

## Python programming example

This example uses VSCode as the editor and python as the programming language

This example implements functions: find instruments, connect instruments, send commands, and read instrument return values.

1. To install the python package library of the visa dynamic library, use the command: `pip install pyvisa`
2. Install the python interface library, use pyqt5 here, use the command: `pip install pyqt5`
3. Create a new main.py file to add library references, write interface controls, and write control binding functions

```

1  import pyvisa as visa
2  import sys
3  from PyQt5.QtWidgets import QWidget, QLabel, QApplication, QPushButton, QLineEdit, QTextEdit
4
5  class Example(QWidget):
6      def __init__(self):
7          super().__init__()
8          self.initUI()
9          self.rm = visa.ResourceManager()
10
11     def initUI(self):
12         self.setGeometry(300, 300, 660, 400)
13         self.setWindowTitle('DemoPython')
14
15         lbl1 = QLabel('Instrument Address', self)
16         lbl1.move(22, 22)
17         self.textBox_address = QLineEdit(
18             "USB0::0x049F::0x605E::CN999999998::0::INSTR", self)
19         self.textBox_address.setMinimumWidth(362)
20         self.textBox_address.height = 23
21         self.textBox_address.move(162, 19)
22         self.btn_connect = QPushButton("connect", self)
23         self.btn_connect.move(548, 19)
24
25         lbl2 = QLabel('Command', self)
26         lbl2.move(22, 88)
27         self.textBox_command = QLineEdit('read?', self)
28         self.textBox_command.setMinimumWidth(362)
29         self.textBox_command.height = 23
30         self.textBox_command.move(162, 85)
31
32         self.btn_send = QPushButton("Send Command", self)
33         self.btn_send.move(171, 138)
34         self.btn_read = QPushButton("Read Response", self)
35         self.btn_read.move(302, 138)
36         self.btn_sendAndRead = QPushButton("Send & Read", self)
37         self.btn_sendAndRead.move(432, 138)
38
39         self.logEdit = QTextEdit('', self)
40         self.logEdit.isReadOnly = True
41         self.logEdit.setMinimumWidth(600)
42         self.logEdit.setMaximumHeight(142)
43         self.logEdit.move(22, 214)
44
45         self.btn_connect.clicked.connect(self.btn_connect_click)
46         self.btn_send.clicked.connect(self.btn_send_click)
47         self.btn_read.clicked.connect(self.btn_read_click)
48         self.btn_sendAndRead.clicked.connect(self.btn_sendAndRead_click)
49         self.show()

```

```

52     def btn_connect_click(self):
53         address = self.textBox_address.text()
54         if address != '':
55             self.inst = self.rm.open_resource(address)
56         else:
57             addressList = self.rm.list_resources()
58             for _address in addressList:
59                 _inst = self.rm.open_resource(_address)
60                 if _inst != None:
61                     _inst.visalib.write(_inst.session, '*IDN?' + '\n')
62                     readData = _inst.visalib.read(_inst.session, 100)
63                     readValue = str(readData[0])
64                     if readValue.__contains__('HDM'):
65                         self.inst = _inst
66                         self.textBox_address.setText(_address)
67             return
68
69     def btn_send_click(self):
70         cmdStr = self.textBox_command.text() + '\n'
71         self.inst.visalib.write(self.inst.session, cmdStr)
72
73     def btn_read_click(self):
74         readData = self.inst.visalib.read(self.inst.session, 100)
75         readValue = str(readData[0])
76         self.logEdit.append(readValue)
77
78     def btn_sendAndRead_click(self):
79         cmdStr = self.textBox_command.text() + '\n'
80         self.inst.visalib.write(self.inst.session, cmdStr)
81         readValue = self.inst.visalib.read(self.inst.session, 100)
82         # print(readValue)
83         self.logEdit.append(str(readValue[0]))
84
85
86 if __name__ == '__main__':
87     app = QApplication(sys.argv)
88     ex = Example()
89     sys.exit(app.exec_())

```

4. Modify the address in the LineEdit control to the address of the command instrument to test the connection, send commands, read the return value and other functions.