A VXI-11 MODULE FOR PYTHON LANGUAGE AND ITS APPLICATION TO ACCELERATOR CONTROLS

Noboru Yamamoto†
Japan Proton Accelerator Research Complex (J-PARC, KEK & JAEA)
High Energy Accelerator Research Organization Tokai Campus
Shirakata-Shirane 2-4, Tokai, Naka, Ibaraki
Japan 319-1195

ABSTRACT

VXI-11 is an industrial standard to control equipment through network. A module to control these equipment through Python scripting Language was developed. This module can be used for quick testing of equipment and for the rapid application development. The implementation of the module will be discussed and some application of the module will be reported.

WHAT IS VXI11

VXI-11is an international standard for control of equipment over standard TCP/IP network. You may consider VXI-11 as network based GP-IB. Although GP-IB is stilly used widely in the field, most modern measurement devices are equipped with Ethernet interface and support VXI1 standard over Ethernet. Using Ethernet based VXI-11, we can setup measurement system using a standard PC and these equipment without additional hardware, such as GPIB interface boards or boxes..

VXI11 standard[1, 2, 3, 4] defines a network instrument protocol and its API based on ONC/RPC and its mapping to VXI-bus/IEEE 488.1[5]/IEEE488.2[5] devices. VXI-11 network instrument protocol is defined based on the well accepted network standard such as ONC/RPC, XDR and TCP/IP. The definition of the protocol is given as RPCL description in the VXI11 standard documentation. An ONC/RPC tool, rpcgen, can be used to generate C language source code and header files for the RPC server and clients.

VISA and PyVISA

If you want utilize VXI-11, there are several ways. For example, you can purchase commercial software which supports VISA(Virtual Instrument Standard Architecture), such as LabView[7]L from National Instrument VISA(Virtual Instrument Standard Architecture) is a standard defined by the VXIplug & play Systems Alliance including Agilent Technologies and National Instruments. These company provides VISA compatible drivers for their products. VISA supports VXI11 as one of supported busses. VISA libraries are supplied by multiple companies and should be compatible in principle. Its license is usually included in the products and/or application development environment.

Python module to access VISA library, PyVISA[10] is also available as a free software. Combining PyVISA with other Python modules, you can create a control application efficiently. Once you have VISA driver installed on you PC, PyVISA will work without additional software. In the environment with equipment from multiple vendors, the compatibility with the library and driver can be a problem. VISA driver from one company may or may not work with the device from other company. You mus alos careful if the VISA driver license allow you to use it in such a mixed environment.

WHY PYTHON-VXI11?

When we get a new measurement device, we usually want to run simple tests to check the functionality and behavior of the device which are not so obvious from just reading a documentation. Combination of VISA and PyVISA can be a good choice for this purpose. Drawback of this approach is that VISA drivers are proprietary product of these commercial vendors. So if the device is not from the vendor of VISA driver, the license will not allow you to use their driver in this case. These drivers are distributed as binary form so you may encounter the incompatibility between your host system(CPU, OS , OS version, and so on) and VISA drivers. Python-VXI11 module can fill this gap.

Python-VXI-11 modules was developed based on Open Source product and distributed as a source code. So if your host system support these open source and TCP/IP network, which is common to most of modern PC, it eliminate the issues related to the proprietarily VISA driver.

Python [8] is an object oriented interpreted programming language, originally designed and implemented by Guido von Rossum. Its source code is distributed as open source and support most modern platform for computing. It is widely used over the world, most notably in the Google. Python is also used to implement large applications such as Mailman and Zope. Python modules including PyLab, Gnuplot, matplotlib for graph generation are provided as

† noboru.yamamoto@kek.jp

standard or external modules. It allow us to develop a program to test a measurement device such as Oscilloscope in very quick way. Python has been used as a tool for testing and even developing control applications in KEKB control system and J-PARC control system.

PYVXI11 IMPLEMENTATION

In this section, we will see how Python-VXI11 module is implemented. The main components of Python-VXI11 module are `VXI11.rpcl` , `VXI11.i` and *setup.py*.1 VXI11.rpcl is a VXI-11 protocol description in ONC/RPC standard. It is converted to C source code and header files using "rpcgen" command. VXI11.i is an input file for SWIG[9]. It is mostly direct conversion from a header file generated by "rpcgen". "swig" command converts this file into C source code and header file and python module which is imported into Python interpreter. In other words, we don't need much, if any, C programming for Python-VXI11 module. "setup.py", like "Makefile" automate the process to build and install the module into your system. Python-VXI11 modules also includes some pure python modules to help use of Python-VXI11 module.

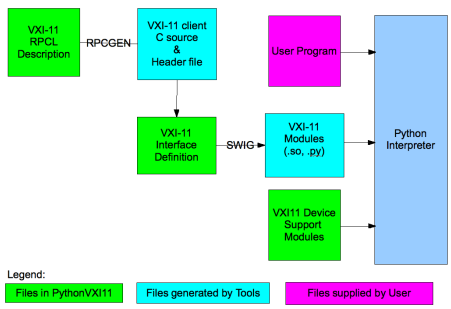

Figure 1: Components in Python-VXI11 module and their relations

Binary data transfer and "END" bit handling

VXI-11 allows us to exchange waveform data in binary format. So Python-VXI11 module developed here also supports it. VXI-11 defines "reason" field in the structure for the response to "read" operation. "reason" field of the Device_ReadResp structure consists of "END", "REQCNT" and "CHR" bit fields. Use of these fields are defined in VXI-11 specification as:

RULE B.6.23: To successfully complete a device_read RPC, a network instrument server *SHALL*:

1. Transfer bytes into the data parameter until one of the following termination conditions are met:
   (a) An END indicator is read. The END bit in reason *SHALL* be set.
   (b) requestSize bytes are transferred. The REQCNT bit in reason *SHALL* be set. This termination condition *SHALL* be used if requestSize is zero.
   (c) termchrset is set in flags and a character which matches termChar is transferred. The CHR bit in reason *SHALL* be set.
   (d) The buffer used to return the response is full. No bits in reason *SHALL* BE set.
2. Return with error set to 0, no error, to indicate successful completion. If more than one termination condition is valid, reason contains the bitwise inclusive OR of all the reasons.

As far as a device follows these rules, binary data exchange using Python-VXI11 modules should not have problem. If you encounter the device which does not strictly follows this rule, you may need some tweak to Python-VXI11 module.

USAGE

Use of Python-VXI11 is simple and straightforward. The table 1 shows a sample code to read waveform data from an oscilloscope and display these waveforms on the screen of the host. A supplement python module, TekOSC.py, hides the detail of Python-VXI11 module and can be used as a template to support your device.

```
import TekOSC # TekOSC module uses
              Python-VXI11 module internally
import time, pylab, matplotlib

#-----  generate oscilloscope object
osc=TekOSC.TekOSC("xx.xx.xx.xx")
#-----  setup a oscilloscope
osc.set_wf_binary()
osc.set_fulldata()
#-----  get waveforms from channels 1 and 2.
w1=osc.get_waveform(1)
w2=osc.get_waveform(2)

#-----  set up graph for drawing data
fig = pylab.figure()
fig.subplots_adjust(hspace=0.2, bottom=.14,
 left=.14, right=.97, top=.92)
pylab.subplot(1, 1, 1)
pylab.title("PyVXI-11 Sample with PyLab")
pylab.xlabel("time(sec)")
pylab.ylabel("signal(volt)")
#-----  plot waveform 1 and 2
pylab.plot(w1.x,w1.y)
pylab.plot(w2.x,w2.y)
#-----  show graph on a screen.
pylab.show()
```

Table 1: A python script to generate the figure 2

This simple script of Python generates the graph shown in the figure 2

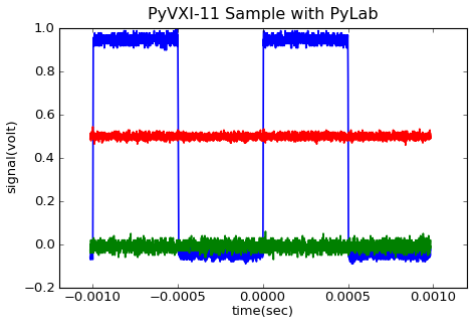

Figure 2: Sample display of waveforms taken from a VXI11 based oscilloscope

CONCLUSION

Python module to access VXI11 compatible device is developed. Combination of python interpreter and this modules simplifies the test of these equipment. Rich set of libraries available in Python even make it possible to develop the application used for daily operation of accelerator. Currently implementation of the modules is based on SWIG to generate glue routines for Python and RPC library. To improve the performance of the module, ctypes module, which is now a part of standard Python library, or cython, another wrapper generator for Python-C library can be used to replace SWIG.

REFERENCES

[1] " VMEbus Extensions for Instrumentation: TCP/IP Instrument Protocol Specification, VXI-11, Revision 1.0.", The VXIbus Consortium, 1995 http://www.vxibus.org/files/VXI_Specs/VXI-11.zip.

[2] "VMEbus Extensions for Instrumentation: TCP/IP-VXIbus Interface Specification, VXI-11.1, Revision 1.0. ", The VXIbus Consortium,1995

[3] "VMEbus Extensions for Instrumentation: TCP/IP-IEEE 488.1 Interface Specification, VXI-11.2, Revision 1.0.", The VXIbus Consortium,1995

[4] " VMEbus Extensions for Instrumentation: TCP/IP-IEEE 488.2 Instrument Interface Specification, VXI-11.3, Revision 1.0.", The VXIbus Consortium,1995

[5] "IEEE Standard for Higher Performance Protocol for the Standard Digital Interface for Programmable Instrumentation", IEEE Std 488.1-2003 (Revision of IEEE Std 488.1-1987), 2003

[6] "Standard Digital Interface for Programmable Instrumentation - Part 2: Codes, Formats, Protocols and Common Commands (Adoption of (IEEE Std 488.2-1992)", IEC 60488-2 First edition 2004-05; IEEE 488.2, 2004

[7] LabView is a name of product by National Instrument.

[8] "Python Tutorial", Guido von Rossum, O'Reilly Japan, Tokyo, 2007; http://www.python.org; http://www.python.jp .

[9] "SWIG : An Easy to Use Tool for Integrating Scripting Languages with C and C++", David M. Beazley, the 4th Annual Tcl/Tk Workshop, Monterey, CA. July 6-10, 1996.;"SWIG 1.1 Users Manual", David M. Beazley, available online at http://www.swig.org/Doc1.1/HTML/Contents.html, 1977.

[10] "PyVISA (Release 1.1)", Torsten Bronger, available online at http://sourceforge.net/projects/pyvisa/, 2006.