Remote Control Programming

This section explains how to remote-control the BP series from the personal computer through the USB interface.

This manual is described as an example the BP4610. Please read according to the your model

This section assumes an understanding of the VISA (Virtual Instrument System Architecture) programming interface and programming language supported by the VISA library and using USBTMC driver.

VISA library	: National Instruments' NI-VISA and .net Framework 2.0 Language Support.
Programming Environment	: Microsoft's .net 2.0 or higher with C# language For other programming environments, see the respective reference.

This section describes:

1. **Continuous output setting**
   Make the output setting, then turn on the output. After that, use the measurement function to read the measured output value.

2. **Sequence function control**
   Read the sequence data pre-stored in the main unit, and start the sequence execution. After that, use the measurement function to read the measured output value.

3. **Obtain status register**
   After setting the warning status transition filter and the warning event enable register, obtain the status byte to check for the warning state.

These explanations are intended for helping you understand the remote command control procedure, and thus omit general programming considerations such as error detection processing and variable initialization processing.

For the VISA library installation, see the document from the VISA library distributors.

1. Continuous Output Setting

```
using System.Windows.Forms;

// <Process flow>
// ■Initialize the communication state.
// NameSpace declaration of NI-VISA library
using NationalInstruments.VisaNS;

// Search for BP4610 using the resource string
string[] strResNames;

// Generate the VISA session of the NI-VISA library
ResourceManager rm;
rm = ResourceManager.GetLocalManager();

// Use the VISA session to specify the BP4610 serial number and
// establish the communication session with the BP4610
//
// In your program, write the exception handling.
// Use catch to write the error handling when communication session is established.
// Whenever an error occurs, NI-VISA library throws it to the exception.
//
// If the bipolar DC power supply is "BP4610", "USB0::0x0D4A::0x0040::?*::INSTR"
// If the bipolar DC power supply is "BP4620", "USB0::0x0D4A::0x0041::?*::INSTR"
strResNames = rm.FindResources("USB0::0x0D4A::0x0040::?*::INSTR");
MessageBasedSession bp;
bp = new MessageBasedSession(strResNames[0]);

// Set to the Remote State
new UsbSession(bp.ResourceName).ControlRen(RenMode.Assert);

// Clear the device
bp.Clear();

// Clear the error status
bp.Write("*CLS");
System.Threading.Thread.Sleep(1000);

// Return the BP4610 to the initial state just after the startup.
// Return the output setting to the state just after the startup.
bp.Write("*RST");
System.Threading.Thread.Sleep(1000);
```

```
// ■Set the operation mode.
bp.Write("MODE 0");
System.Threading.Thread.Sleep(1000);

// ■Set the DC voltage.
bp.Write("VOLT 10.00");
System.Threading.Thread.Sleep(1000);

// ■Set the superimposed AC voltage.
bp.Write("ACVL 100.0");
System.Threading.Thread.Sleep(1000);

// ■Set the superimposed AC frequency.
bp.Write("FREQ 50.0");
System.Threading.Thread.Sleep(1000);

// ■Set the superimposed AC waveform.
bp.Write("WAVE 0");
System.Threading.Thread.Sleep(1000);

// ■Turn on the output.
bp.Write("OUTP 1");
System.Threading.Thread.Sleep(1000);

// ■Obtain the measured output voltage value.
string strMeasureDCVoltage = "";
string strMeasureACVoltage = "";
strMeasureDCVoltage = bp.Query("MVLT?");
strMeasureACVoltage = bp.Query("MACV?");

// ■Obtain the measured output current value.
string strMeasureDCCurrent = "";
string strMeasureACCurrent = "";
strMeasureDCCurrent = bp.Query("MCUR?");
strMeasureACCurrent = bp.Query("MACC?");

// ■Turn off the output.
bp.Write(“OUTP 0");
System.Threading.Thread.Sleep(1000);

// Clear the Remote state
new UsbSession(bp.ResourceName).ControlRen(RenMode.Deassert);

// ■Terminate the communication and release the session.
bp.Terminate();
bp.Dispose();
```

2. Sequence Function Control

```
using System.Windows.Forms;

// <Process flow>
// ■Initialize the communication state.
// NameSpace declaration of NI-VISA library
using NationalInstruments.VisaNS;

// Search for BP4610 using the resource string
string[] strResNames;

// Generate the VISA session of the NI-VISA library
ResourceManager rm;
rm = ResourceManager.GetLocalManager();

// Use the VISA session to specify the BP4610 serial number and
// establish the communication session with the BP4610
//
// In your program, write the exception handling.
// Use catch to write the error handling when communication session is established.
// Whenever an error occurs, NI-VISA library throws it to the exception.
 //
// If the bipolar DC power supply is “BP4610”, "USB0::0x0D4A::0x0040::?*::INSTR"
// If the bipolar DC power supply is “BP4620”, "USB0::0x0D4A::0x0041::?*::INSTR"
strResNames = rm.FindResources("USB0::0x0D4A::0x0040::?*::INSTR");
MessageBasedSession bp;
bp = new MessageBasedSession(strResNames[0]);

// Set to the Remote State
new UsbSession(bp.ResourceName).ControlRen(RenMode.Assert);

// Clear the device
bp.Clear();

// Clear the error status
bp.Write("*CLS");
System.Threading.Thread.Sleep(1000);

// ■Turn on the output.
bp.Write("OUTP 1");
System.Threading.Thread.Sleep(1000);
```

```
// ■Start the sequence.
bp.Write("SCTL 1");
System.Threading.Thread.Sleep(1000);

// ■Obtain the measured output voltage value.
string strMeasureDCVoltage = "";
string strMeasureACVoltage = "";
strMeasureDCVoltage = bp.Query("MVLT?");
strMeasureACVoltage = bp.Query("MACV?");

// ■Obtain the measured output current value.
string strMeasureDCCurrent = "";
string strMeasureACCurrent = "";
strMeasureDCCurrent = bp.Query("MCUR?");
strMeasureACCurrent = bp.Query("MACC?");

// ■Turn off the output.
bp.Write("OUTP 0");
System.Threading.Thread.Sleep(1000);

// Clear the Remote state
new UsbSession(bp.ResourceName).ControlRen(RenMode.Deassert);

// ■Terminate the communication and release the session.
bp.Terminate();
bp.Dispose();
```

3. Obtain Status Register

```
using System.Windows.Forms;

// <Process flow>
// ■Initialize the communication state.
// NameSpace declaration of NI-VISA library
using NationalInstruments.VisaNS;

// Search for BP4610 using the resource string
string[] strResNames;

// Generate the VISA session of the NI-VISA library
ResourceManager rm;
rm = ResourceManager.GetLocalManager();

// Use the VISA session to specify the BP4610 serial number and
// establish the communication session with the BP4610
//
// In your program, write the exception handling.
// Use catch to write the error handling when communication session is established.
// Whenever an error occurs, NI-VISA library throws it to the exception.
//
// If the bipolar DC power supply is “BP4610”, "USB0::0x0D4A::0x0040::?*::INSTR"
// If the bipolar DC power supply is “BP4620”, "USB0::0x0D4A::0x0041::?*::INSTR"
strResNames = rm.FindResources("USB0::0x0D4A::0x0040::?*::INSTR");
MessageBasedSession bp;
bp = new MessageBasedSession(strResNames[0]);

// Set to the Remote State
new UsbSession(bp.ResourceName).ControlRen(RenMode.Assert);

// Clear the device
bp.Clear();

// Clear the error status
bp.Write("*CLS");
System.Threading.Thread.Sleep(1000);

// Set permission of the warning event register
bp.Write("WREE 1");
System.Threading.Thread.Sleep(1000);
```

```
// ■Obtain the status byte to check if the warning state is detected

// ReadStatusByte() needs to be repeatedly queried in order to detect the change
// in each bit of the register.
// In actual programming, the status byte is acquired by worker-threading or
// other means.

StatusByteFlags sbFlag = 0;
sbFlag = bp.ReadStatusByte();

short sFlag = (short)sbFlag;

// Status changes
if (sFlag != 0)
{
        // Is the warning register detected?
        if ((sFlag & 2) == 2)
        {
                // Obtain the warning state
                string strWarn = "";
                strWarn = bp.Query("WRER?");
                System.Threading.Thread.Sleep(1000);

        }
}

// Clear the Remote state
new UsbSession(bp.ResourceName).ControlRen(RenMode.Deassert);

// ■Terminate the communication and release the session.
bp.Terminate();
bp.Dispose();
```

Remote Control Programming

NF Corporation

6-3-20, Tsunashima Higashi, Kohoku-ku, Yokohama
223-8508 JAPAN
Phone +81-45-545-8128　Fax +81-45-545-8187