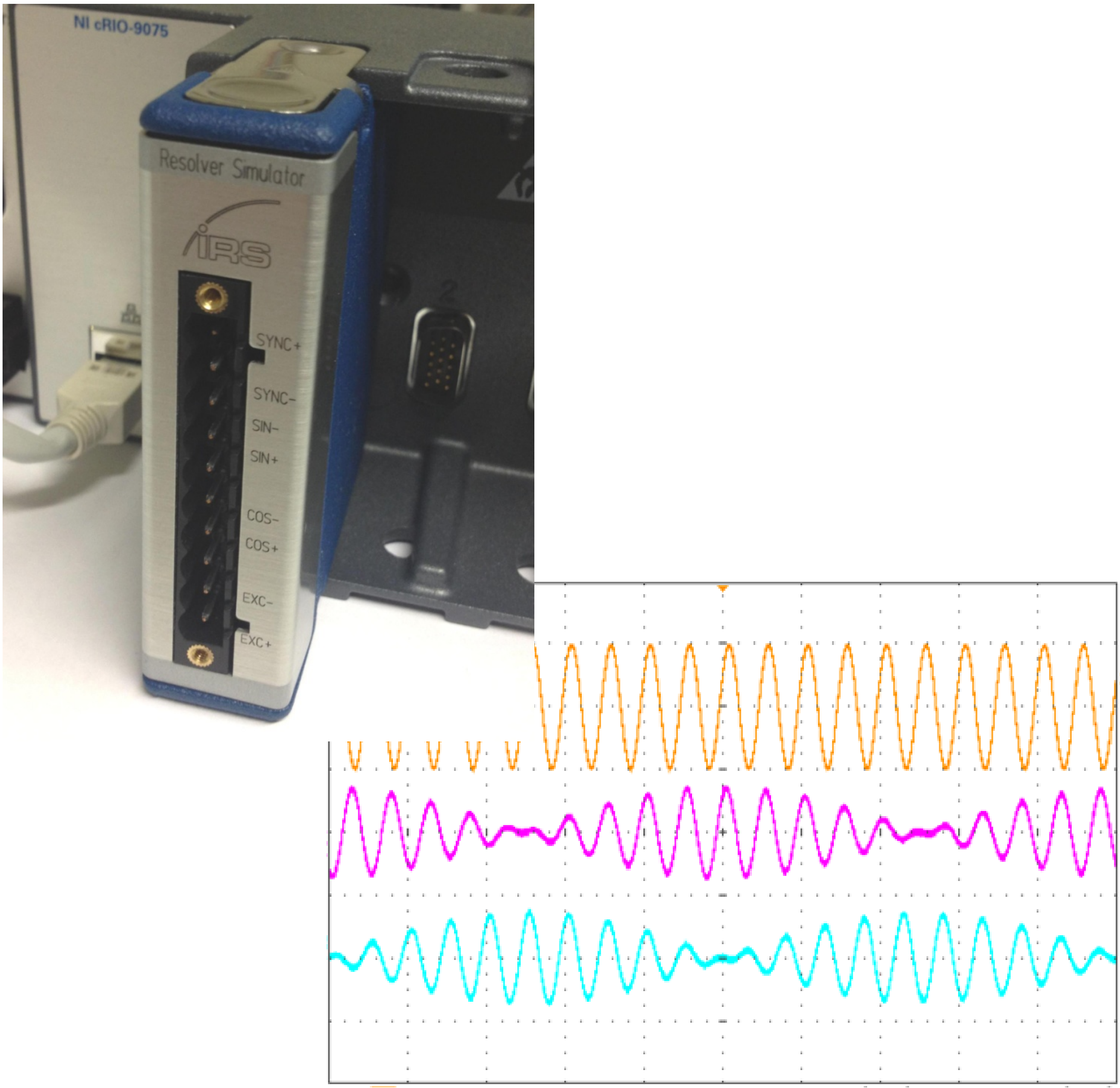
cRIO Resolver Simulation
cRIO RVDT Simulation

Manual
V3.0



Content

1. 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
29.11.2016	3.0	- Added RVDT Support - Updated FPGA (New TOP-VI, Prepared multichannel support) - New software part (RVDT mode, ResSi mode still missing)

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
ResSi	Resolver Simulation
RVDT	Rotary Variable Differential Transformer

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.

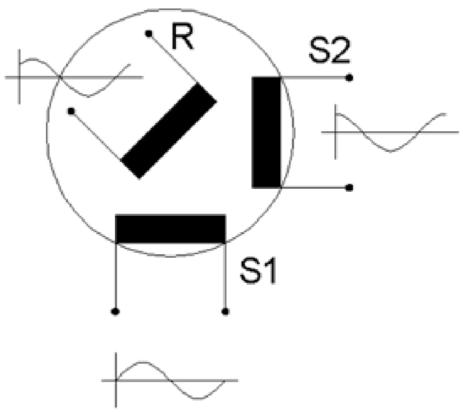


Figure 1: Resolver principle

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

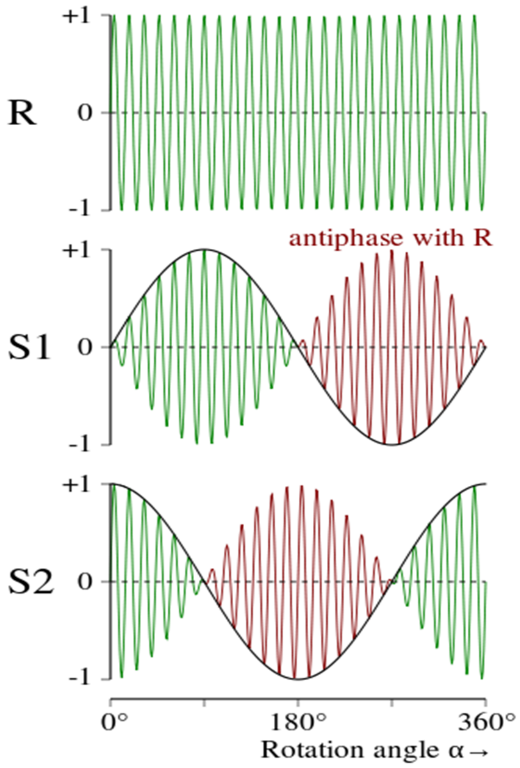


Figure 2: Resolver signal

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:

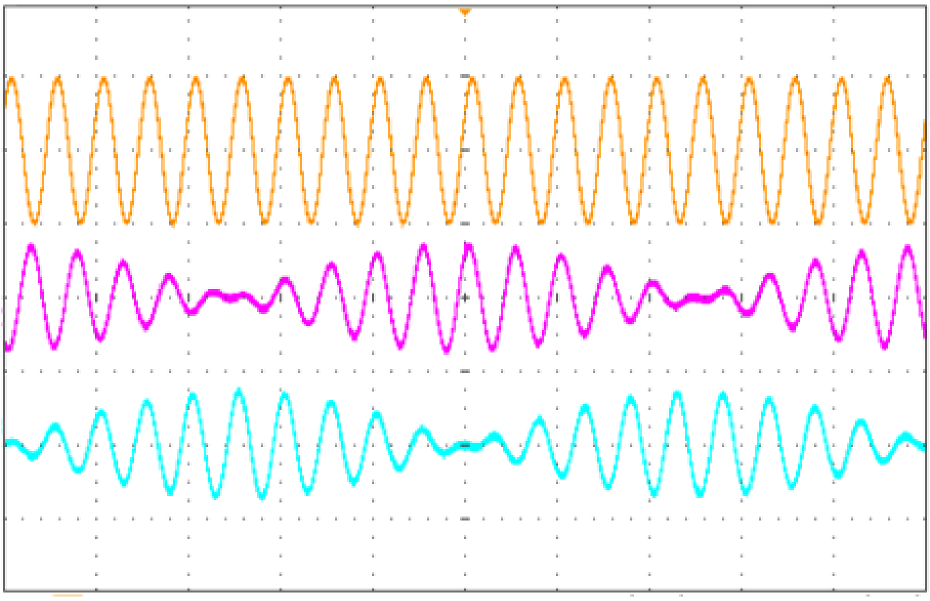


Figure 3: typical simulation waveform

2.3 RVDT sensor

RVDT (rotary variable differential transformer) is a transformer used for measuring angular displacements. Following figure shows the principle of a typical sensor, where the metal core is rotatable in a limited angular range. The two windings on side “B” receive the energy from the excitation on side “A” and generate a signal with an amplitude depending on the core position. The displacement direction (sign) can be determined by the phase of the output signal compared to the excitation signal (positive angle → 0°; negative angle → 180°).

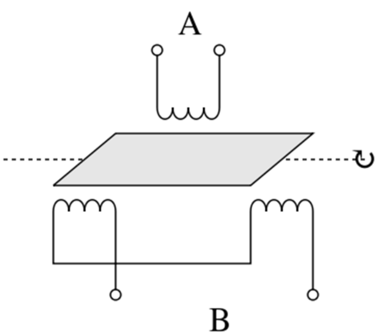


Figure 4: RVDT principle

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.

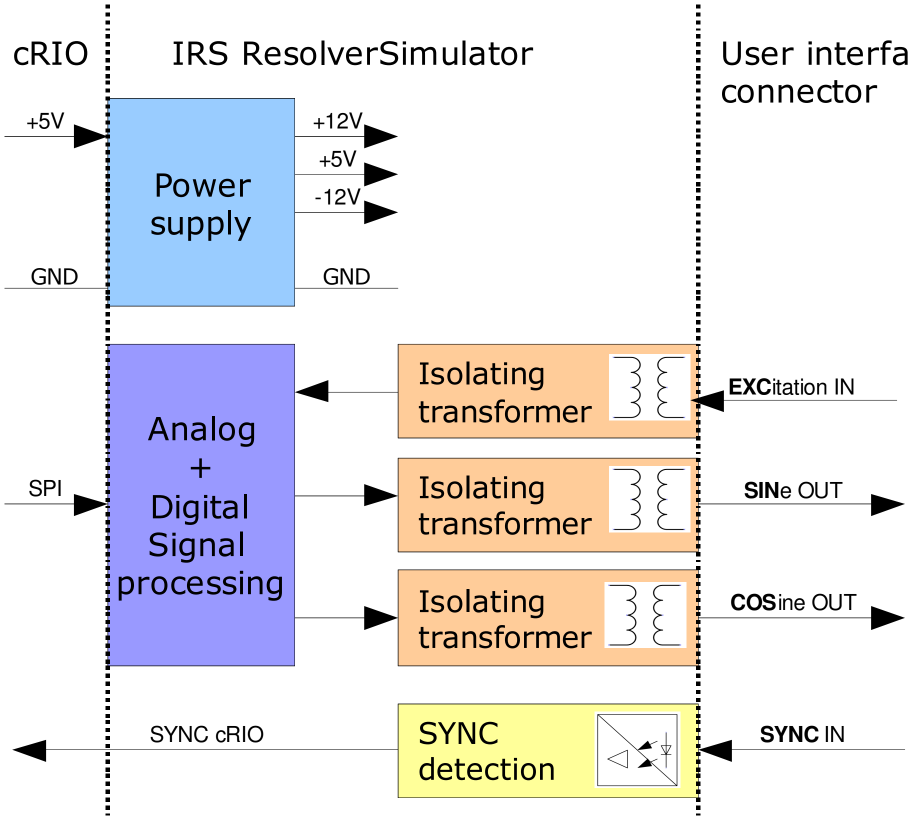


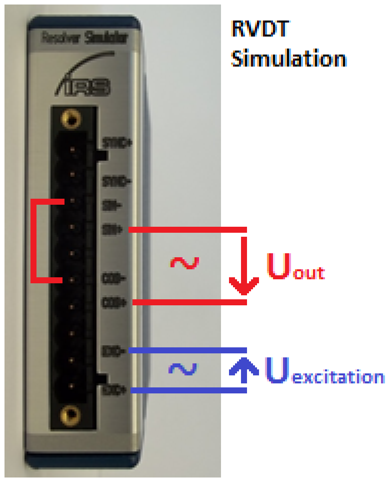
Figure 5: Block diagram

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.
  - In **RVDT** mode the output signal is taken between Sin+ and Cos+ (Sin- and Cos- needs to be tied together) → Difference output signal:



- **SYNC input (ResSi only)**:
  - This signal can be used to synchronize the modulation frequency of sine and cosine outputs to a speed signal (not used in RVDT mode).
  - I.e. RPM (speed) of the motor can be applied here.

3.3 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 +

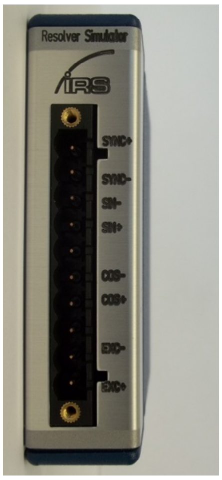


Table 4: Pin out front connector

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.

3.4 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)		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 5: Technical data

3.5 Example Waveforms (Resolver)

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

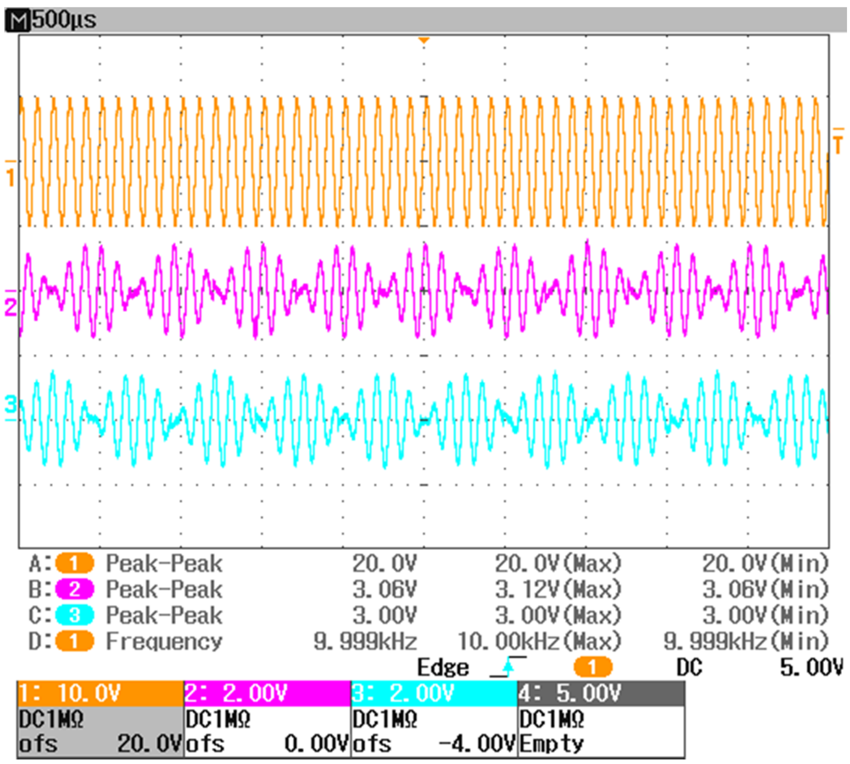


Figure 6: Input and output signals at 10000 rpm

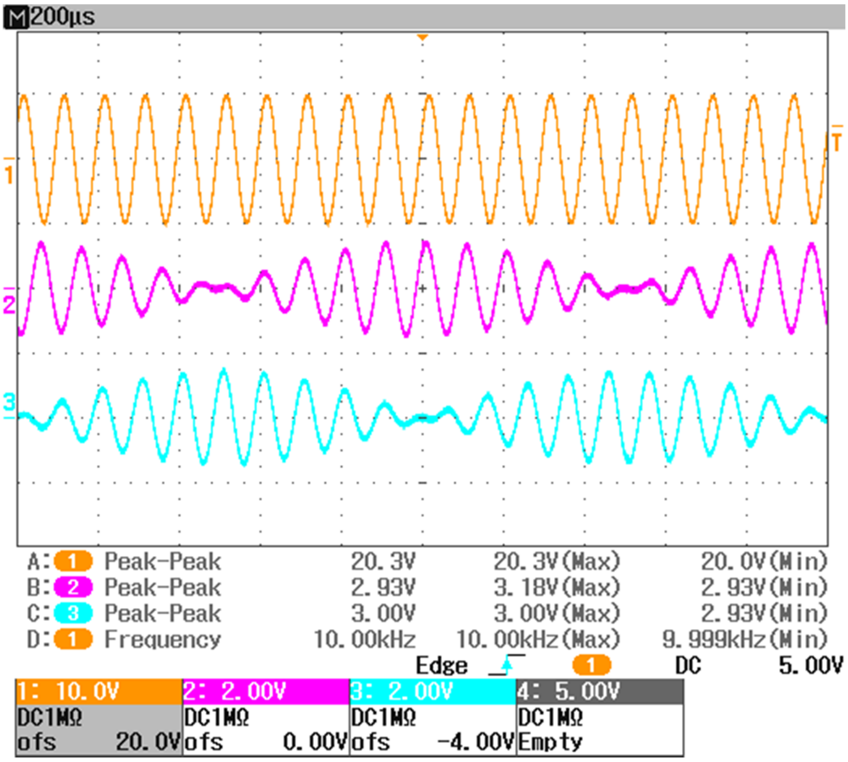


Figure 7: Input and output signals at 4000 rpm

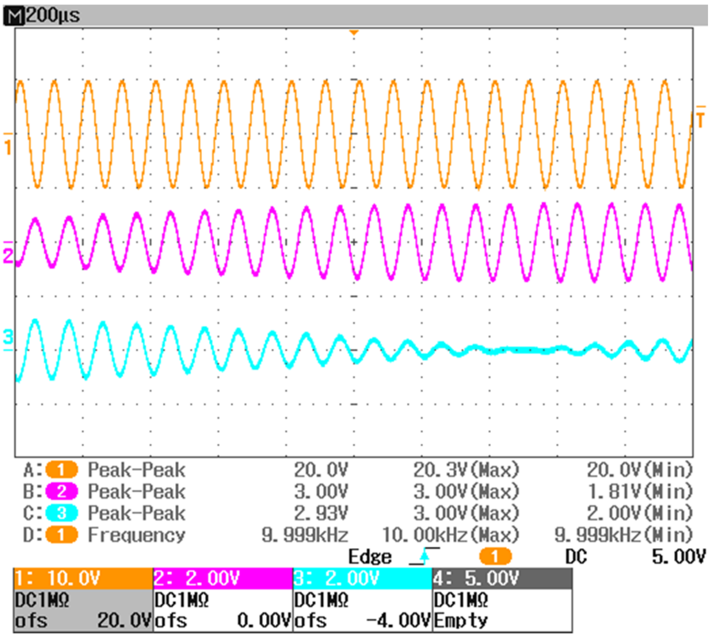


Figure 8: Input and output signals at 1000 rpm

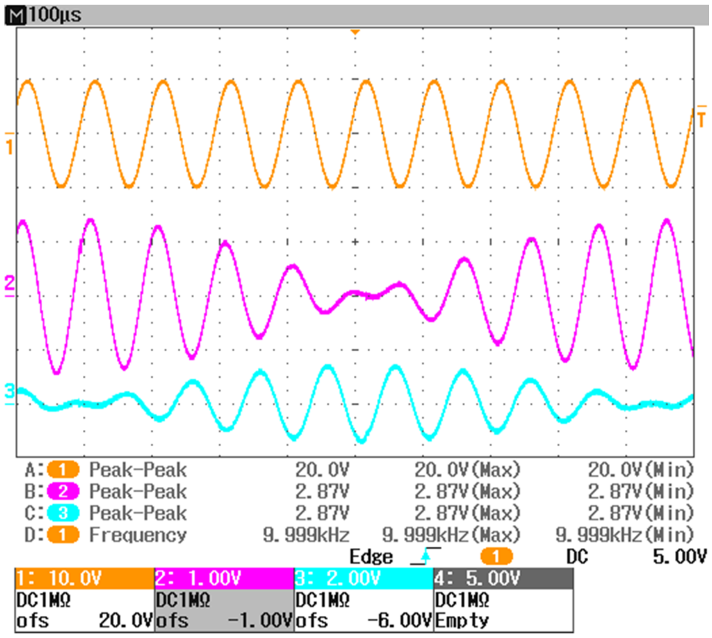


Figure 9: Details on switching point

3.6 Example Waveforms (RVDT)

The next screenshots illustrate the functionality (channel 3: excitation; channel M: differential output signal; max. angle (setup): 30°).

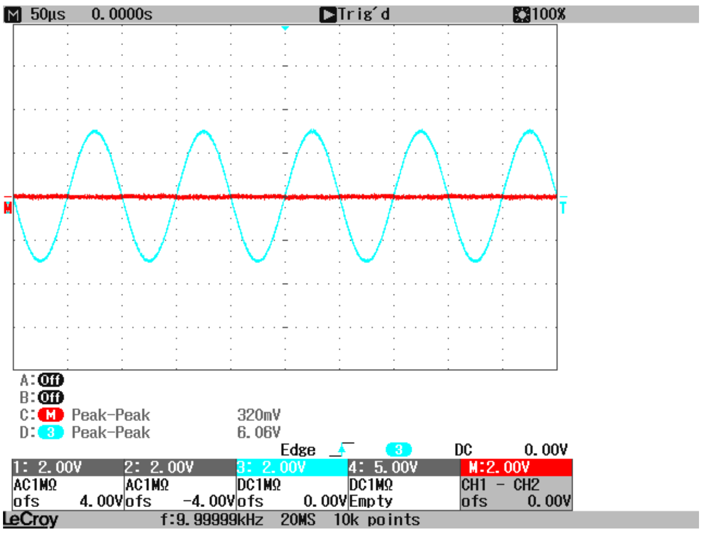


Figure 10: RVDT input and differential output signal at 0°

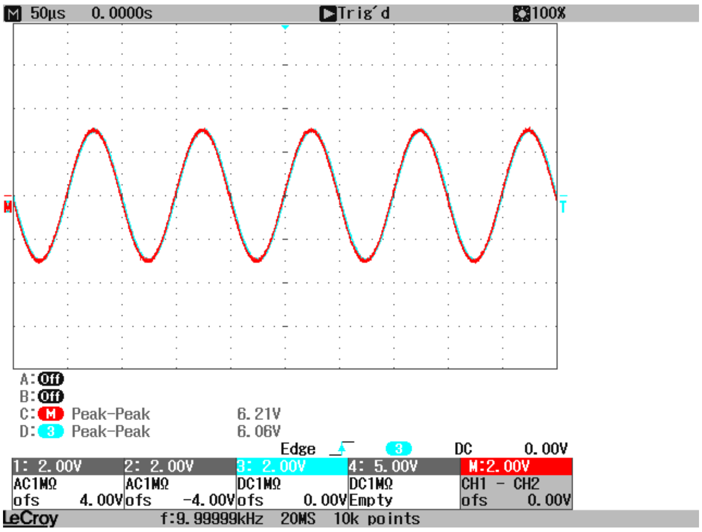


Figure 11: RVDT input and differential output signal at +30° (max.)

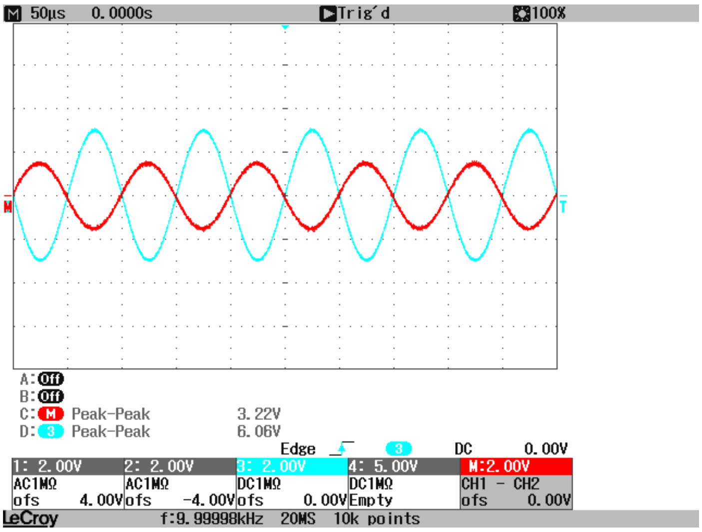


Figure 12: RVDT input and differential output signal at -15°

4. Software

4.1 Labview Project

4.1.1 General

The project contains the complete software for the target:

- FPGA
- Real Time System: RT_ResSi & RT_RVDT
- Computer: PC_ResSi & PC_RVDT

The targets cRIO-9074, cRIO-9075 and sbRIO-9602 are already predefined and precompiled for one channel, but it's easy to add further targets and/or channels to the project.

Note 1: The PC software is for demonstration purposes only.

Note 2: The RT system is not necessary required but recommended. The FPGA can also be directly controlled by a computer, but this manual will only handle the recommended way (FPGA ←→ RT ←→ Network Shared Variables ←→ PC).

4.1.2 Adding a new RIO target

Adding a new RIO target is very simple. Following a short description:

1. Create a new target device
    - Right click on project → New → Targets and Devices…
    - Select the target
    - The new target is appears in project explorer
2. Copy required elements from old targets to the same location in the new target
    - Folders: RT, RT_ResSi, RT_RVDT
    - Add the FPGA target (Right click on new Chassis → New → FPGA Target)
    - Copy FPGA Elements: "[0]_ResSi" (Folder & Device), "FPGA", "Memory", "SPI_Resolver_Clock"
3. Create new build specifications for FPGA, RT_ResSi and RT_RVDT by using the old ones as templates (modify just the "Name" [1], "Target Directory" [2] and update the "Source Files" [3])
4. Build the FPGA build specification and wait until finish (takes some time) → Bit File
5. Add the new bit file to "RT_ResSi/RT_OpenFPGA.vi":
    - Duplicate Subdiagramm
    - Set the Symbol to "DeviceCode" and the value to the target device code, then hit "OK" (to get the device code right click on new target → Properties → Conditional Disable Symbols → DeviceCode)
    - Configure the "Open FPGA VI Reference" to point to the new generated FPGA bit file
6. Build the required RT build specification ("Run as startup")

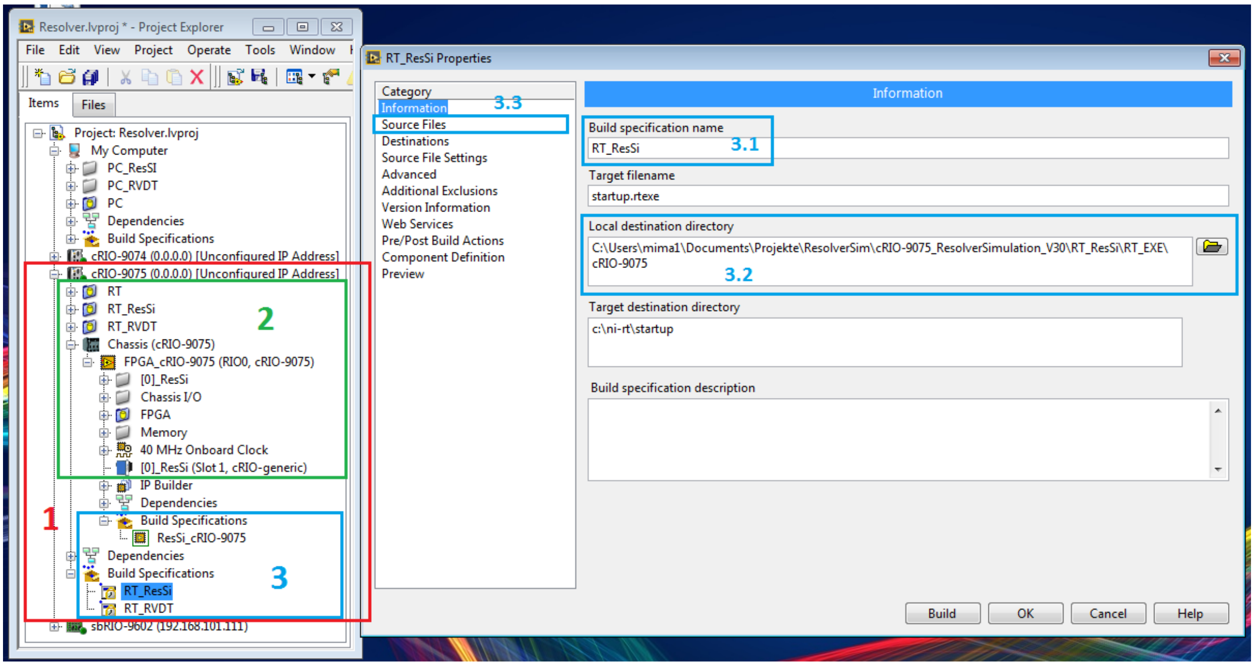


Figure 13: Add a new RIO target (Step: 1-3)

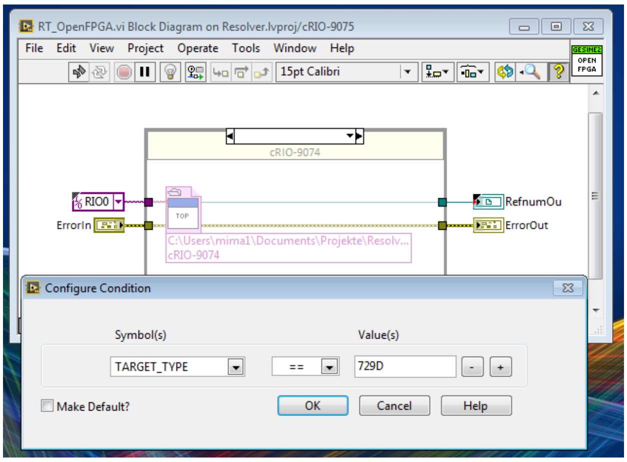


Figure 14: Add a new RIO target (Step: 5)

4.2 FPGA

This part handles the code for the FPGA. It’s setup for one device on slot 1 (channel 0) and is already prepared for extending with further channels.

4.2.1 Top VI & FPGA Interface

The top vi includes all input and output variables which are needed for operation.

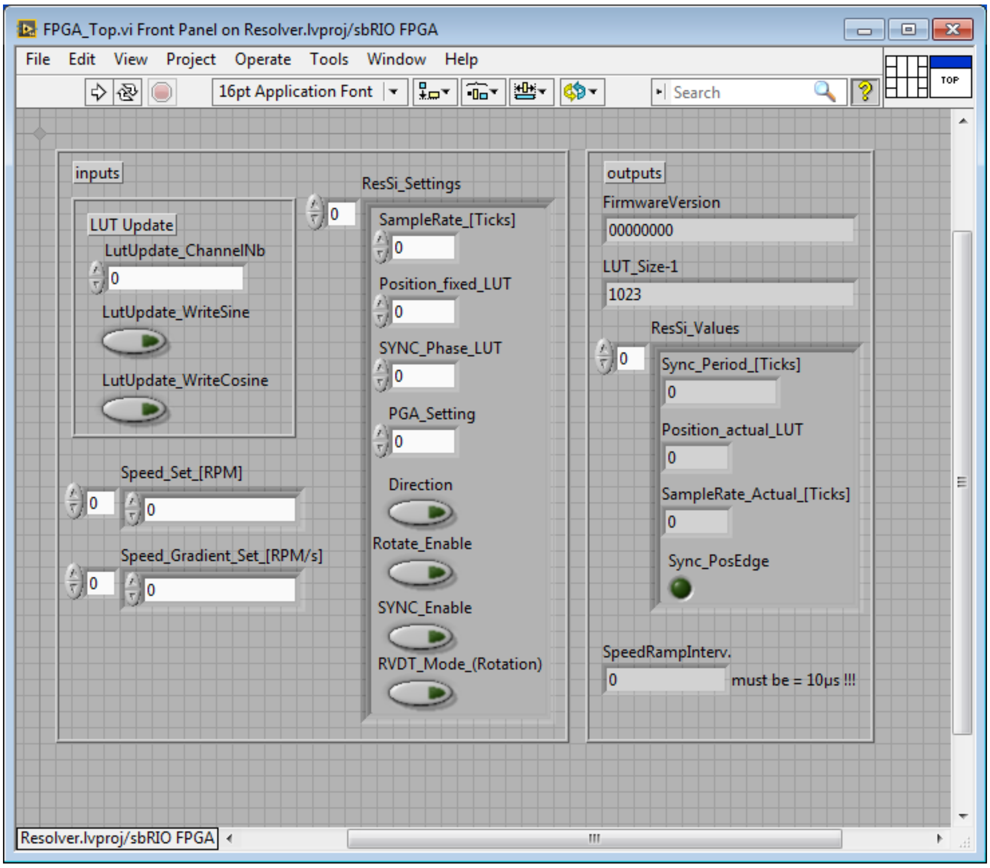


Figure 15: FPGA Top VI

Inputs:

- LUT Update Section (see: 4.2.2)
  Controls for updating the sine and cosine lookup tables. Use DMA_FIFO for writing new values.

Parameter	Description
LutUpdate_ChannelNb	Channel number for LUT write (only channel 0 is predefined)
LutUpdate_WriteSine	Set to start writing to sine LUT (DMA_FIFO).
LutUpdate_WriteCosine	Set to start writing to cosine LUT (DMA_FIFO).

- ResSi_Settings[x], Speed_Set_[RPM][x], Speed_Gradient_Set_[RPM/s][x]
  Resolver simulation control values (Arrays where index **x** represents the channels number).

Parameter	Description
SampleRate_[Ticks]	**ResSi Mode** Update rate of the output samples to define the rotating speed. When **RPM** is the desired speed in rounds per minute: $SampeRate_{[Ticks]} = \frac{40^6 * 60}{RPM * 1024}$ **RVDT Mode** Angle changing speed rate. When $\left[\frac{\Delta\varphi}{sec}\right]$ is the desired angle changing rate per second: $SampleRate_{[Ticks]} = \frac{\lvert\varphi_{max}\rvert * 40^6}{\left[\frac{\Delta\varphi}{sec}\right] * (1023 \gg 1)}$
Position_fixed_LUT	Sets a constant position of the simulation. The value is in the range 0…1023. **ResSi Mode** $Range = 0° \dots 360°$ **RVDT Mode** $Range = -\varphi_{max} \dots 0° \dots \varphi_{max}$ (0° → Position = 1023>>1 = 511)
SYNC_Phase_LUT	**ResSi Mode** 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
PGA_Setting	Maximum transformation ratio when sine/cosine LUT is defined to full scale (Amplitude 100%). $\hat{U}_{OUT} = 0.078125 * 2^{\boldsymbol{PGA}} * \hat{U}_{LUT}\ where\ PGA = 0 \dots 7$ **Set this value to "2" for hardware version 1.3!**
Direction	**ResSi Mode** Sets sense of rotation FALSE: Left TRUE: Right
Rotate_Enable	**ResSi Mode** TRUE: a rotating machine is simulated. FALSE: a fixed position is simulated
SYNC_Enable	**ResSi Mode** TRUE: rotating speed derived from the SYNC input frequency FALSE: rotating speed derived from the Sample_Rate[Ticks] input
RVDT_Mode_ (Rotation)	TRUE: RVDT mode FALSE: Resolver simulation mode (default)
Speed_Set_[RPM]	**ResSi Mode** Speed ramp target speed in RPM
Speed_Gradient_Set _[RPM/s]	**ResSi Mode** Speed change rate in RPM per second, take new speed immediately, when 0. Otherwise 1 RPM/s up to 16e6 RPM/s

Outputs:

- LUT_Size-1
  Constant: The value represents the last index of the sinus and cosine lookup tables.
- FirmwareVersion
  Constant: FPGA Firmware versions date (HEX-Format: yyyymmdd)
- ResSi_Values[x]
  Resolver simulation status values (Array where index **x** represents the channels number).

Parameter	Description
Sync_Period_[Ticks]	**ResSi Mode** Number of ticks between two positive edges on SYNC input (Ticks of 25ns)
Position_actual_LUT	Actual position of the simulation (see: “Position_fixed_LUT”).
SampleRate_Actual_[Ticks]	**ResSi Mode** Current update rate of the output samples defining the rotating speed.
Sync_PosEdge	**ResSi Mode** Sync external speed signal to positive edge.

4.2.2 Sine & Cosine Lookup Tables

The output waveforms are saved in two lookup tables with 1024 16-bit signed integer elements each. One LUT is responsible for the sine output and the second one for the cosine output. These tables can (should) be update by the user. For this purpose there are three controls and one DMA available.

For updating, the developer can use “RT_PC_LUT_Write.vi” located in “RT_Share”. It does the following steps:

- Write the desired channel number to “LutUpdate_ChannelNb”. In default state there is just one channel available.
- Set “LutUpdate_WriteSine” or “LutUpdate_WriteCosine” to true to initiate a new transfer.
- Write 1024 values to the DMA “DMA_FIFO” and wait some milliseconds after.
- Reset the binary write value to false.

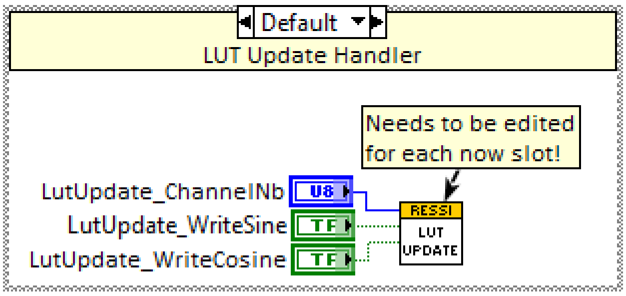


Figure 16: FPGA_Top.vi - LUT Update

4.2.3 Adding a new channel

- Create a new slot
  - Right click on the FPGA target → New → C Series Module → New target or device → C Series Module
  - Name: “[x]_ResSi” where x = channel number (in this example x = 1)

- Type: cRIO-generic

  Remark: If this option is not available, the following line must be added to the LabVIEW.ini file (LabVIEW must be closed):

  cRIO_FavoriteBrand=generic

- Location: Slot number

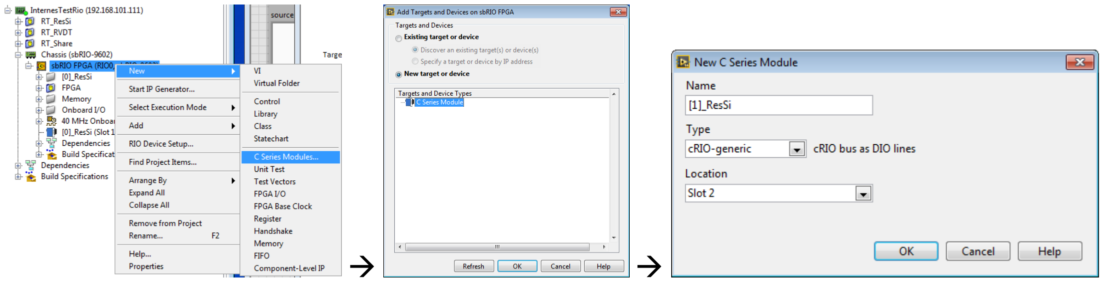


Figure 17: Creat a new channel on FPGA

- Create a copy of all memories except DMA_FIFO (Change the name of each memory to represent the new channel number, in this case 1).

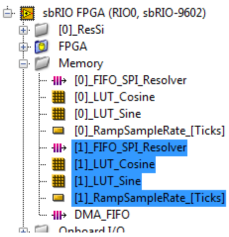


Figure 18: Copy FPGA memories

- The following changes needs to be made:
  - FPGA_Top.vi → SPI Handlers
    Duplicate frame and set the new port pins (“[1]_ResSi/...” in this case).
  - FPGA_Top.vi → Speed Ramp Generator
    - Increase array size of the two input variables by one (in this case to 2):
      Right click on variable → Properties → Size → Fixed
    - Duplicate frame, set the new memory (“[1]_RampSampleRate_[Ticks]” in this case) and connect the vi to the next array elements (by extending the “Index Array” elements)
  - FPGA_LutUpdate.vi
    Duplicate case element and update the memories (“[1]_LUT_Sine” and “[1]_LUT_Cosine” in this case).
  - FPGA_ResolverSim.vi
    - Increment the size of “ResSi-Settings” by one (in this case to 2) and repeat this step also on the top vi.
    - Duplicate frame and update all constants (“[1]_...” in this case).
    - Connected the input and output variable to the new array entry.
- Rebuild the FPGA target
- If you use the real time system, you need to create a copy of the network variables for the new channel. For further information’s see chapter 0.

4.3 Real Time System

The software for the real time system controls the device according to the values it gets from network interface (Shared Network Variables).

Note: It is also possible to control the FPGA without using the real time system. The user may design his own application.

4.3.1 Network Interface (Shared Network Variables)

To control the simulation the real time system is communicating with the host application by using a technique called "Shared Network Variables". This are containers of variables which can be accessed by a special network address. The containers are located in the RT folders ("RT_ResSi" or "RT_RVDT"):

- **ResSi_Variables.lvlib / RVDT_Variables.lvlib**
  Contains non channel related variables like FPGA version
- **ResSi_Variables_x.lvlib / RVDT_Variables_x.lvlib**
  Contains channel related variables where x represents the channel number (each channel will have its own file).

4.3.1.1 Access a variable from host computer

To access a variable from a remote system, the developer can use the VI "RT_PC_ResSi_NetVarPathGen.vi" ("RT_PC_RVDT_NetVarPathGen.vi" respectively) located in "RT_ResSi/Share" ("RT_RVDT/Share" respectively) together with a "Read Variable" or "Write Variable" control.

PC example for reading the actual angle and for writing a new angle in RVDT simulator mode:

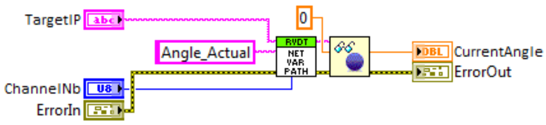


Figure 19: Example for reading a network variable (PC)

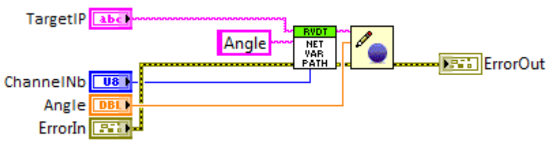


Figure 20: Example for writing a network variable (PC)

- TargetIP: RIO ip address or hostname
- Angle_Actual / Angle: Name of the desired variable in "RVDT_Variables_x.lvlib"
- ChannelNb: Channel number → x (if x = 255, access goes to "RVDT_Variables.lvlib")

4.3.1.2 Adding a new channel (example for RVDT channel 1):

1. If not done, add the new channel to the FPGA code (see 4.2.3)
2. Create copy of “ResSi_Variables_0.lvlib” and rename to “ResSi_Variables_1.lvlib” (this step cannot be done in project explore, instead use windows explorer)
3. Update FPGA reference
   - Open “RT_RVDT/RT_RVDT_Thread.vi”
   - Right click on “RefnumIn” and select “Open Type Def.”
   - Right click on “Reference” and chose “Configure FPGA VI Reference…”
   - Click on “Import from bitfile…”, browse to the required bit file located in “FPGA Bitfiles” and open it
   - Save everything
4. Update build specification and rebuild
   - Right click on “RIO-System/Build_Specification/RT_RVDT” → Properties
   - Select “Source Files” on right side
   - Browse to “RT_RVDT/RVDT_Variables_1.lvlib” and add it to section “Always Included”
   - Rebuild and flash (“Run as startup”)

4.3.2 Resolver Simulation (RT_ResSi)

Placeholder → Use version 2.2 instead!

4.3.3 RVDT Simulation (RT_RVDT)

This section describe the real time system for RVDT simulation mode.

4.3.3.1 Control Interface (RVDT_Variables)

Like told in section 4.3.1 the RVDT simulation will be controlled by network shared variables. The following table describe them:

Variable	Type	Access	Description	Unit
FPGA_Version (common)	UInt32	r	HEX coded FPGA compilation date: Format: 0xYYYYMMDD	
Angle	Double	r/w	Simulation angle in degree	°
Angle_Actual	Double	r	Current simulation angle in degree	°
Angle_DegreePerSec	Double	r/w	Moving speed to new angle position	$\frac{\Delta°}{sec}$
Angle_Max	Double	r/w	Maximum angle swing (absolute value): Angle where the difference output signal is at his maximum amplitude set up by "LUT_Amplitude" and PGA.	°
LUT_Amplitude	Double	r/w	Maximum amplitude of the differential output signal relative to the excitation signal amplitude multiplied with PGA (see "LUT_Write").	%
LUT_Calibration	CTL[1]	r/w	Calibration settings (see "LUT_Write")	
LUT_Write	Boolean	r/w	Set to update the lookup tables with new data calculated with the values in "LUT_Amplitude" and "LUT_Calibration". True: Start update process (set by user) False: Update process finished (set by RT)	
PGA	UInt8	r/w	PGA setting (Range: 0…7): $\hat{U}_{OUT} = 0.078125 * 2^{\boldsymbol{PGA}} * \hat{U}_{LUT}$	
SaveConfig	Boolean	r/w	Set to permanently save current configuration and angle setup in real time system flash memory (it will be loaded on every start up). True: Start saving process (set by user) False: Saving process finished (set by RT)	

CTL[1] → RT_PC_RVDT_LUT_Calibration.ctl

Variable	Type		Description	Unit
SineAmplitude_% CosineAmplitude_%	Double		Amplitude correction factors for lookup table (see "LUT_Amplitude") $Amplitude_{LUT} = \text{"}LUT\_Amplitude\text{"} * x$	%
SineDCOffset_% CosineDCOffset_%	Double		LUT DC offset correction amplitude relative to the excitation signal amplitude multiplied with PGA $Amplitude_{LUT} = \text{"LUT\_Amplitude"} + x$	%

4.3.3.2 RT_RVDT_Top.vi

Top vi of the real time system in RVDT simulation mode.

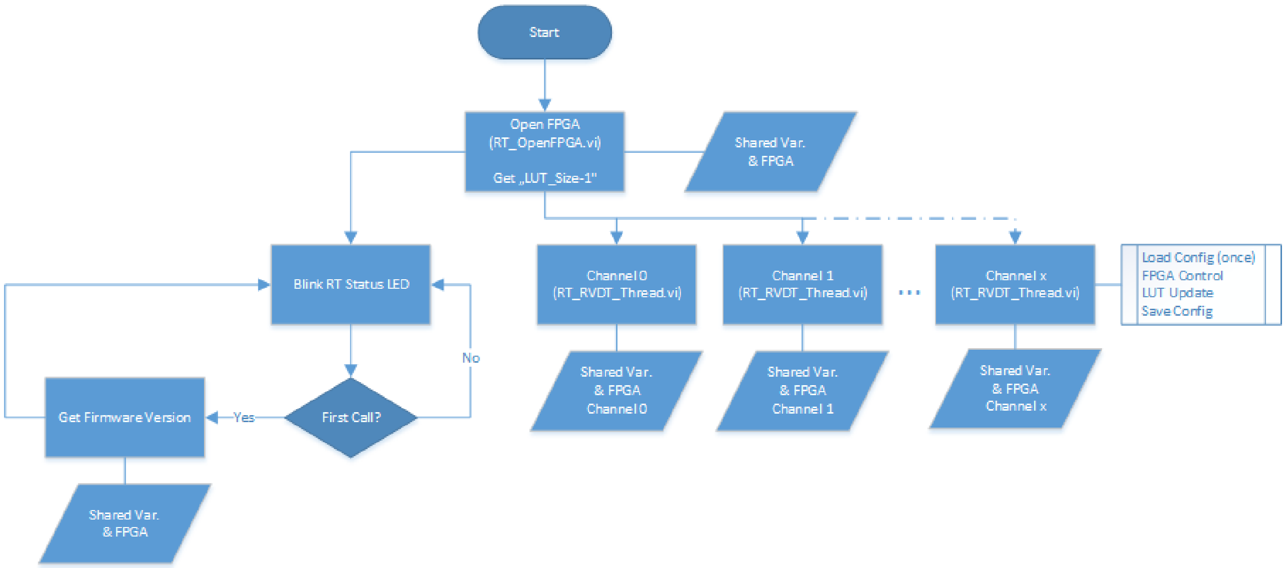
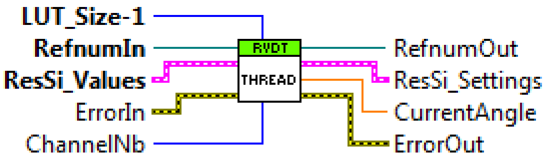


Figure 21: RT_RVDT_Top.vi - Flow Chart

4.3.3.3 RT_RVDT_Thread.vi

This vi is responsible for controlling the FPGA. It is called cyclic (100ms) from top vi for each available channel on FPGA (parallel as multi thread).



- LUT_Size-1: Last index of the lookup tables (get it from FPGA)
- ChannelNb: Channel number
- ResSi_Values: Connect to FPGA “ResSi_Values” array element at index defined by “ChannelNb”
- ResSi_Settings: Connect to FPGA “ResSi_Settings” array element at index defined by “ChannelNb”

4.3.3.4 RT_RVDT_SaveCfg.vi & RT_RVDT_LoadCfg.vi

This are used for saving the current configuration to flash memory and for loading it again.

4.3.3.5 RT_PC_RVDT_LUT_ValueWrite.vi

With this vi the update date for the lookup tables will be created. It takes the amplitude (LUT_Amplitude) and the calibration settings (LUT_Calibration) as input parameters and generates the lookup table data for both tables. The following diagram shows example data (LUT-Size = 1024; Max. Angle = +-30°; LUT_Amplitude = 100%):

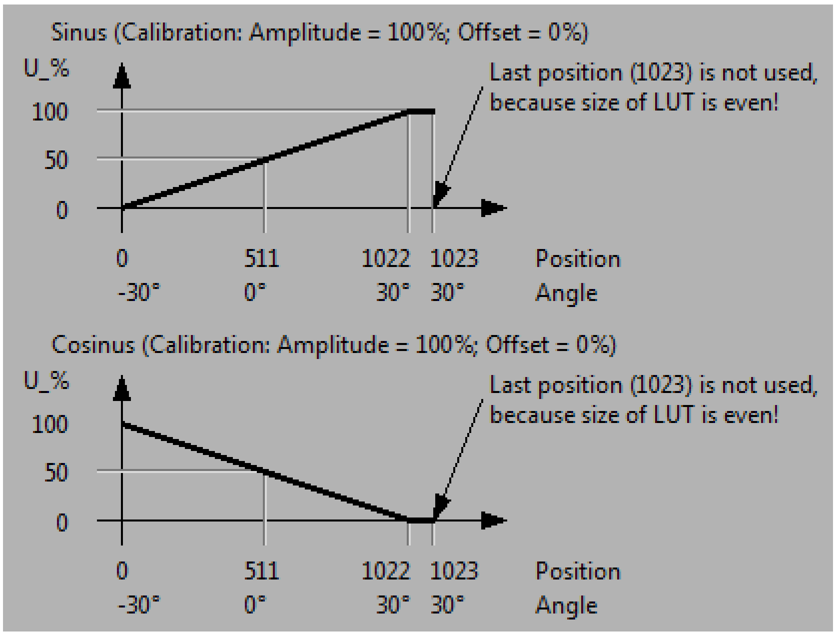


Figure 22: RVDT LUT Example Data Diagram

4.4 PC

The Software for PC is mostly for demonstrations and test purposes only, however many of its components (vi's) can be used also in own software implementations. This makes the developing of costumer applications faster and easier.

Note: The PC software included in this project is not suitable for direct FPGA control without using the real time system!

4.4.1 Execution of demo software

To start the software there are two possibilities available. The first approach is to open the desired top vi ("PC_ResSi_Top.vi" or "PC_RVDT_Top.vi") and push the run button. The second possibility is to build the predefined build specification (if not already done) and then to start the compiled executable located in sub folder "PC_EXE". The advantage of second method is that it needs just the run time installed on the target machine.

4.4.2 Resolver Simulation

See documentation version 2.2.

4.4.3 RVDT Simulation

4.4.3.1 Demo & Test GUI

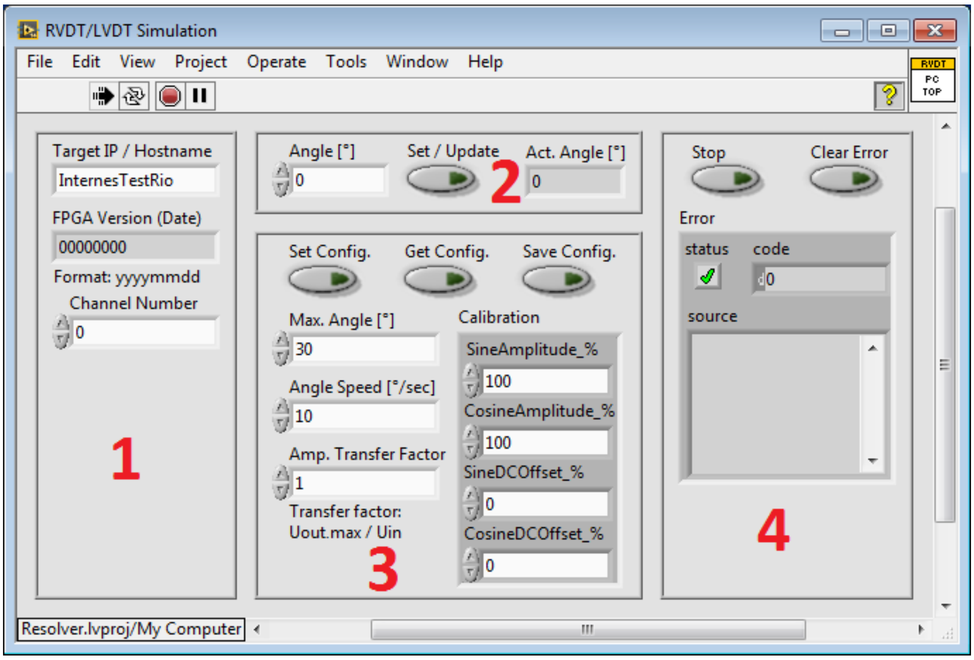


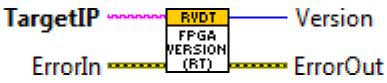
Figure 23: PC_RVDT_Top.vi – GUI

1. Basic configuration
   - Target IP / Hostname → RIO target address
   - FPGA Versions (Date) → FPGA compile date (Format: YYYYMMDD)
   - Channel Number → Current selected channel to control
2. Angled Control
   - Angle [°] → Desired simulation angle in degree
   - Set / Update → Start moving to position defined by "Angle [°]"
   - Act. Angle [°] → Actual simulation position in degree
3. Device Setup
   - Set Config. → Write configuration to target
   - Get Config. → Read current configuration from target
   - Save Config. → Save current configuration permanently to target flash memory
   - Max. Angle [°] → Maximum angle swing (see 0)
   - Angle Speed [°/sec] → Moving speed to new angle position (see 0)
   - Amp. Transger Factor → Transfer factor between excitation signal and differential output signal
   - Calibration → Calibration settings (see 0 > RT_PC_RVDT_LUT_Calibration.ctl)
4. Application Control

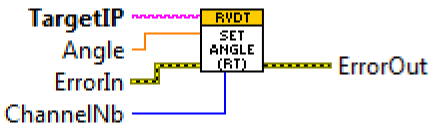
4.4.3.2 VIs for integration into user application

There are some vi's for communicating with the real time system. This vi's can be used in any user application:

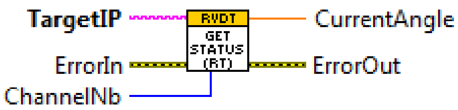
- PC_RVDT_FPGAVersion.vi → Read the FPGA compilation date



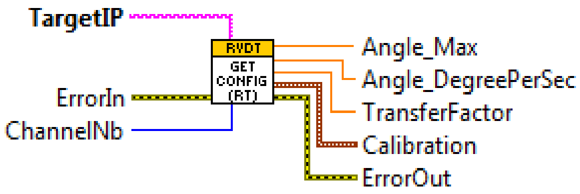
- PC_RVDT_SetAngle.vi → Set new simulation angle



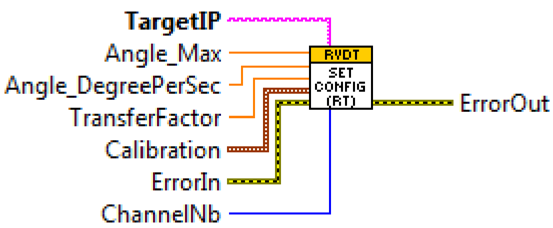
- PC_RVDT_GetStatus.vi → Read current status (actual angle)



- PC_RVDT_GetConfig.vi → Get current configuration



- PC_RVDT_SetConfig.vi → Set a new configuration



- PC_RVDT_SaveConfig.vi -> Save configuration permanently to flash memory (boot config)