Department of Communications and Networking

Remote control of measurement equipments

Aalto University
School of Electrical
Engineering

January 2, 2023

1 Introduction

Remote management of measurements equipments means measurements automation and automatic collection of measurements results. The automatic control is usually realized remotely over network. The remote management environments allow the sequence of equipments control messages that can be executed as remote programs. Over the network one control unit could manage multiple test equipments. Equipments collected into such remote control systems are called automatic test equipments (ATE).

Purpose of this document is to introduce to the history of test automation and to give short introduction of alternative remote control methods. In more detail the document describes SCPI programming language and most general methods for sending control commands to the equipments.

1.1 History of measurement automation

Automatic tests have been integral part of aerospace industry since middle of 20'th century. In particular, US Air force developed and used automatic tests for validating various airplane subsystems. Firs general purpose automatic test system GPATS contained over 20 different equipment. Each of these were controlled remotely over private bus and with compute programmed by punch cards. Also private companies developed similar systems: Aeronautical Radio Inc incorporation (ARINC) published first test automatic language for avionics systems (ATLAS) as its own standard language in 1966 (ARINC-416). Nowadays the trend is, as also in other fields, to use XML based control language (IEEE Std 1671. *) **ate-trends**.

For standardizing interfaces between the control equipments Hewlet-Packard (HP now Keysight) developed in 1970s a special HP-IP bus (*Hewlett Packard Interface Bus*) that in 1975 was accepted as IEEE specification (IEEE-488). Since other companies opposed HP-IP name in this specification the name of the bus was changed to *General Purpose Interface Bus* (GPIB) **gpib-101**.

In 1987 IEEE introduced IEEE-488.2 standard and the IEEE-488 standard was renamed to IEEE 488.1. The 488 standards first part IEEE 488.1 defines which equipments can be used in test set ups and second part IEEE 488.2 defines which protocols and commands are in use.

While both hardware and protocols were standardized, the used commands were equipment and producer specific. In 1989 HP developed a protocol that unified commands over various equipments. In 1990 this protocol was the based for SCPI (textit Standard Commands for Programmable Instrumentation) standard. The SCPI uses IEEE-488.2 syntax and defines most commonly used commands **ate-trends**.

1.2 GPIB

GPIB bus is the standard that defines hos different equipments can communicate with each other. GPIB bus uses 24 pin version of the SCSI 50 pin connector. The structure of the connector is defined in IEEE-488.1 **ieee-4881**. GPIB bus is capable of 8 bit parallel transmission (i.e. it has 8 I/O pins). Rest of the pins are used for handshaking (3 pins), management (5 pins) and for ground (8 pins). The bus data speed is 1 Mb/s. In later protocol version and extensions the protocol is simplified and the bus is capable to 8 Mb/s transmission speed.

Each equipment has unique 5 bit primary address (address is selected in range from 0 to 30). Maximum of 15 equipments can be connected to one physical bus. Between two equipments the cable has to be less than 2 m. Total maximum bus length is about 20 m. The connector design allows daisy chaining of multiple equipments **ieee-4881**.

GPIB is relatively old standard, that it will be replaced with new technologies has been predicted since 1994**gpib-shoulder**. However, even nowadays most equipments are provided with Ethernet and GPIB interface. GPIB popularity can be explained by simple protocol, good endurance of cables and connectors, relatively few telecommunication problems and long lifetime of industrial equipments. **gpib-never-dead**.

1.3 SCPI command language

The test equipments control commands are defined by SCPI standard (textit Standard Commands for Programmable Instruments). SCPI standard defines how some specific equipment property has to be presented. This allows replacing the measurements equipments with similar equipments from other manufacturers without changes to control programs cite [s.1- 2–1-3] SCPI-standard

SCPI defines hierarchical commands structure. The commands are separated by the colon. The *keyword* after colon belongs to lower hierarchy level. Alternatively the keywords and parameters can be separated by square brackets ([]). Commands have both long and short form and a short form. The commands short forms are written with capital letters. For instance, the command the long form of the command "FREQuency:CW" has corresponding short form "FREQ:CW". Both forms can be used for device control. A command could be interpreted as query by inserting question mark (?) after it cite [p. 5-1–5-2] SCPI-standard.

The following is an example of a script that resets a device settings (IEEE-488.2 command), adds cursor to a spectrum analyzer screen, places the courser on the spectrum peak and reads the peak value

```
*RST
CALCulate:SPECtrum:MARKer:ADD
```

```
CALCulate:SPECtrum:MARKer0:MAX
CALCulate:SPECtrum:MARKer0:X?
```

Commands are usually self explanatory. Equipment specific commands can be found from the equipment manuals, of from specific device specific programming manuals.

Commands can be grouped by separating them with semicolon. Command groups can be send together. By default the grouped commands are executed at the same hierarchical level. For instance, the command above could be expressed in its short form

```
*RST
CALC:SPEC:MARK:ADD; :CALC:SPEC:MARK0:MAX; X?
```

In addition to SCPI commans IEEE 488.2 standards defines some of general commands. For instance, command for resetting the device ***RST**. Other similarly defined commands are also device ID query command ***IDN?** and *operation complete* query.

1.4 Remote control

Majority on measurement devices are equipped with Ethernet interface and they can be connected to the local area network (LAN). For instance Rhode & Schwarz Linux based devices (such as SMBV100A class signal generators **smbv-manual**) listen port 5025. To this port we can send SCPI control signals by using Telnet connection of by writing simple program that feeds commands to this port. Since most modern devices use Windows or Linux operating system it is possible to use VNC or RDC sessions. However, such session do not provide as flexible programming possibility as sockets based connections do.

Virtual Instrument Software Architecture (VISA) is a software that provides simple unified management interface for controlling multiple devices connected to a same bus. VISA standard contains rules how the devices communicate over different buses, such as GPIP or VXI.

VISA interface is available for many programming languages. For instance there is API for C programming language, for Microsoft .NET environment and for Python (PyVISA).

2 Equipment

In this laboratory work we are using following equipments

- Computer where is installed Nation Instrument (NI) VISA program with corresponding drivers.
- NI GPIB-USB-HS adapter and GPIB cable
- SMBV100A signal generator from Rohde & Schwarz
- RSA 6114A spectrum analyzer from Tektronix
- One cable with N type connectors (male - male)

3 Instructions

The instructions below and designed for measurement set up described in Figure 1. Computer and signal generator are connected to a same network through a switch. Spectrum analyzer is connected to the computer through National Instrument GPIB-USB connector.

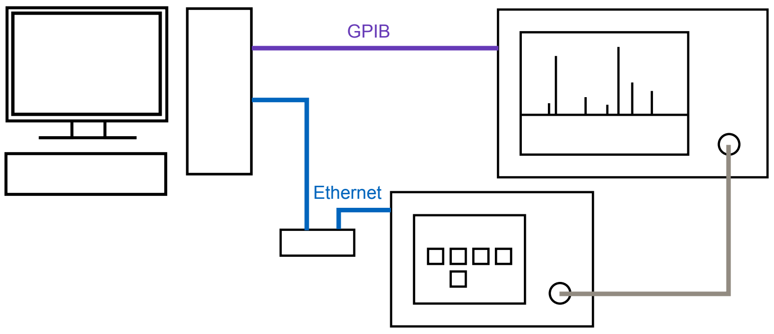

Figure 1. Measurement setup.

3.1 Remote control by using Telnet connection

First we control SMBV100A signal generator over Telnet connection, in other words we connect computer directly to the port that the generator is listening.

1. Start **PuTTY** program in the computer. In the main windon set the connection type to *Telnet* and port 5025. Set the IP address to be IP address of the signal generator (should be glued on top of the generator). Select **Open** from the bottom of the window. In the opened window push ENTER key and the Telnet connection should start.

2. Tell to the signal generator to identify itself by typing ***IDN?**. Check that the answering device is SMBV100A.

   Go through the following commands and send each command to the device (one by one).

   ```
   // Reset generator configurations and state of the registers.
   // (It operates as RESET button)
   *RST
   *CLS

   // Ask the device to report permanent errors
   // (despite of reset these errors are permanent)
   SYST:SERR?
   ```

```
// Use ARB feature and create 5000 Hz square pulse
// the pulse has 500 samples per pulse and full (100%) amplitude.
BB:ARBitrary:TSIGnal:RECTangle:FREQuency 5000
BB:ARBitrary:TSIGnal:RECTangle:SAMPles 500
BB:ARBitrary:TSIGnal:RECTangle:AMPLitude 1
BB:ARBitrary:TSIGnal:RECTangle:OFFSet 1
BB:ARBitrary:TSIGnal:RECTangle:CREate

// Set frequency to 30 3 GHz ja signal level to -25 dBm
FREQuency 3000000000
LEVel -25

// Start generators Baseband and RF/A Mod blocks
BB:ARBitrary:STATe 1
OUTPut:STATe 1
```

After those commands in the signal generator screen the blocks **Baseband** and **RF / A Mod** (and automatically also **I / Q Mod**) should now be enabled and feed continuous square waveform (as defined).

3.2 Control over GPIB bus (NI-488.2)

In this section we explore how GPIB operates and how to submit commands defined in IEEE-488 standards. GPIB bus is used in National Instruments Interactive Control application. The application reads and writes data directly from/to the bus.

3. Start Interactive Control application from the desktop (or start it from NI-488.2 folder that is located inside of National Instruments folder). Interactive Control application is a command line window, where the commands are directly entered into the device connected to the GPIB bus.

4. By default, the command line is in "general" **:** mode. To be able to manage the device, we have to open connection to it. The connections towards devices are called *device handles*. A device handle is created with **ibdev** command.

   In order to open connection you have to know the device IP address. You check the IP of the spectrum analyzer from its **Tools** menu **Options** window. The GPIP address is on **GPIB** tap. On the same menu you can also change the IP.

5. Ask the spectrum analyzer parameters by submitting **ibdev** command. A computer may have multiple GPIB connectors. The

address of the GPIP driver where you would like to connect your device is given by *Board index*.

The primary address is the address you give to the spectrum analyzer. The secondary address is 0 ie. there is no address. Additonally you can define connection timeout in seconds (eq. 10). Switch END message visibility on and EOS state off. The code below describes establishing a connection to a deveice at address 13.

```
Interactive Control
Copyright 2007 National Instruments Corporation
All rights reserved.

Type 'help' for help or 'q' to quit.


: ibdev
    enter board index: 0
    enter primary address: 13
    enter secondary address: 0
    enter timeout: 10
    enter 'EOI on last byte' flag: 1
    enter end-of-string mode/byte: 0

ud0:
```

6. After establishing a connection device gives you a device handle (**ud0:**). Send to the handle command **ibclr** that initializes handlers all previous settings.

(!) 7. GPIB is a serial bus. It carries one byte of a data at a time. Interactive Control program writes data with **ibwrt** command and reads with **IBRD** command. **IBRD** command has a parameter that indicates how many bytes should be read.

   Ask from the analyzer its ID by submitting **"*IDN?"** using the **ibwrt** command and reading the answer by **IBRD** command. How the application/analyzer indicates that the all the bytes in the answer have been read? What happens if you try to read the analyzer after that?

(!) 8. Use **ibwrt** command and send to the analyzer resetting commands **"*RST"** and **"*CLS"**. Ask for the current errors in the system by entering into **ibwrt** device specific command **"SYSTem:ERRor:COUNt?"**. The transmitted SCPI command is placed inside quotation marks. Check what was the answer from the device by using **ibrd 10** command. This command reads 10 bytes of the analyzer answer data. What message the device uses for answering the inquiry?

(!) 9. Ask spectrum analyzer to its current center frequency by using command **SPECtrum:FREQuency:CENTer?**. In what format the spectrum analyzers shows the center frequency?

10.
   - Set the spectrum analyzer center frequency to 3 GHz by using command **SPECtrum:FREQuency:CENTer**.
   - Test in which formats you can provide the frequency information.
   - Set the frequency range 5 MHz by using command **SPECtrum:FREQuency:SPAN**.
   - Check whether there were any errors by using commands **SYSTem:ERRor:COUNT?** and **SYSTem:ERRor:ALL?**.

11. Set the spectrum analyzer resolution bandwidth to 1KHz. For that you can use command **SPECtrum:BANDwidth:RESolution**.

    Set new market on the spectrum view screen. The corresponding command is **CALCulate:MARKer:ADD**.

(!) 12. Move the marker to the maximum spectrum peak. The command is **CALCulate:SPECtrum:MARKer0:MAXimum**. Read the marker frequency and amplitude. The commands are **CALCulate:SPECtrum:MARKer0:X?** and **CALCulate:SPECtrum:MARKer0:Y?**. Record the values for the report.

(!) 13. Move the marker to the next peak. The command is **CALCulate:SPECtrum:MARKer0:PEAK:RIGHt**. Read this marker location frequency and compute distance to the previous marker location. Why there are those peaks? What is the source of these peaks?

14. Close the device handle by command **ibonl 0** Use command **q** to close the program.

3.3 VISA

Controlling devices from GPIB interface with command line is laborious. Beside, ofter the same set up (command sequence) is used to control multiple devices. For simplifying GPIB programming we can use VISA programming architecture.

15. Before we can control a device we have to add it to VISA library. For that: start from the desktop or from National Instruments directory "NI Measurement & Automation Explorer" program.

    First we add spectrum analyzer. Open from tree view **Devices and Interfaces** subbranch and there you find list of registered devices. Run the option **GPIB0 (GPIB-USB-HS)** which scans for available instruments. After the scan the spectrum analyzer should be visible in the list.

Next we add the signal generator. Right click on **Devices and Interfaces** and select **Create New VISA TCP/IP Resource...**. You will see a window where you select **Auto-detect of LAN Instrument** and then **Next >**. The program scans the local network for available devices and show them in the list.

From the list select signal generator. If there are more than one you find right one by its IP address. Click on **Finish** button. The signal generator should appear under **Network Devices** options.

One a device is added to VISA library it can be used by any program that uses VISA driver. The interface can be used not only for libraries for various programming languages but also by scripting languages as MATLAB. In MATLAB VISA support is included into *Instrument Control Toolbox*.

(!) 16. Appendix contains A is shown a Python script, which measures channel power with spectrum analyzer. Explore the script. Can you make any improvements to the script?

17. Copy the script in the Annex into a file (remember to indent the required four rows) and run it using "IDLE (Python GUI)". (Hint: in order to run the script you need Python 2.x environment and PyVISA package to be installed).

A PyVISA example

The following Python script uses PyVISA library to connect to a spectrum analyzer. The script resets the spectrum analyzer settings, opens a view for measuring a channel power and makes a channel power measurement at 3 GHz.

```python
import visa # Import the PyVISA library
import time # Python's time library for sleeping

frequency = "3e+9" # Center frequency at 3 GHz
identifier = "GPIB0::13::INSTR" # VISA ID of the analyzer
resourcemanager = visa.ResourceManager()

# List all available VISA devices
print "Available VISA devices:"
for device in resourcemanager.list_resources():
        print device

# Connect to the spectrum analyzer
instrument = resourcemanager.get_instrument(identifier)
idn = instrument.ask("*IDN?")
print "Connected to " + idn
instrument.write("*RST; *CLS")

# Add the channel power display and adjust its settings
instrument.write("DISPlay:GPRF:MEASview:NEW ACPower")
instrument.write("SENSe:ACPower:FREQuency " + str(frequency))
instrument.write("SENSe:ACPower:CHANnel:PAIRs 0")
instrument.write("SENSe:ACPower:BANDwidth:RESolution 1.0e+3")
time.sleep(2) # Let the analyzer settle

# Perform the measurement only if the device has no errors
error_count = instrument.ask("SYST:ERR:COUN?")
print "Error count: " + error_count.rstrip("\n")
if not int(error_count) == 0:
        print instrument.ask("SYST:ERR:ALL?")
else:
        power = instrument.ask("FETCh:ACPower:CHANnel:POWer?")
        print "Channel power: " + power
```

B Exercises

1. Find the errors in the following SCPI commands? You can assume that the names of the commands are correst, ie. there is no error in the names.

```
SPEC:FREQ:SPAN 100000; SPEC:FREQ:CENT 100000000
INPut:ATTenuation 0
DISP:WIND:ACT:MEAS?; :DISP:GEN:MEAS:NEW SGR
:CALC:MARK:ADD
```

2. Howe we can contol remotely through SCPI, that the command sent to the device has been executed?

3. Explore SCPI standard **scpi-standard**. Why after **RST* command is often followed by **CLS* command. What the **CLS* command does in addition to the command **RST*?