Software Information

REGATRON programmable power supplies

Programming Overview

TCP GSS

Manual and software-based control of TopCon devices – Introduction and General Overview

Regatron is manufacturer of the well-known TopCon Power Supply units (PSU). Controlling the TopCon PSU can be done with direct user interaction or by using the various programming interface options.
In particular it is offered

Manual control by user interaction:

- by the Human Machine Interface (HMI)
- by the Remote Control Unit RCU (HMI in external housing)
- by the PC-Software TopControl

Besides these user interfaces there exists a set of ways to control the TopCon PSU from other software:

All following APIs are available on our website www.regatron.com

- **By using the Low-Level Protocol (LLP)**
  This is the basic protocol to communicate with the TopCon PSU and is commonly used by the "higher" programming APIs. In rare cases (when using a PLC and thus not being able to include DLLs in the system) its use is indicated, too.
  Short messages (5-9 Bytes) are exchanged in a request / response manner. To use the LLP the programmer needs the addresses of the correct internal registers. The most important ones can be found in the LLP documentation Low-Level Protocol for TopCon Device / TFE / SAS that can be downloaded from the Regatron.com homepage.

  In general it is not recommended to use this protocol if any other option can be used, as it poses some severe drawbacks due to its basic functionality. There exist better alternatives that allow for a proper abstraction to the needed functionality.

- **By using the TopCon Input-Output DLL TCIO.dll**
  This is a high-level DLL written in C/C++ that offers a comprehensive, covering set of functions for the different TopCon PSU versions. Examples of the commands in this DLL are:

  int TC4GetVoltageAct (double *p_vact)
  or
  int TC4SetControlIn (unsigned int voltage_on)

  that return the actual voltage of the TopCon (Quadro) and allows to switch the PSU on or off resp. The int return value indicates if the command was executed successfully.

  It should be understood that the command TCGetVoltageAct (…) (! TC.. , not TC4...) that is included in the tcio.dll, too, implements the getVoltageAct command for older TopCon systems (pre Quadro).

  As summary it can be said that the tcio.dll is covering all commands available in TopControl (actually: TopControl is based on the tcio). It is a set of commands wellbalanced between convenience and performance.

Regatron AG
Feldmühlestrasse 50
9400 Rorschach
Switzerland
E-Mail: tc.info@regatron.ch
Phone: +41 71 846 67

- **By using the .NET Progamming API**
  The .NET programming API is another way to control the TopCon PSU. It implements a true object-oriented high-level access to the TopCon devices. It abstracts from the particular device and offers a well-documented set of functions to very conveniently program the PSU.
  The following commented C# example explicates the simplicity of using these commands:

```
// create a new TopCon device and connect the PC with it via COM5
// switch on the PSU
// output 4 times the actual voltage
// switch power off and close the connection
DEV.TopCon tc = new DEV.TopCon(
new DEV.TopConConfiguration_Dummy(), //-- create a Dummy configuration
TopCon.Broker.Device.DEVICE_1); //-- identify this as device1
tc.Connect(5);
tc.SetPowerON();
for (int i = 0; i < 4; i++)
{
Console.WriteLine(" Voltage now: ["+ tc.GetVoltageAct()+"Volt ]");
Console.ReadKey();
}
tc.SetPowerOFF();
tc.Disconnect();
```

- This example shows the high level of abstraction of the particular device. After having installed the connection to the PSU, general commands are available for accessing the PSU. This concept holds true for further TopCon generations thus reducing the burden of adapting your existing software to new TopCon models.

  The importance of abstraction gets obvious especially for the various curves being used in the TopCon Function Engine (TFE) and Solar Array Simulation (SAS) domain. Here in-depth knowledge of the underlying C structures would be needed in all other programming approaches (like e.g. tcio.dll). In LLP programming own curves on the TopCon is hardly possible.

  The following example shows how easy it is to create a sinus voltage curve in the TopCon TFE. After creation of the particular voltage controller component (sinus), a TFECurve is generated with the necessary voltage controller directions. Afterwards a curve transport object (curveContainer) is created from the curve object and this container is sent to the TopCon.

  Creating a solar array simulation curve (calculated- or customSASCurve) or loading a battery/supercap model Energa Storage Simulation follows a likewise simple approach. This significantly enhances the productivity of the software programmer and helps to avoid programming errors.

```
//------------ creating the sinusodial CurveComponent
TFE.SineCurveComponent sineCC_1 = new TFE.SineCurveComponent(160.0, 5.00, 20.0);
//--------- create a new curve with number, name, voltage controller component (here: our sinus),
//--------- no current controller, no power controller
//--------- component and generalEnable set to true
myNewCurve = TFE.CurveFactory.createTFECurveFromTimeBasedCurveComponents(
987,
"curve987_simple_Sinus",
sineCC_1,
null,
null,
true);
//------------ create a new “CurveContainer” transport object ------------------
myCC = new TFE.CurveContainer(tc.getBroker());
//------------ convert data from the abstract curve object structure to curveContainer ----------
myCC.UpdateFromTFECurve(myNewCurve);
//----------- send the CurveContainer to TopCon
myFE.StoreCurveContainerToTopConFlash(myCC);
```

- **By using predefined LabVIEW VIs.**
  The delivered VI can be used to establish a connection to the device, readout actual values, set configuration parameters and, of course, reference values.
  The VIs (TopCon API.lvlib) can directly be used in the LabVIEW environment.
  An overview, how the VI looks like.