

```

1 using System;
2 using System.IO;
3 using System.Windows.Forms;
4 using Ivi.Visa.Interop; // VISA library namespace specification
5
6 // Sample code for measuring instrument (power supply, load device, etc.) using KI-VISA
7 // Copyright by Kikusui Electronics Corp. 2015-2020
8 // Code Creation: SE2 Section Yajima
9
10 // 2020.09.01 Ver1.00e (Ver1.05 base)
11
12
13 namespace SampleCode_KIVISA
14 {
15     public partial class Form1 : Form
16     {
17         // Create resource manager object
18         IResourceManager3 rm;
19         IMessage msg;
20
21         // Dialog box title
22         readonly string sAppTitle = "KI-VISA Sample for CSharp";
23
24         // Main form
25         public Form1()
26         {
27             InitializeComponent();
28             //Set the form display position to the center of the screen.
29             this.StartPosition = FormStartPosition.CenterScreen;
30         }
31
32         // Processing when displaying form
33         private void Form1_Load(object sender, EventArgs e)
34         {
35
36             // A storage variable for the VISA resource list found by FindResource
37             string[] foundResources;
38
39             // Initialize the Resource Manager
40             rm = new ResourceManager();
41
42             // Get the name of a VISA resource containing an INSTR available in FindResource
43             // For more information, please refer to KI-VISA Library
44             // VISA COM Guidebook 8 - Dynamically Setting VISA Addresses (starting on page 48).
45             try
46             {
47                 foundResources = rm.FindRsrc("?*INSTR");
48             }
49             catch (Exception)
50             {
51                 MessageBox.Show("No available VISA resources found on the computer." + "¥r"
52                     + "Make sure your device is connected",
53                     sAppTitle, MessageBoxButtons.OK, MessageBoxIcon.Stop);
54                 return;
55             }
56
57
58             // Add the acquired VISA resource name to the VISA Resource Name combo box.
59             cmb_Resources.DataSource = foundResources;
60
61
62             // Add a setting command to the combo box.
63             // Commands not listed in the combo box can be entered manually at run time.
64             // If you edit the commands here, you can select any command from the pull-down menu.
65             // As a typical example, the following commands are available for the
66             // PMX18-5A (DC power supply 18V/5A model).
67             cmb_CommandString.Items.Add("SYST:REMOTE");
68             cmb_CommandString.Items.Add("SYST:LOCAL");
69             cmb_CommandString.Items.Add("*CLS");
70             cmb_CommandString.Items.Add("OUTP ON");
71             cmb_CommandString.Items.Add("OUTP OFF");
72             cmb_CommandString.Items.Add("VOLT 10");

```

```

73     cmb_CommandString.Items.Add("VOLT 1");
74     cmb_CommandString.Items.Add("CURR 1");
75     cmb_CommandString.Items.Add("CURR 0.1");
76     cmb_CommandString.SelectedIndex = 0;           // Display the first command
77
78     // Add a query command to the combo box
79     cmb_QueryString.Items.Add("*IDN?");
80     cmb_QueryString.Items.Add("*STB?");
81     cmb_QueryString.Items.Add("SYST:ERR?");
82     cmb_QueryString.Items.Add("OUTP?");
83     cmb_QueryString.Items.Add("VOLT?");
84     cmb_QueryString.Items.Add("CURR?");
85     cmb_QueryString.Items.Add("READ:VOLT?");
86     cmb_QueryString.Items.Add("READ:CURR?");
87     cmb_QueryString.SelectedIndex = 0;           // Display the first command
88
89     // Disable button before opening a VISA session.
90     btn_SendCommand.Enabled = false;
91     btn_SendQuery.Enabled = false;
92     btn_SerialPolling.Enabled = false;
93
94     //-----
95     // [Caution] Searching for LAN Devices
96     // Dynamic search may not be available for LAN devices.
97     // In that case, you can use it by Finding with KI-VISA Instruments Explorer.
98     //-----
99 }
100
101 // Open VISA session
102 private void btn_Open_Click(object sender, EventArgs e)
103 {
104     // For more information, see KI-VISA Library VISA COM Guidebook 6 – Opening VISA Sessions (
starting on page 35).
105     IGpib gpib;
106     ISerial seri;
107     ITcpipInstr tcp;
108     IUsb usb;
109
110     // Open a VISA session with the resource name specified in the combo box
111     try
112     {
113         msg = (IMessage)rm.Open(cmb_Resources.Text, AccessMode.NO_LOCK, 0, ""); // Support the
change of timeout specification in NI-VISA
114         MessageBox.Show("VISA session opened successfully!" + "¥r" + cmb_Resources.Text, sAppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
115     }
116     catch (Exception)
117     {
118         MessageBox.Show("VISA session opening failed." + "¥r" + cmb_Resources.Text, sAppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Stop);
119         return;
120     }
121
122     // Configuration by Interface Type
123     // Please refer to KI-VISA Library VISA COM Guidebook 7 – Basic IO (from page 39 onwards) for
details.
124     msg.TerminationCharacter = 10;
125     msg.TerminationCharacterEnabled = true; // When performing ReadString() on an IEEE 488.2
measuring instrument,
126                                           // "true" setting is recommended for correct terminating.
127     msg.SendEndEnabled = true;
128
129     // Configure interface type by HardwareInterfaceType
130     switch (msg.HardwareInterfaceType)
131     {
132     case 1:
133         // GPIB
134         gpib = (IGpib)msg;
135         gpib.RepeatAddressingEnabled = true;
136         gpib.UnaddressingEnabled = false;
137
138         // Enable the serial polling button (since only GPIB and USB can be used)

```

```

139         btn_SerialPolling.Enabled = true;
140
141         break;
142
143     case 4:
144         // ASRL(Serial port)
145         // You can set the baud rate by changing the parameters here.
146         seri = (ISerial)msg;
147         seri.BaudRate = 19200; // baud rate 19200bps
148         seri.DataBits = 8; // Data Bits 8bit
149         seri.StopBits = SerialStopBits.ASRL_STOP_ONE; // Stop Bits 1bit
150         seri.Parity = SerialParity.ASRL_PAR_NONE; // No parity
151         seri.FlowControl = SerialFlowControl.ASRL_FLOW_NONE; // Without flow control
152
153         // Disabling the serial polling button
154         btn_SerialPolling.Enabled = false;
155
156         break;
157
158     case 6:
159         // LAN
160         // [Caution] Dynamic search may not be available for LAN devices.
161         // In that case, you can use it by Finding with KI-VISA Instruments Explorer.
162         tcp = (ITcpipInstr)msg;
163
164         //Disabling the serial polling button
165         btn_SerialPolling.Enabled = false;
166
167         break;
168
169     case 7:
170         // USB
171         usb = (IUsb)msg;
172
173         // Enable the serial polling button(since only GPIB and USB can be used)
174         btn_SerialPolling.Enabled = true;
175
176         break;
177     }
178
179     // Enable the button.
180     btn_SendCommand.Enabled = true;
181     btn_SendQuery.Enabled = true;
182 }
183
184 // Sending commands
185 private void btn_writeString_Click(object sender, EventArgs e)
186 {
187     int r;
188
189     // Send the command selected by cmb_CommandString
190     r = msg.WriteString(cmb_CommandString.Text + "\n"); // Add a delimiter after the command
191 message
192
193     // If you want to write the command directly, you can also write it like this.
194     // r = msg.WriteString("VOLT 10\n");
195     // r = msg.WriteString("OUTP ON\n");
196 }
197
198 // Example of sending query command and receiving result
199 private void btn_SendQuery_Click(object sender, EventArgs e)
200 {
201     int r;
202     string strret = "";
203
204     // query command sending
205     r = msg.WriteString(cmb_QueryString.Text + "\n"); // Add a delimiter after the command
206 message
207
208     // Receive query results
209     try

```

Form1.cs

```
209     {
210         strret = msg.ReadString(1024); // Buffer size 1024 bytes
211     }
212     catch (Exception)
213     {
214         MessageBox.Show("Failed to receive query message."
215             + "¥r" + cmb_Resources.Text,
216             sAppTitle, MessageBoxButtons.OK, MessageBoxIcon.Stop);
217         return;
218     }
219
220
221     //Returns the value received by ReadString.
222     tb_Result.Text = strret;
223 }
224
225 //Example of serial polling the status byte register
226 private void btn_SerialPolling_Click(object sender, EventArgs e)
227 {
228     short stb;
229
230     // Serial polling the status byte register.
231     // ReadSTB is only available for GPIB and USB interfaces.
232     // For other interfaces, the button are disabled. (because exception is occur)
233     stb = msg.ReadSTB();
234
235     // Write result data to text box
236     tb_Result.Text = stb.ToString();
237 }
238
239 // Processing when the form was closed
240 private void quitToolStripMenuItem_Click(object sender, EventArgs e)
241 {
242     // Close the VISA session
243     if (msg != null)
244     {
245         msg.Close();
246     }
247     // Close the form
248     this.Close();
249 }
250
251 // Clear the Result text box
252 private void btn_ClearResult_Click(object sender, EventArgs e)
253 {
254     // Clear the Result text box
255     tb_Result.Text = "";
256 }
257
258 // AboutBox Version/Contact Information Display
259 private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
260 {
261     MessageBox.Show(
262         "Samplecode for KI-VISA CSharp Ver1.00e" + "¥r"
263         + "Copyright by Kikusui Electronics Corp. 2015-2020" + "¥r", sAppTitle);
264 }
265 }
266 }
267
268
269 }
```