SYNC input frequency FALSE: rotating speed derived from the Sample_Rate[Ticks] input
SYNC_Phase_LUT	IN	U16	Sets the 0° simulation value, relative to the positive edge of the SYNC input signal The value is in the range 0…1023, representing an angle of 0…360° SYNC_Phase_LUT = Angle * 1024/360
Speed_Gradient [RPMps]	IN	FXP24	Speed change rate in RPM per second, take new speed immediately, when 0. Otherwise 1 RPM/s up to 16e6 RPM/s
Sync_Period_[Ticks]	OUT	U32	Number of ticks between two positive edges on SYNC input (Ticks of 25ns)
Position_actual_LUT	OUT	U32	Actual position of the simulation.
FirmwareVersion	OUT	U32	Returns the compilation date, readable in hexadecimal: YYYYMMDD
SampleRate_Actual [Ticks]	OUT	U32	Current update rate of the output samples defining the rotating speed. The actual speed RPM is calculated: RPM = 40e6 * 60 / (1024 * SampleRate_Actual [Ticks])

Table 10: FPGA Parameter description (speed and position)

Please note, that the input and outputs are handled as “raw” data to keep size of the FPGA code low and efficiency high, since calculations of positions in degrees or speeds in RPM can be easy calculated on a computer. These calculations would consume some resources of the FPGA. The calculation of the values is explained in the following table.

Depending on the Boolean controls the functionality is as described in the following table:

Rotate Enable	Direction	SYNC Enable	Functional state
0			Set to fixed position. When this flag is set, the module changes the actual position until it reaches the fixed position “*Position_fixed_LUT*”
1	**0**	**0**	Rotate left with speed, specified by “*Sample_Rate [Ticks]*”
1	**1**	**0**	Rotate right with speed, specified by “*Sample_Rate [Ticks]*”
1	**0**	**1**	Rotate left with speed, specified by SYNC input
1	**1**	**1**	Rotate right with speed, specified by SYNC input

Table 11: Module state

The parameter “*Sample_Rate [Ticks]*” is derived from the Parameter RPM_Set and has influence on the functional state of the module according to Table 7: RPM meaning depending on current mode.

4.3.2 Amplitude control interfaces

The following interfaces can be used to control the transformation ratio of the module. Please note, that **PGA_Set is only functional available for resolver simulation hardware version 2.0** or higher.

Parameter	Type		Description	
PGA_Set	Numeric value to set	0	0.078125	Maximum transformation ratio when sine LUT is defined to full scale Set this value to “2” for hardware version 1.3!
		1	0.15625	
		2	0.3125	
		3	0.625	
		4	1.25	
		5	2.5	
		6	5	
		7	10	
Write_LUT	Binary switch to initiate DMA transfer (see also DMA_FIFO)		Set “Write_LUT” to True, fill the DMA_FIFO. Reset “Write_LUT” to False after DMA has completed.	
DMA_FIFO	DMA Buffer to write a sine wave lookup table		Write a sine wave amplitude waveform with 1024 elements as 14-Bit integer values as shown in the following figure	

Table 12: FPGA Parameter description (amplitude control)

With PGA_Set the transformation ratio range may be specified. When this value is set, the ratio may be additionally fine tuned by writing the sine wave LUT via the DMA_FIFO.

The sine wave must be represented by **1024 integer values (i16)** according to the following figure. This figure shows the sine wave for full scale - where the **maximum value is 16383** for full amplitude.

Use the LabVIEW-VI “LUT_Init” to create a standard sine wave for the DMA_Buffer with an adjustable Amplitude-factor of 0…100% as input parameter. See also chapter 4.1.2 for detailed description.

4.3.3 Integration in user FPGA application

When several modules or a different Compact-RIO chassis have to be applied, the user may integrate

NI-RIO 4.0 (or higher) should be installed on the Compact-RIO.

In order to use the module in user applications first of

- all the required signals (marked in the next figure)
- as well as the memories (SPI FIFO block memory and look-up table)
- and clock settings

must be copied from the sample project to the new application. Then the module must be added to the project. The next figure also shows that.

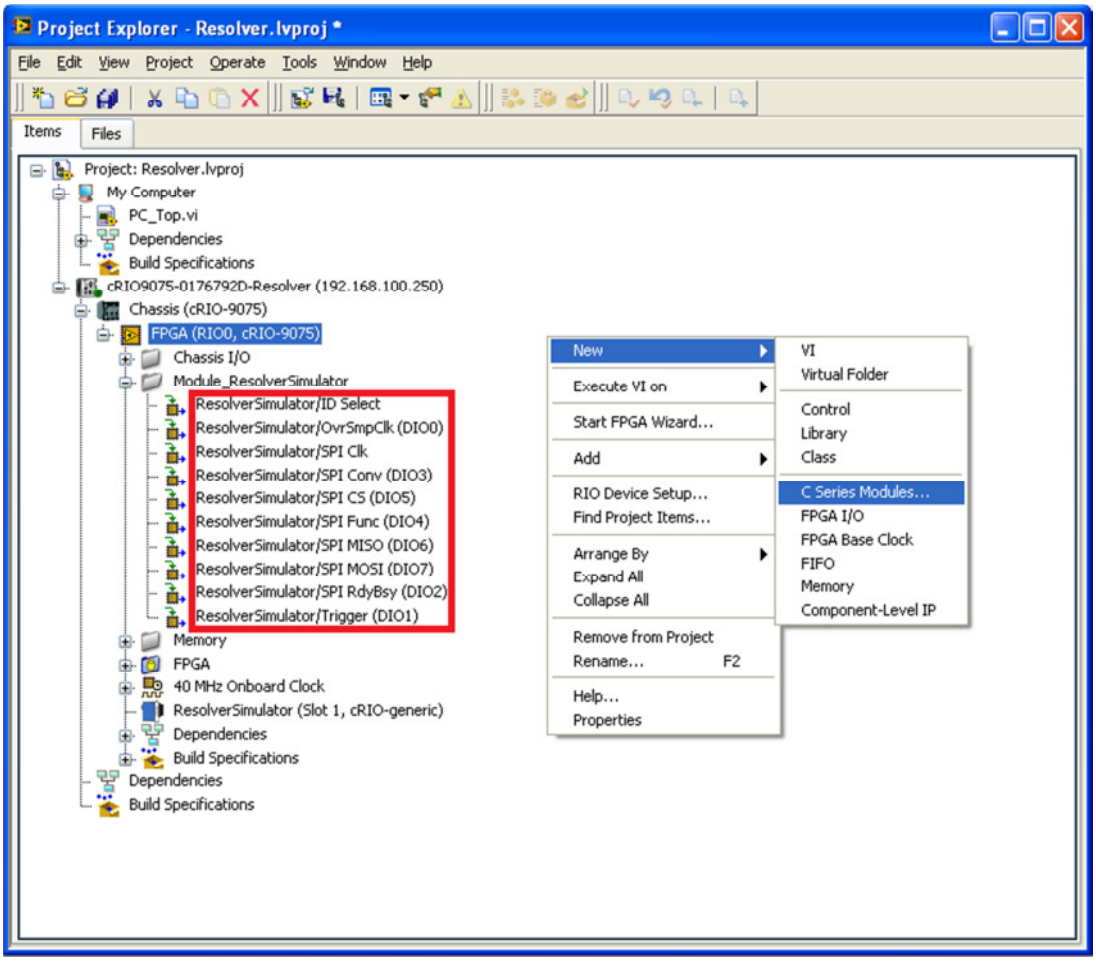


Figure 11: Project explorer

In the next window select *New target or device*, then *C Series Module*. In the next window type in a module name and select the required slot. The module type must be set to *cRIO-generic*. If this option is not available, the following line must be added to the LabVIEW.ini file (LabVIEW must be closed).

```
cRIO_FavoriteBrand=generic
```

It is recommended to rename some default signal names as it can be seen in the next table.

Default name	Recommended name
OvrSmpClk (DIO0)	FPGA_IO SYNC input
SPI Func (DIO4)	FPGA_IO COS Sign
SPI RdyBsy (DIO2)	FPGA_IO SIN Sign
SPI Conv (DIO3)	FPGA_IO PGA G0
Trigger (DIO1)	FPGA_IO PGA G1
SPI MISO (DIO6)	FPGA_IO PGA G2

Table 13: FPGA IO special function names

The FPGA Top level design runs in two Single-Cycle Timed Loops. If the design has to be integrated into a user application, you should care about the following

- Route FPGA-IOs to the module's corresponding IO in the project.
- Make sure that the "Resolver_Main" runs in the 40Mhz clock domain
- Make sure that the "SPI-Handler" runs in the SPI clock domain (48MHz)

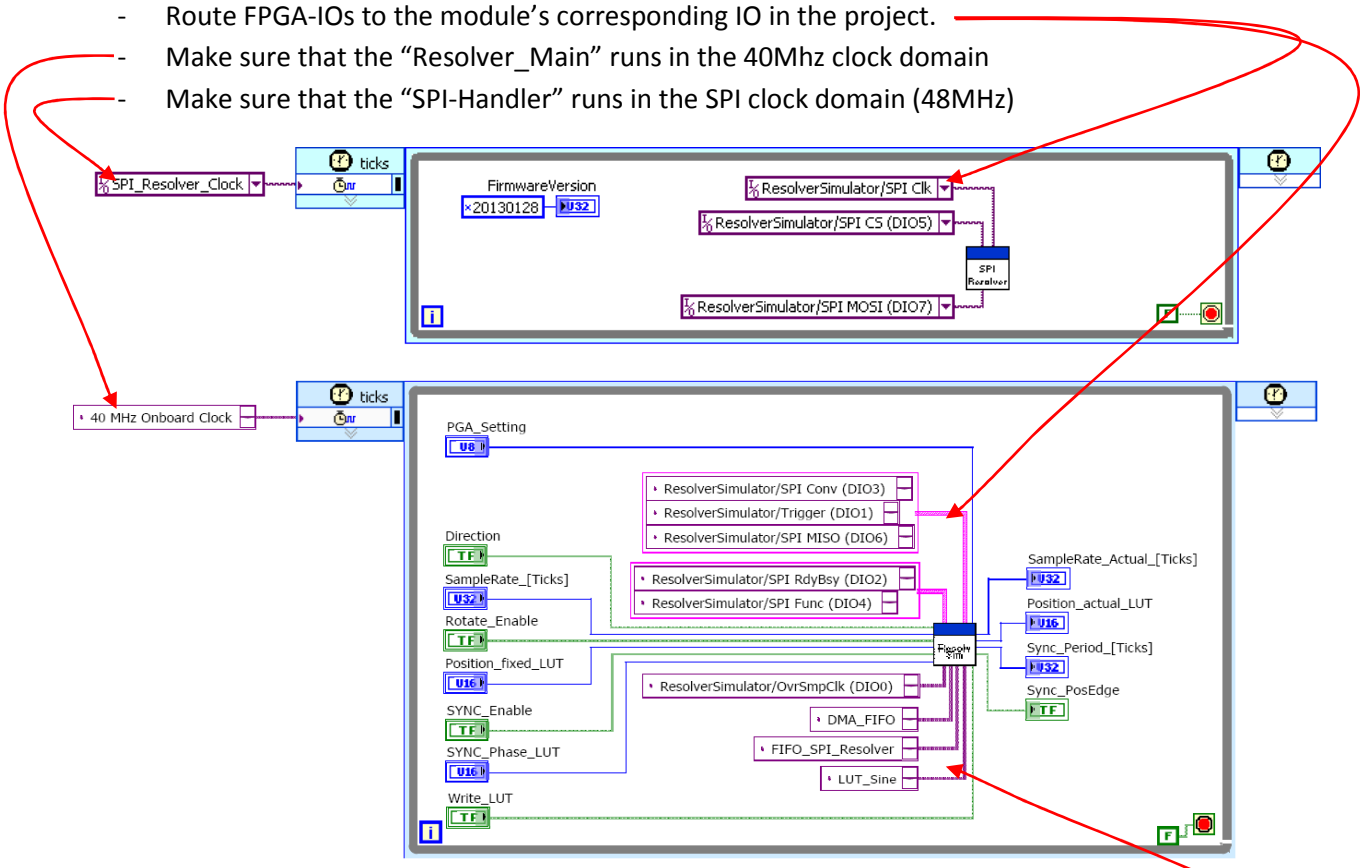


Figure 12: Top-VI interfaces

- Route Memory references to the defined memories in the project:

Memory Name	Type	Elements	Data Type	Remark
DMA_FIFO	Host to Target DMA	1024	I16	FIFO to update LUT_Sine from PC
FIFO_SPI_Resolver	Target scoped FIFO (Block memory)	37	U32	FIFO to transfer samples to SPI
LUT_Sine	Block memory	1024	I16	Memory holding the sine wave samples

Table 14: FPGA memory description

4.3.4 Integration of several modules

The user may integrate several modules into the LabVIEW-FPGA application. An example code for NI9074-chassis is included on the documentation and software CD. Following figure illustrates the project window and the block diagram:

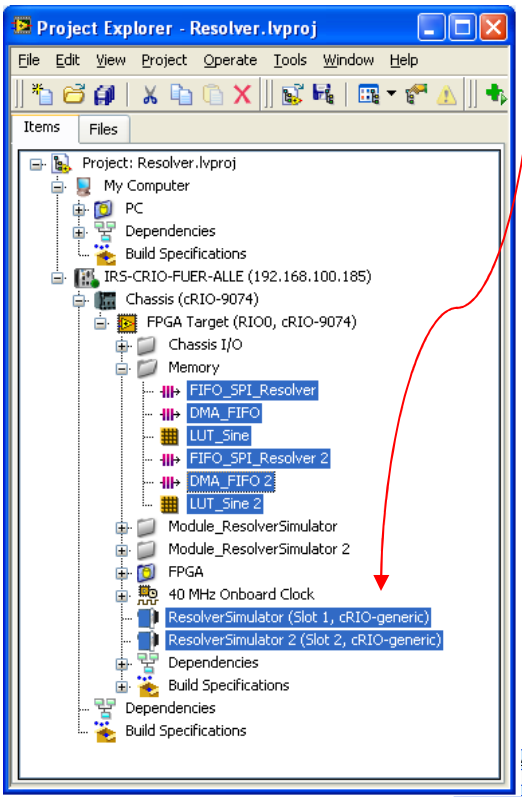


Figure 13: Two modules in one chassis

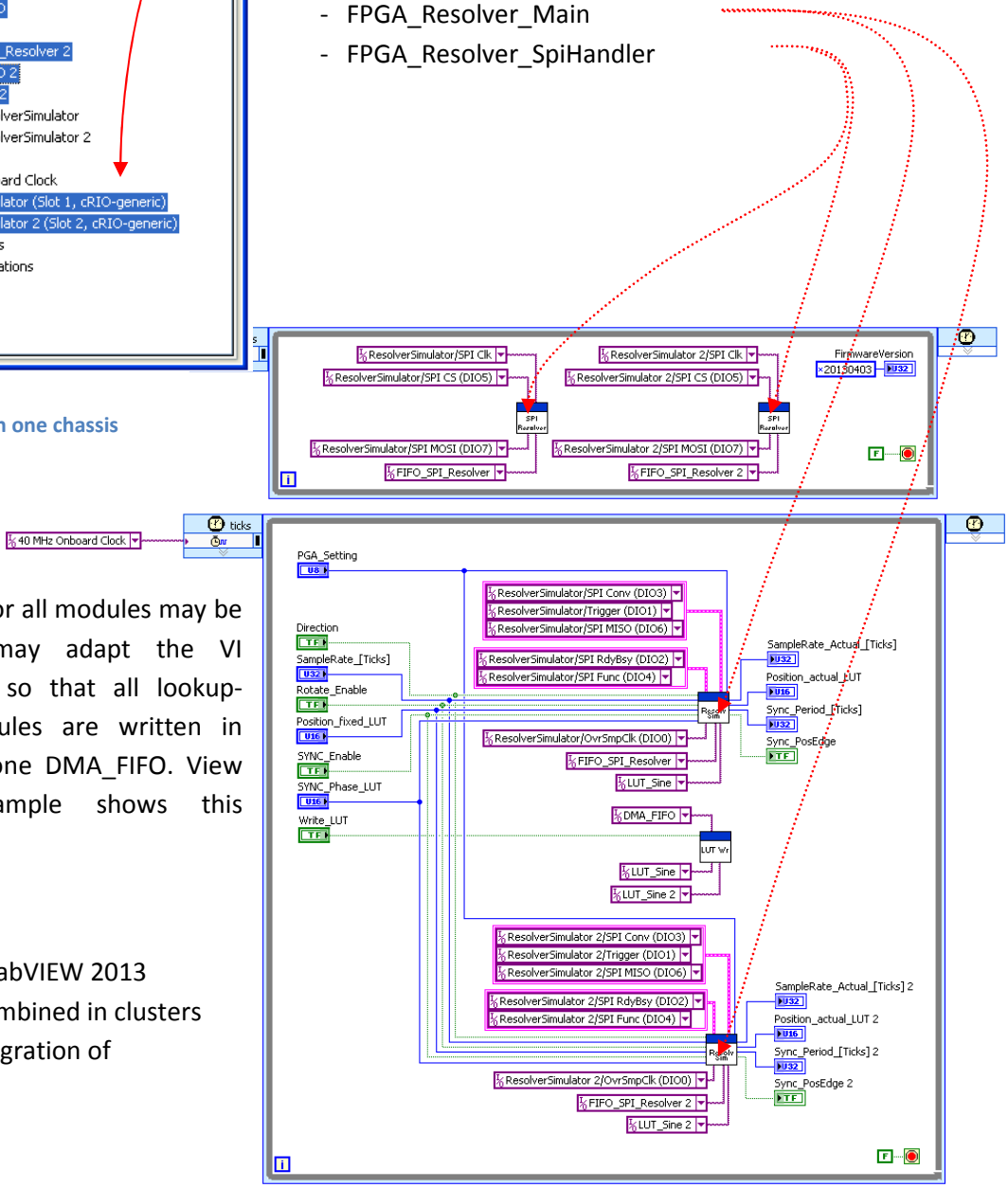
Every module has to be defined in the project window.

For every module following memories must be available once:

- LUT_sine
- FIFO_SPI_Resolver

In the block diagram following Vis must be included once for every module:

- FPGA_Resolver_Main
- FPGA_Resolver_SpiHandler



When the sine LUT for all modules may be equal, the user may adapt the VI FPGA_SubWriteLUT, so that all lookup-tables for all modules are written in parallel using only one DMA_FIFO. View the LabVIEW example shows this procedure.

Please note, that in LabVIEW 2013 some controls are combined in clusters to provide easier integration of several modules.

4.4 Direct FPGA-Control

The module may be controlled by a LabVIEW application running either on a PC or on the real-time controller on the Compact-RIO chassis. The LabVIEW application may directly access the FPGA controls. Please note, that in LabVIEW 2013 examples some controls are combined in clusters to provide easier integration of several modules.

4.4.1 Simple Example

The FPGA-IO's, described in chapter 4.3.1 can be accessed by LabVIEW. Simple demo software is provided, which includes the general control functions as described in chapter 4.1.

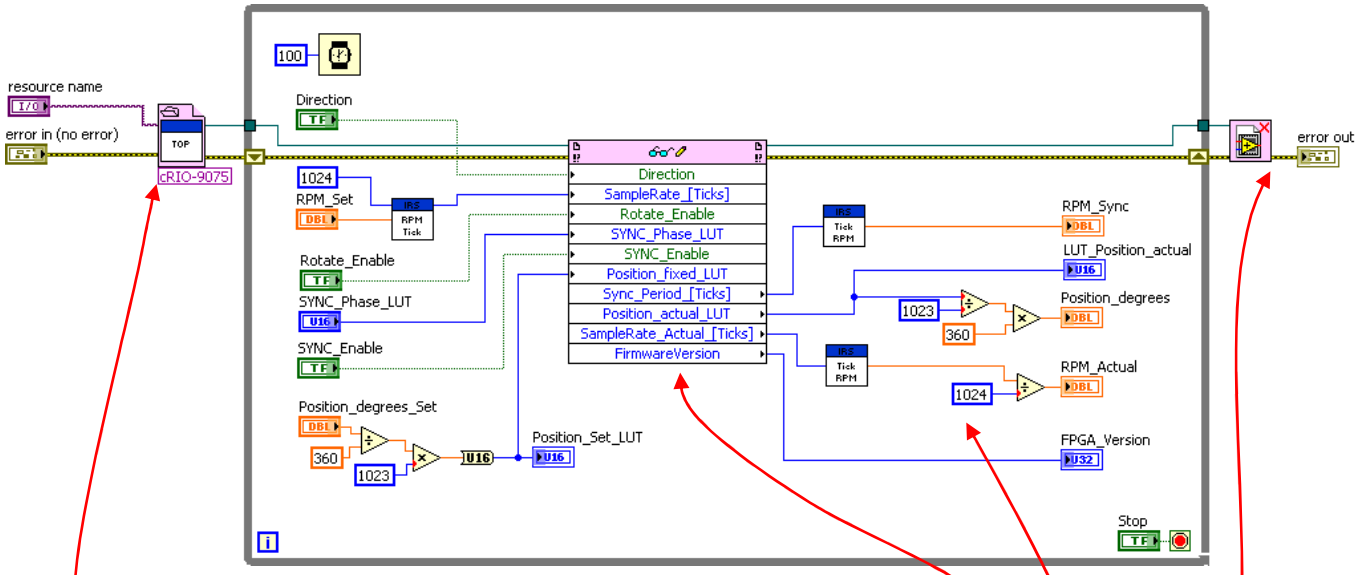


Figure 14: Simple Example VI

- First OPEN reference to the FPGA bitfile.
- Controls and indicators are available as described in chapter 4.3.1
- The input and output values have to be scaled in an appropriate way.
- The reference must be closed on exit.

The example VI can be run on a PC or directly on the Compact-RIO RT-controller. The only difference is the resource name, which has to be adapted to

- "**rio://RIO0**" in case of RT
- "**rio://<IP-Address>/RIO0**" in case of PC (e.g. "rio://192.168.101.88/RIO0")

4.4.2 Amplitude control

The simulation's output amplitude may be adjusted by software by changing the transformation ratio of the module. There are two interfaces available to change the transformation ratio:

- Changing the sine wave amplitude by changing the sine LUT
- Changing the full scale maximum by setting a programmable amplifier.

Changing the PGA setting can be done by adjusting the control "PGA_Set" (see also chapter 4.1.2 for detailed description).

Following figure shows an example, how to "fine tune" the transformation ratio by changing the sine wave LUT:

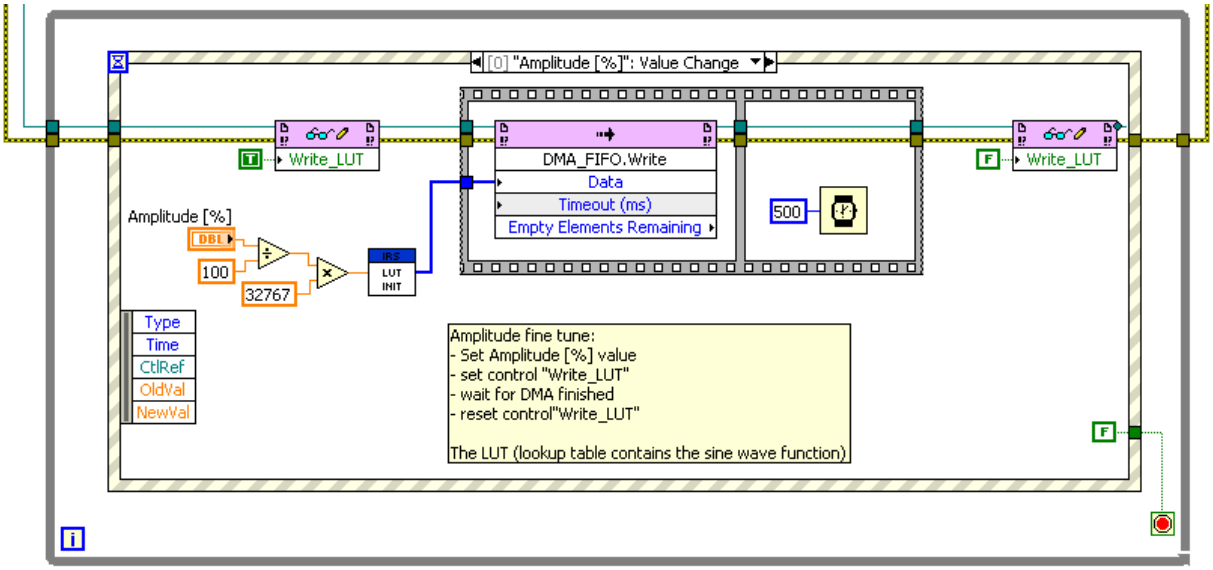


Figure 15: Amplitude change Example

4.5 GUI for stand-alone system

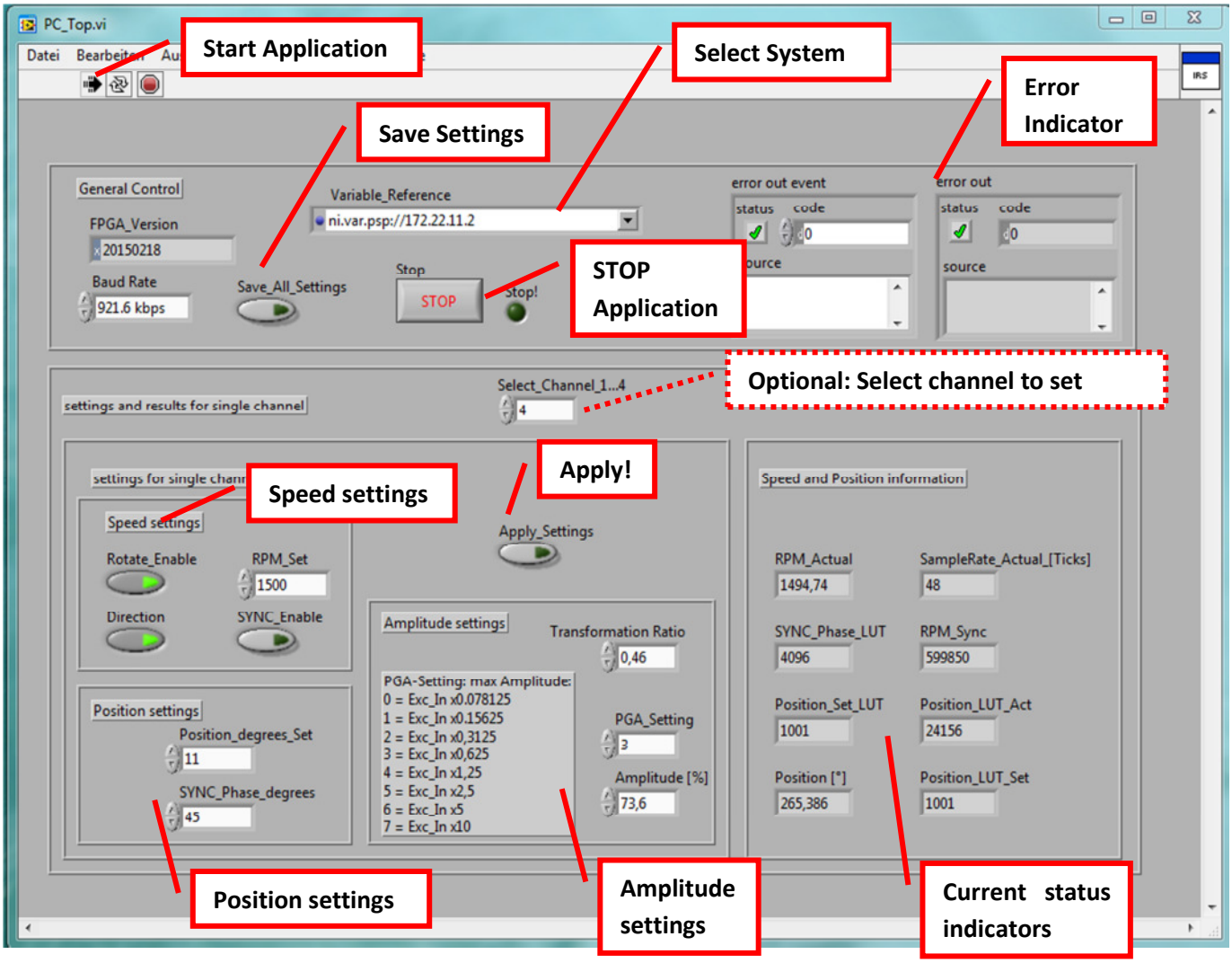
For changing application specific settings a custom graphical user interface (GUI) is implemented.

4.5.1 Installation

To run the GUI, an executable “ResSi.exe” can be found on the documentation disc. The entire folder, which contains this executable should be copied to the PC.

Additionally, LabVIEW 2013 runtime has to be installed. The runtime is available on the documentation disc, but may be also downloaded at NI.com.

4.5.2 Front Panel



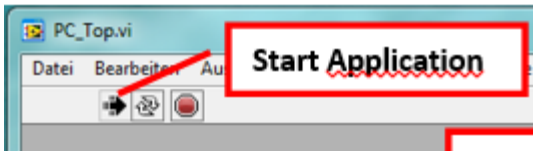
All changes of the settings are only applied, when:

- GUI is running (see chapter 4.5.3)
- System is properly selected (see chapter 4.5.4) – otherwise GUI stops with error.

4.5.3 Start/Stop GUI

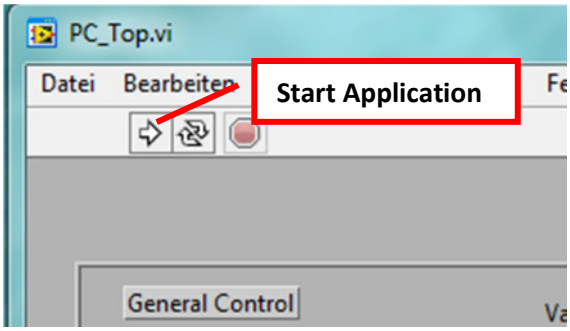
The GUI automatically starts, when “ResSi.exe” is executed.

The “Start application” button is shown as in the following figure, when the GUI is running:

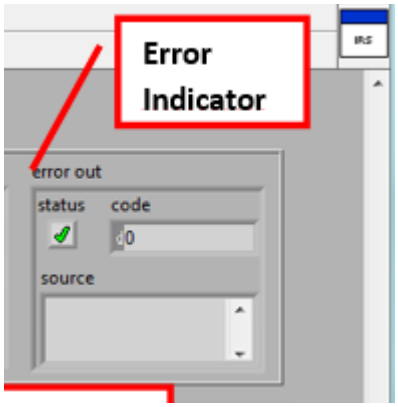


To stop the GUI, press the STOP-button.

To restart GUI, press the “Start Application” button on the upper left corner.



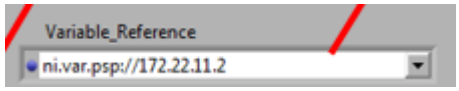
The GUI automatically stops, when an error occurs. An error is signalled and explained on the error_out indicator



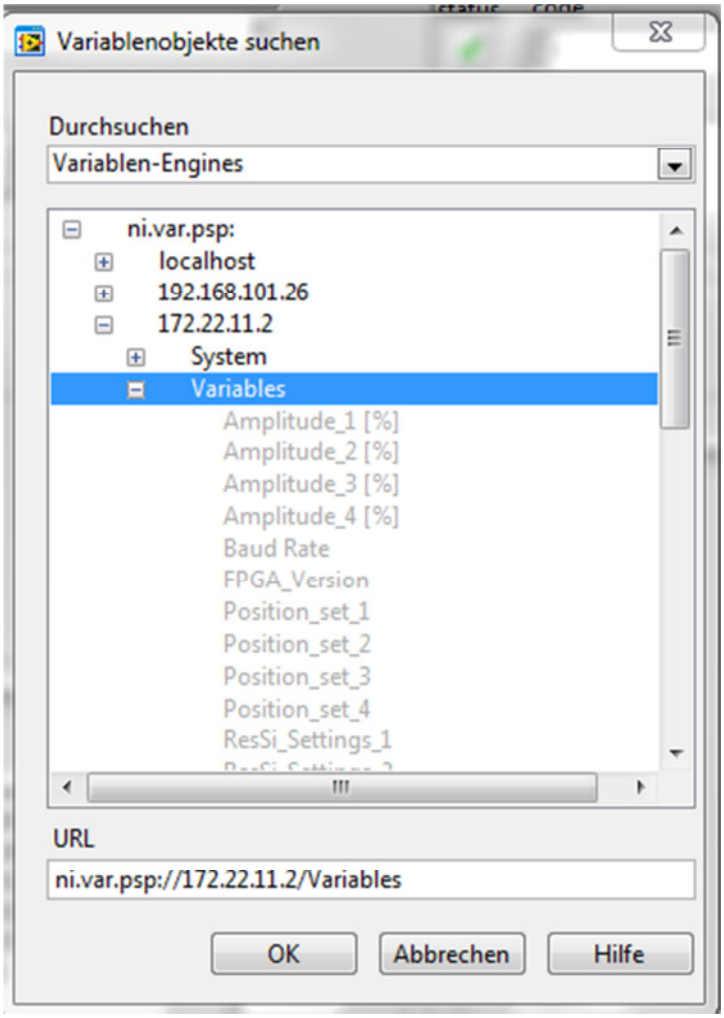
If the system is not properly selected, an error occurs and the GUI is stopped. Select the system according to chapter 4.5.4 and restart GUI by pressing the “Start Application” button.

4.5.4 Select System

Accessing the settings of the system is achieved using NI-PSP shared variables. The variable server runs on the cRIO system. Since the IP-address of the system may vary, the user must select the correct system.

Press the “Variable Reference” 

The following window should appear:



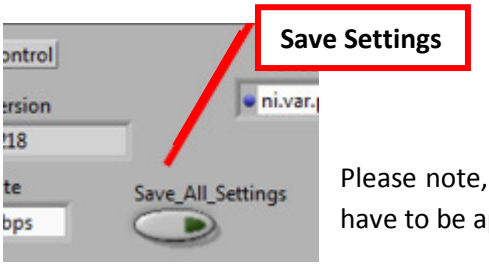
Select the item “Variables” on the system with the correct IP-address.

- 172.22.11.2 in case of USB devices
- For Ethernet depending on DHCP

If the system is properly selected, the GUI will run, if there is an error, the GUI stops and indicates an error. In the latter case, make sure that all connections are established and the system is powered. After selecting the system restart the GUI as described in chapter 0.

4.5.5 Save all settings

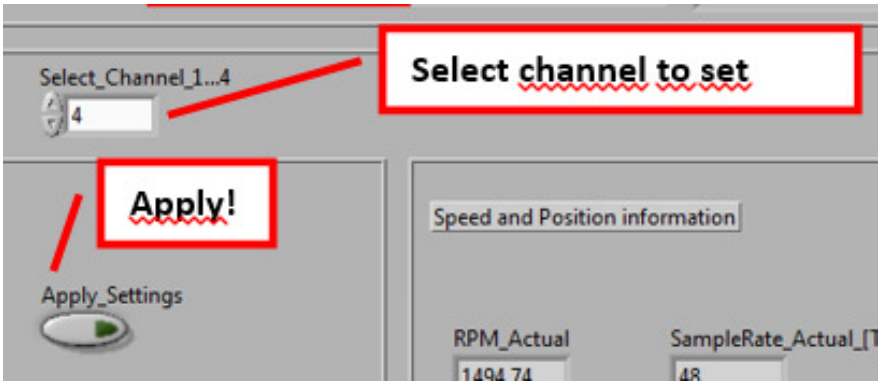
Current settings may be saved to be active after reboot again. If the button “Save_All_Settings” is pressed, all settings are stored permanently in the system.



Please note, that current settings of the system are stored. I.e. the settings have to be applied before saving according to chapter 4.5.6.

4.5.6 Change Resolver settings

All following settings may be changed in the GUI. If several Resolver modules are applied, settings take effect only for a single selected channel, after the “Apply_Settings” button has been pressed.



I.e. first “Select_channel_1…4” (Optional for several modules)

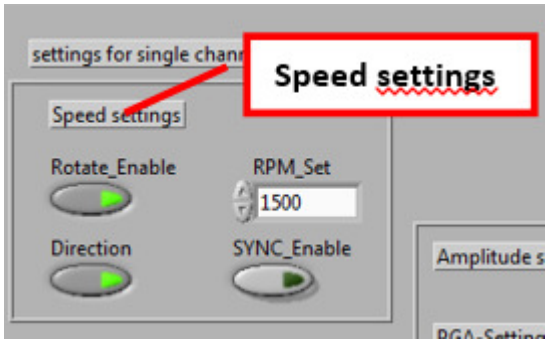
Then change settings as described in the following sub-chapters

Then press “Apply_Settings”.

If settings should be stored permanently, press “Save_All_Settings”

4.5.6.1 Speed settings

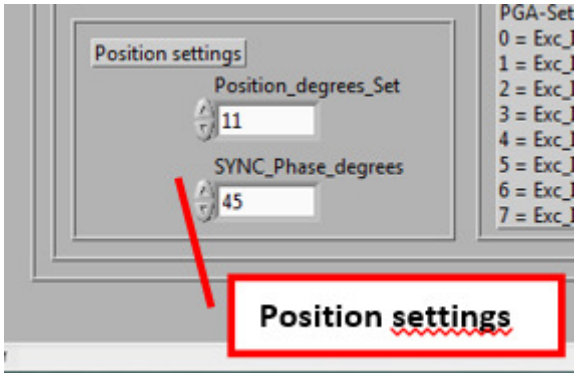
The speed settings include the following:



- Rotate_Enable:
  - continuously rotate, when TRUE
  - approach fixed position when FALSE.
- Direction:
  - Rotate "left" with increasing angle when TRUE
  - Rotate "right" with decreasing angle when FALSE
- RPM_Set:
  - Sets the speed of rotation in "rounds per minute", when SYNC_Enable is FALSE.
  - When rotation is disabled, RPM_Set defines the maximum speed to approach a fixed position, when fixed position changes.
- SYNC_Enable:
  - Use external synchronisation input when TRUE
    - Time between positive edges on SYNC input defines speed
    - Note, that the speed is limited to RPM_Set, when RPM_Set is higher than SYNC
  - Use RPM_Set as speed value when FALSE.

4.5.6.2 Position settings

Position settings include the following:

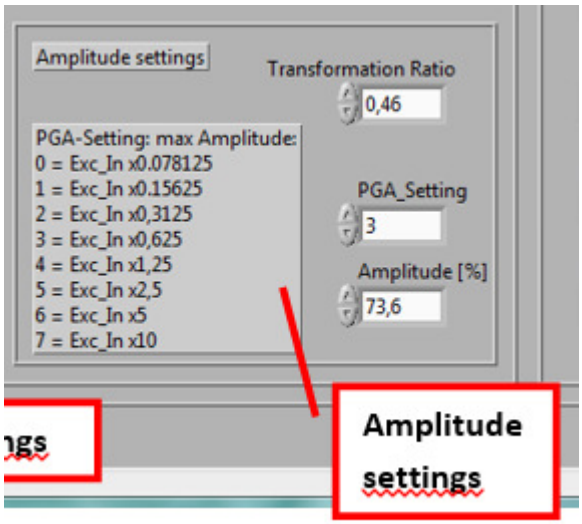


- Position_degrees_Set:
  - Position to set in the range 0…360°
  - Only valid when rotation is disabled
  - and no position is received on serial line for this channel.
- SYNC_Phase_degrees:
  - Angle in the range 0…360°

4.5.6.3 Amplitude settings

The transformation ratio of a channel may be changed by Amplitude settings. There are two values, which define the transformation ratio:

- PGA_Setting:
  - programmable gain amplifier defines the maximum range of the transformation ratio
  - Setting between 0…7 according to the table in the following figure
- Amplitude [%]:
  - Defines the fine tuning within the current range, defined by the PGA
  - Set in the range 0…100%.

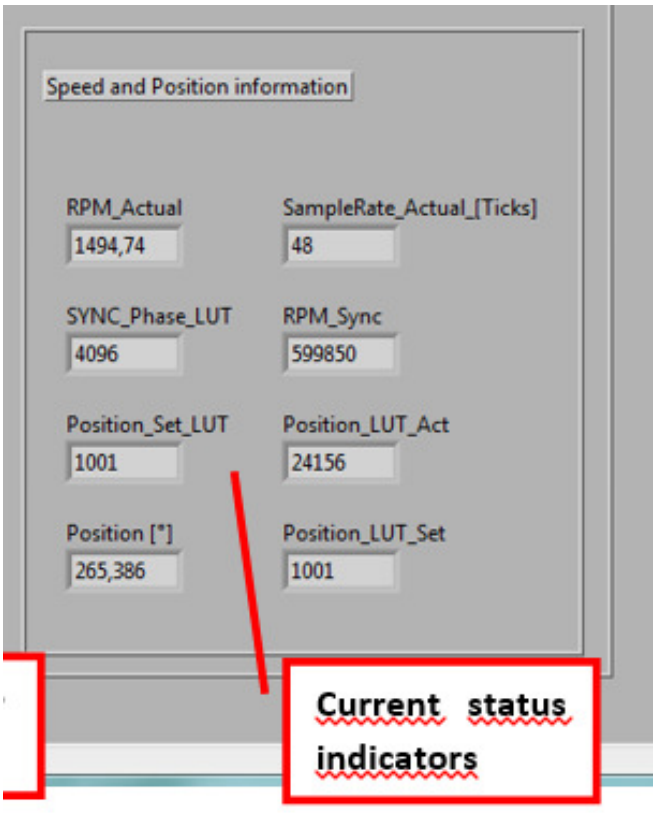


These two values are written to the system and really define the transformation ratio.

To make amplitude settings easy, a Transformation ratio may be entered. When a new value for “Transformation_Ratio” is entered, “PGA_Setting” and “Amplitude [%]” are calculated automatically to an optimal value.

4.5.7 Current Status indicators

Current status indicators are as follows:



- RPM_actual:
  - Current speed, depending on
  - Either RPM_Set
  - or synchronisation input speed.
- SYNC_Phase_LUT
  - Internal representation of SYNC_Phase
- Position_Set_LUT
  - Internal representation of fixed position
- Current position [°]
  - Current Position in the range 0…360°
- Sample_Rate_Actual
  - Internal speed representation
  - =25ns-Ticks for a single step
- RPM_Sync
  - Speed of the external SYNC input in RPM
- Position_LUT_Act
  - Current Position (internal value)
- Position_LUT_Set
  - Internal representation of fixed position
  - Either fixed setting or received data

4.6 VIs for integration into user application

To set simulation parameters or read current status, two main VIs may be used to be integrated in a user application. Both VIs are available in the Stand-alone LabVIEW-example GUI.

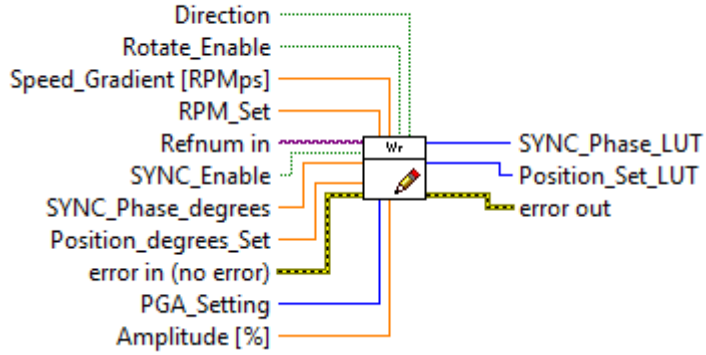
These VIs don't need an open or close VI. They may be called by any LabVIEW application or TestStand Sequence.

4.6.1 PC_ResSi_Wr.vi

PC_ResSi_Wr.vi sets the new status of the resolver simulation. The input parameters are described in chapter 4.1. The output parameters ..._LUT are internal representations of the settings. These outputs may be ignored.

For the "Refnum in", please connect a valid reference to the device according to chapter 4.5.4

PC_ResSi_Wr.vi



4.6.2 PC_ResSi_Rd.vi

PC_ResSi_Rd.vi reads the current status of the resolver simulation. The output parameters are described in chapter 4.1

For the "Refnum in", please connect a valid reference to the device according to chapter 4.5.4

PC_ResSi_Rd.vi