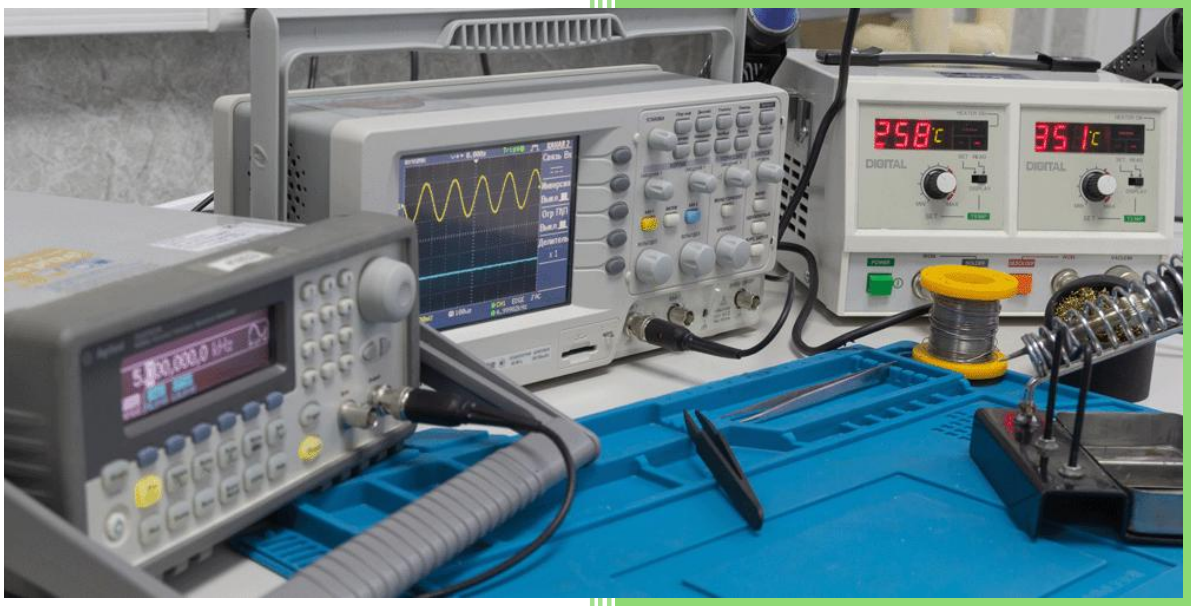
2025

SCPI Fundamentals and Industrial Applications



Fundamentos do SCPI e sua Aplicação na Indústria

Gabriel Marques

Abstract

This work presents a practical approach to instrument automation using Python, SCPI commands, and the VISA communication standard. The methodology enables deterministic and repeatable electrical testing for industrial validation and research environments. A programmable power supply is remotely controlled to execute a fast voltage ramp typically used in module validation procedures, ensuring high levels of traceability and measurement reliability. The proposed solution provides reduced operator dependency, flexible test design, and simplified integration with data acquisition and automated reporting systems. The approach is aligned with modern industrial digitalization trends, especially those associated with smart testing strategies and quality assurance.

Keywords

SCPI; VISA; automation; power validation; programmable power supply; Python; PyVISA; electrical testing; measurement systems.

Summary

1. Introduction

Instrument automation plays a central role in Research & Development (R&D) and validation environments, enabling precise and repeatable execution of electrical testing based on standardized norms and technical procedures. In the automotive sector, automated instrumentation is widely applied in the evaluation of Electronic Control Units (ECUs), power modules, sensors, and actuators, supporting durability testing, compliance with standards such as LV124, LV148, and ISO 16750, and the functional characterization of embedded systems [6].

Through the integration of software tools and programmable instruments—such as power supplies, digital multimeters, and data acquisition systems—human variability is minimized, measurement reproducibility is enhanced, and traceability is reinforced. These are essential requirements for qualification processes and quality audits in industry.

This work demonstrates a practical approach to remotely controlling a programmable power supply using VISA and SCPI commands, applying a fast voltage ramp test typically used in automotive validation [1–7].

2. Fundamental Concepts

2.1 SCPI (Standard Commands for Programmable Instruments)

SCPI (Standard Commands for Programmable Instruments) is a textual command standard developed in 1990 to standardize the remote control of electronic instruments. Its main contribution is ensuring that different equipment — even from different manufacturers — share a common syntax for configuration, measurement, and data querying.

It operates over communication protocols such as:

- USBTMC (USB Test & Measurement Class)
- LAN (TCP/IP)
- GPIB
- RS-232

By abstracting the physical layer, SCPI ensures interoperability, software portability, and reduced development time.

Every SCPI command is formed by headers structured in a hierarchical tree of instrumental functions:

SUBSYSTEM:FUNCTION PARAMETER

SCPI Command	Function
VOLT 12.0	Sets output voltage to 12 V
CURR 1.5	Adjusts current limit to 1.5 A
OUTP ON	Turns output ON
MEAS:VOLT?	Requests voltage measurement

Commands accept uppercase abbreviations:

Full form	Valid Abbreviation
VOLTAGE	VOLT
CURRENT	CURR
OUTPUT	OUTP
MEASURE:VOLTAGE?	MEAS:VOLT?

There are two main types of SCPI instructions:

Type	Symbol	Example	Resul
Command	No ?	OUTP ON	Executes action
Query	?	MEAS:CURR?	Returns reading

Queries are essential for feedback and traceability during validation.

Internal States and Buffers

SCPI allows:

• Managing internal errors
• Querying event registers
• Synchronizing test sequences

Example:

SYST:ERR?

*OPC?

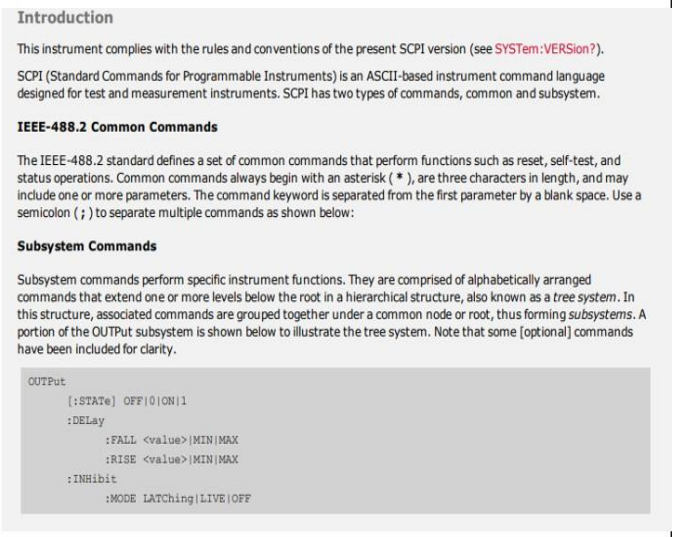
These resources are essential in reliable automation.

For the practical test in this document, we use a Keysight power supply. Therefore, it is necessary to consult the equipment’s operation manual.

SCPI Command
To set the output voltage to 40 volts: VOLT 40

SCPI Command
To set the output current to +5 amperes: CURR 5 To set the output current to -5 amperes: CURR -5

Keysight N6900/N7900 Series Operating and Service Guide



Introduction

This instrument complies with the rules and conventions of the present SCPI version (see SYSTem:VERSion?).

SCPI (Standard Commands for Programmable Instruments) is an ASCII-based instrument command language designed for test and measurement instruments. SCPI has two types of commands, common and subsystem.

IEEE-488.2 Common Commands

The IEEE-488.2 standard defines a set of common commands that perform functions such as reset, self-test, and status operations. Common commands always begin with an asterisk (*), are three characters in length, and may include one or more parameters. The command keyword is separated from the first parameter by a blank space. Use a semicolon (;) to separate multiple commands as shown below:

Subsystem Commands

Subsystem commands perform specific instrument functions. They are comprised of alphabetically arranged commands that extend one or more levels below the root in a hierarchical structure, also known as a *tree system*. In this structure, associated commands are grouped together under a common node or root, thus forming *subsystems*. A portion of the OUTPut subsystem is shown below to illustrate the tree system. Note that some [optional] commands have been included for clarity.

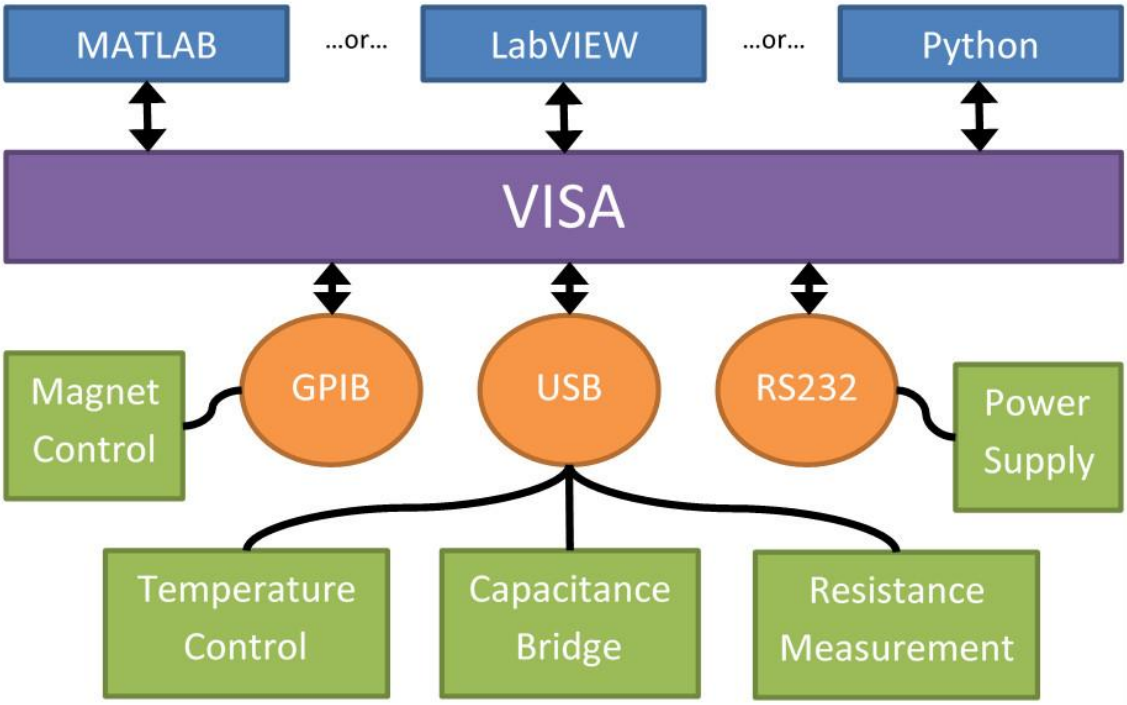
```
OUTPut
      [:STATe] OFF|0|ON|1
      :DELay
            :FALL <value>|MIN|MAX
            :RISE <value>|MIN|MAX
      :INHibit
            :MODE LATChing|LIVE|OFF
```

2.2 VISA (Virtual Instrument Software Architecture) **[3–4]**

VISA (Virtual Instrument Software Architecture) is a standard defined by the VPP-4.3 specification of the VXIplug&play Systems Alliance, created to provide a standardized software interface for communication with test and measurement instruments. Its primary function is to abstract the physical communication medium, allowing a single application to control different instruments regardless of the interface used.

Supported communication protocols include:

- USB (USBTMC — USB Test & Measurement Class)
- LAN (TCP/IP — sockets and VXI-11/HiSLIP)
- GPIB (IEEE-488)
- RS-232 (Serial)



Fonte: Razorbilli instruments, Guide to data acquisition in the modern laboratory

Thanks to this architecture, the application source code does not need modification when connection type or instrument model changes.

Visa Essentials Resources

Feature	***Function***
Instrument Discovery	*Lists connected devices*
Session Management	*Sends SCPI with delivery confirmation*
Command transmission	*Sends SCPI with delivery confirmation*
Data and event handling	*Query respondes and interrupt management*
Error handling	*Realiable diagnostic in automation*

These mechanisms guarantee reliability, traceability, and synchronization — critical requirements in automotive and industrial validation.

2.3 VISA in Python — PyVISA

In Python environments, the PyVISA library implements the VISA API in a cross-platform manner, enabling:

• Direct communication via USB/GPIB/LAN
• Access to SCPI control commands
• Integration with databases and automated testing frameworks

Ideal for Qualification, Durability Testing, and Automotive Homologation environments.

2.4. Python Environment Setup

Install libraries for instrument communication:

pip install pyvisa pyvisa-py,

Optionally install vendor drivers (NI-VISA or Keysight IO Libraries Suite). To list connected devices and initialize communication:

test.py > ...

```python
# Lists all instruments connected via PyVISA
import pyvisa

rm = pyvisa.ResourceManager()
resources = rm.list_resources()
print("Connected instruments:", resources)
```

(Figure 1 — Author's own, 2025)

3. Power Supply Test and Practical Application

Basic example of controlling a programmable power supply via USB: (The supply used in this document was a Keysight unit) [1]

```python
# Lists all instruments connected via PyVISA
import pyvisa
ADDR = "USB0::0X069::0X0346::C010101::INSTR"  # VISA address of the instrument
rm = pyvisa.ResourceManager() # Open VISA resource manager
psu = rm.open_resource(ADDR, timeout=5000 )# Connect to the instrument
psu.write("CURR 15.0")  # Set current to 15.0 A
time.sleep(10)  # Wait for 10 seconds
psu.write("*RST") # Reset the instrument

```

(Figure 2 — Author's own, 2025)

In the example above, this is a simple practical test to verify communication with the power supply.

By setting the current limit to 15 A, we can observe on the front panel whether the command was correctly received and executed.

(Figure 3 — Author's own, 2025)

Once communication is confirmed, Python can be fully used to define programmatic tests and waveforms using only the supply and SCPI communication.

The following example implements a fast voltage ramp from 9 V to 5 V in 5 ms.

This can be done in many ways; a simple method is adjusting the supply's **slew rate**, defined as the rate of change in voltage (or current) over time.

It indicates how fast a signal can rise or fall and is usually expressed in V/s, V/ms, or V/µs.

We need to calculate the required slew rate:

$$\triangle SR = \frac{\triangle V}{\triangle T}$$

(Figure 5 — Slew Rate Formula)

Knowing the voltage transition (9 V → 5 V) and duration (5 ms), the calculation is straightforward:

$$\triangle SR = \frac{(9 - 5)}{0.005} = 800\ V/s$$

(Figure 6 — Slew Rate Calculation)

It is necessary to check the power supply's manual to ensure the device supports this transition rate and to identify the correct SCPI syntax.

4 Using the Advanced Power System

The current slew rate determines the rate at which the current changes to a new programmed setting. This applies to both current settings in current priority mode, and current limit settings in voltage priority mode. When set to MAXimum, INFinity, or to a very large value, the slew rate will be limited by the analog performance of the output circuit.

Front Panel Menu Reference	SCPI Command
Select **Output\Advanced\Slew** Then select Voltage or Current Enter the voltage or current slew rate in the Slew Rate field. Check Max slew rate to program the fastest slew rate.	To set the voltage slew rate to 5 V/s VOLT:SLEW 5 To set the current slew rate to 1 A/s CURR:SLEW 1 To set the fastest slew rate: VOLT:SLEW MAX

(Figure 7 — Keysight SCPI command reference)

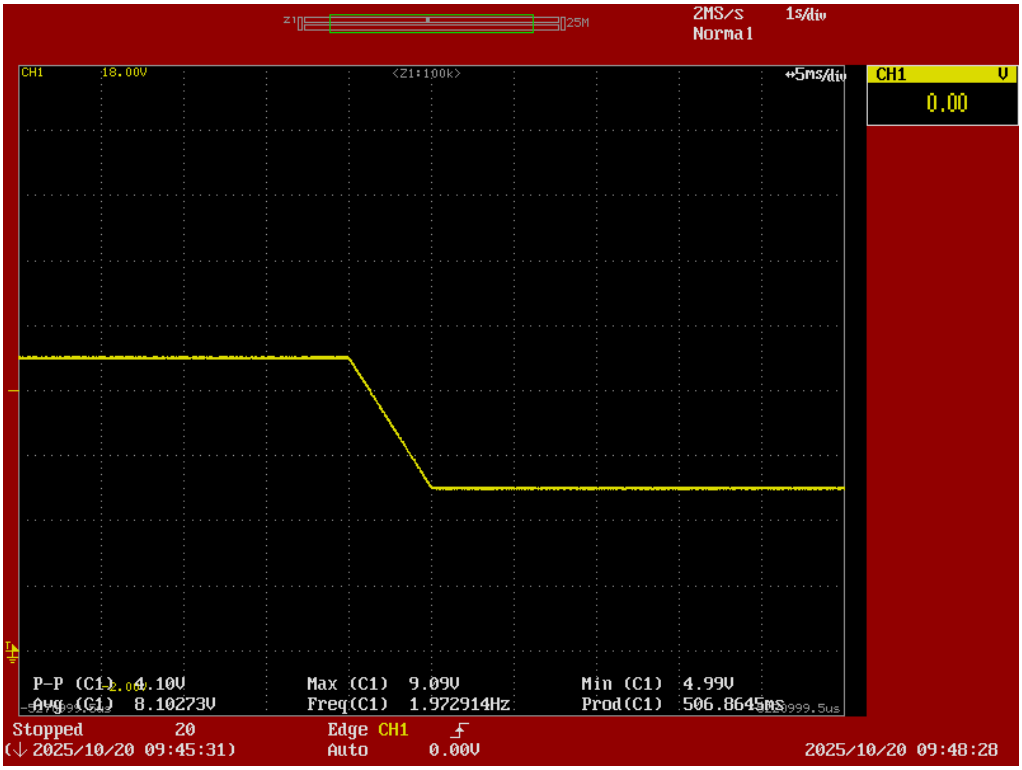
With this information, writing the ramp control script in Python becomes simple:

test.py > ...

```python
import pyvisa, time
ADDR = "USB0::0X069::0X0346::C010101::INSTR"  # VISA address of the instrument
rm = pyvisa.ResourceManager() # Open VISA resource manager
psu = rm.open_resource(ADDR)  # Open communication with the instrument
psu.write("SLEW RATE MAX")  # Set the slew rate to maximum
psu.write("VOLT 9")  # Set the voltage to 9V
psu.write("OUTP ON")  # Turn on the output
psu.write("SLEW RATE 800") # Set the slew rate to 800 V/s
psu.write("VOLT 5")  # Change the voltage to 5V
time.sleep(0.005)  # Wait for 5 milliseconds needed for the voltage ramp
psu.write("OUTP OFF")  # Turn off the output

```

(Figure 8 — Author's own, 2025)



(Figure 9 — Author's own, 2025)

The figure above shows the voltage ramp being executed in exactly 5 ms. As the instrument used is a calibrated precision supply, the commanded transition was achieved within the desired time.

4. Conclusion

The integration between Python, SCPI, and VISA, as discussed in Sections 2.1 and 2.2, establishes a robust foundation for test automation in laboratory and industrial environments. SCPI standardization ensures interoperability between instruments from different manufacturers, while VISA abstracts physical communication layers and enables reliable, multi-interface session management.

In the automotive sector, where standards such as LV124, LV148, and ISO 16750 impose strict requirements on electrical testing applied to electronic devices, the approach demonstrated in Section 4 supports deterministic execution of power supply tests, promoting measurement reproducibility, metrological conformity, and operational efficiency [6].

Importantly, the techniques extend beyond automotive applications. Industrial automation, consumer electronics, aerospace, telecommunications, and biomedical devices benefit from automated methods that improve standardization, reduce operational costs, and mitigate design and production failures.

The use of Python and libraries such as PyVISA enables complete test flows, structured data acquisition, automated analytics, and integrated reporting aligned with Industry 4.0 and digital transformation initiatives.

Future work may include multi-instrument synchronization through triggering, centralized database integration for result management, and advanced analytics techniques such as anomaly detection and predictive modeling to enhance system reliability [7].

Thus, the combination of Python, SCPI, and VISA represents a versatile, scalable, and industry-oriented strategy for modern electronic validation, contributing significantly to the advancement of testing and certification processes.

Official Portuguese Translation

Resumo

Este trabalho apresenta uma abordagem prática para automação de instrumentos utilizando Python, comandos SCPI e o padrão de comunicação VISA. A metodologia possibilita a execução determinística e repetível de testes elétricos voltados à validação industrial e a ambientes de pesquisa. Uma fonte de alimentação programável é controlada remotamente para realizar uma rampa rápida de tensão, tipicamente aplicada em procedimentos de validação de módulos, garantindo elevados níveis de rastreabilidade e confiabilidade metrológica. A solução proposta reduz a dependência do operador, permite maior flexibilidade no desenvolvimento de testes e viabiliza integração simplificada com sistemas de aquisição de dados e geração automatizada de relatórios. A abordagem está alinhada com tendências modernas de digitalização industrial, especialmente aquelas associadas a estratégias inteligentes de ensaio e garantia da qualidade.

Palavras-chave

SCPI; VISA; automação; validação de potência; fonte de alimentação programável; Python; PyVISA; testes elétricos; sistemas de medição.

Sumário

1.**Introdução**

A automação de instrumentos é uma prática central nos setores de Pesquisa e Desenvolvimento (P&D) e validação, pois viabiliza a execução precisa e repetitiva de ensaios elétricos conforme normas técnicas e procedimentos padronizados. Na indústria automotiva, essa abordagem é amplamente empregada em testes de ECUs, módulos de potência, sensores e atuadores, suportando atividades como durabilidade, avaliação de conformidade normativa (ISO, IEC e padrões automotivos específicos) e caracterização funcional de dispositivos.

Ao integrar software e instrumentos de bancada — como fontes programáveis, multímetros e sistemas de aquisição de dados (DAQs) — reduzimos erros humanos, aumentamos a reprodutibilidade experimental e garantimos a rastreabilidade dos resultados, requisitos essenciais para qualificação técnica e auditorias. Este documento apresenta uma aplicação prática em Python para controle de uma fonte de alimentação programável via VISA e comandos SCPI, culminando em um teste de rampa típico de validação em ambiente industrial automotivo.

2.**Conceitos Fundamentais**

2.1 SCPI (Standard Commands for Programmable Instruments)

O SCPI (Standard Commands for Programmable Instruments) é um padrão de comandos textuais desenvolvido em 1990 para padronizar o controle remoto de instrumentos eletrônicos. Sua principal contribuição é garantir que diferentes equipamentos — mesmo de fabricantes distintos — compartilhem uma sintaxe comum para configuração, medição e consulta de dados.

Ele opera sobre protocolos de comunicação como:

- **USBTMC** (USB Test & Measurement Class)
- **LAN (TCP/IP)**
- **GPIB**
- **RS-232**

Ao abstrair a camada física, o SCPI assegura **interoperabilidade**, **portabilidade de software** e **redução de tempo de desenvolvimento**.

Todo comando SCPI é formado por **cabeçalhos** estruturados em uma **árvore hierárquica** de funções instrumentais:

SUBSISTEMA:FUNÇÃO PARÂMETRO

Exemplos:

Comando SCPI	**Função**
VOLT 12.0	Define tensão de saída em 12 V
CURR 1.5	Ajusta limite de corrente para 1,5 A
OUTP ON	Liga a saída
MEAS:VOLT?	Solicita medição de tensão

Os comandos aceitam **abreviações em maiúsculas**:

Forma completa	**Forma abreviada válida**
VOLTAGE	VOLT
CURRENT	CURR
OUTPUT	OUTP
MEASURE:VOLTAGE?	MEAS:VOLT?

Existem dois tipos principais de instruções SCPI:

Tipo	Símbolo	Exemplo	Resultado
Comando	Sem ?	OUTP ON	Executa ação
Query	Com ?	MEAS:CURR?	Retorna leitura do instrumento

Queries são essenciais para **feedback e rastreabilidade** em validação.

Estados e Buffers Internos do Instrumento

O SCPI permite:

- Gerenciar erros internos
- Consultar event registers
- Sincronizar sequências de teste

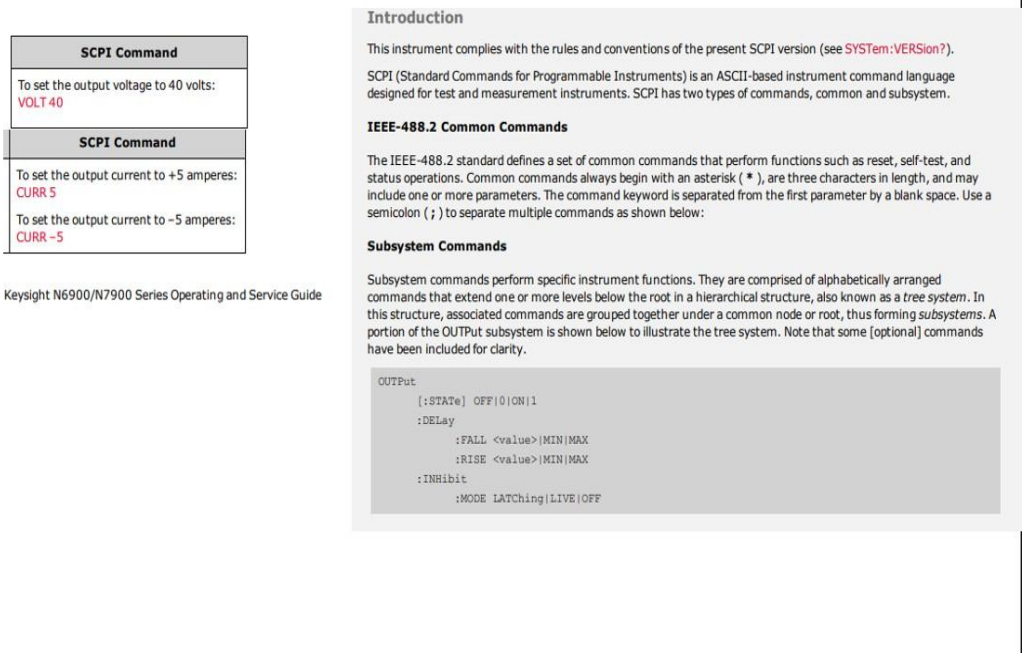
Ex.:

SYST:ERR?

*OPC?

Esses recursos são essenciais em **automação confiável**.

Para teste prático nesse material, utilizaremos um fonte da Keysight, então é necessário que se leia o manual de operação do equipamento utilizado, vide exemplo abaixo.



SCPI Command
To set the output voltage to 40 volts: VOLT 40

SCPI Command
To set the output current to +5 amperes: CURR 5 To set the output current to -5 amperes: CURR -5

Keysight N6900/N7900 Series Operating and Service Guide

Introduction

This instrument complies with the rules and conventions of the present SCPI version (see SYSTem:VERSion?).

SCPI (Standard Commands for Programmable Instruments) is an ASCII-based instrument command language designed for test and measurement instruments. SCPI has two types of commands, common and subsystem.

IEEE-488.2 Common Commands

The IEEE-488.2 standard defines a set of common commands that perform functions such as reset, self-test, and status operations. Common commands always begin with an asterisk (*), are three characters in length, and may include one or more parameters. The command keyword is separated from the first parameter by a blank space. Use a semicolon (;) to separate multiple commands as shown below:

Subsystem Commands

Subsystem commands perform specific instrument functions. They are comprised of alphabetically arranged commands that extend one or more levels below the root in a hierarchical structure, also known as a *tree system*. In this structure, associated commands are grouped together under a common node or root, thus forming *subsystems*. A portion of the OUTPut subsystem is shown below to illustrate the tree system. Note that some [optional] commands have been included for clarity.

```
OUTPut
      [:STATe] OFF|0|ON|1
      :DELay
            :FALL <value>|MIN|MAX
            :RISE <value>|MIN|MAX
      :INHibit
            :MODE LATChing|LIVE|OFF
```

\

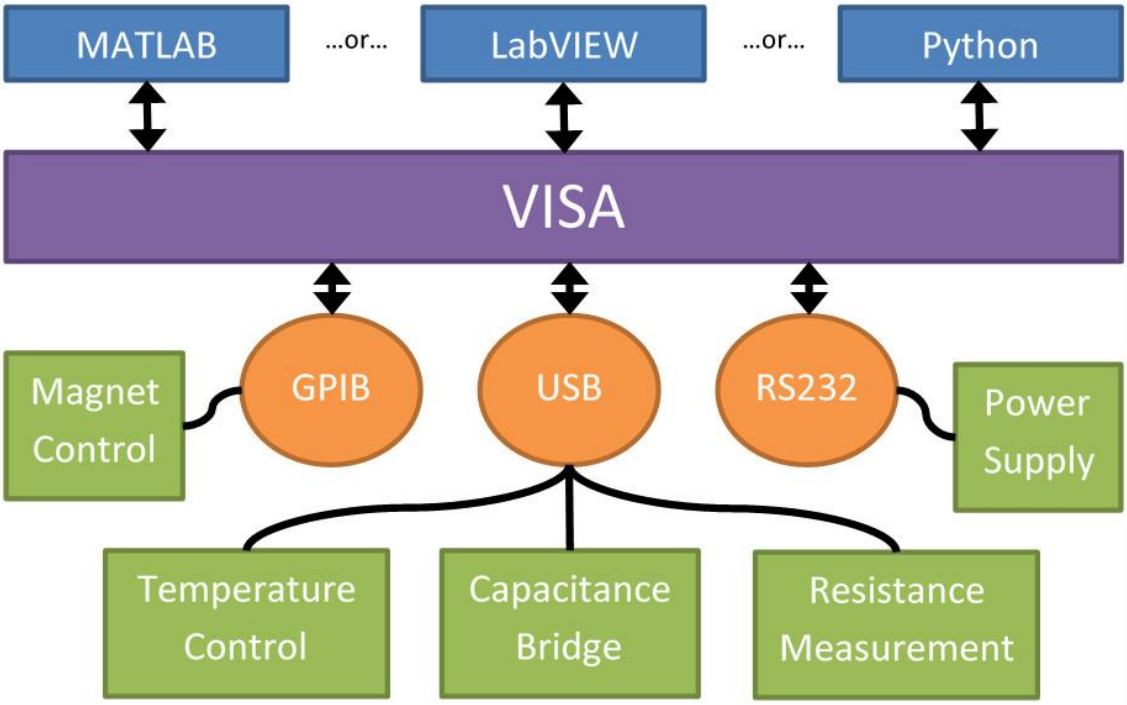
Fonte: Keysight N6900/N7900 Series Operating and Service Guide

2.2 VISA (Virtual Instrument Software Architecture) [3–4]

O VISA (Virtual Instrument Software Architecture) é um padrão definido pela especificação **VPP-4.3** do consórcio **VXIplug&play Systems Alliance**, criado para fornecer uma **interface de software padronizada** para comunicação com instrumentos de teste e medição. Sua principal função é **abstrair o meio físico de comunicação**, permitindo que uma mesma aplicação controle diversos equipamentos independentemente da interface utilizada.

Os principais protocolos suportados incluem:

- **USB (USBTMC — USB Test & Measurement Class)**
- **LAN (TCP/IP — sockets e VXI-11/HiSLIP)**
- **GPIB (IEEE-488)**
- **RS-232 (Serial)**



Fonte: Razorbilli instruments, Guide to data acquisition in the modern laboratory

Graças a essa arquitetura, ***o código da aplicação não precisa ser alterado*** *quando há mudanças no tipo de conexão ou modelo do instrumento.*

Recursos essenciais do VISA

Recurso	***Função***
Descoberta de instrumentos	*Lista todos os equipamentos conectados*
Gerenciamento de sessão	*Abre/fecha conexões e controla buffers*
Transmissão de comandos	*Envio de SCPI com confirmação de entrega*
Leitura de dados e eventos	*Respostas a queries e interrupções*
Tratamento de erros	*Diagnóstico confiável em automação*

Esses mecanismos garantem confiabilidade, rastreabilidade e sincronização — requisitos críticos em validação automotiva e industrial.

2.3 VISA no Python: PyVISA

No ambiente Python, a biblioteca PyVISA implementa a API VISA de forma multiplataforma, permitindo:

- Comunicação direta via USB/GPIB/LAN
- Acesso a comandos SCPI para controle funcional
- Integração com bancos de dados e automação de testes

Ideal para ambientes de Qualificação, Ensaios de Durabilidade e Homologação Automotiva.

2.4 Configuração do Ambiente Python

Instale as bibliotecas para comunicação com instrumentos:

pip install pyvisa pyvisa-py

Opcionalmente, instale drivers do fabricante (NI-VISA ou Keysight IO Libraries Suite). Para listar os dispositivos conectados e iniciar a comunicação:

teste.py > ...

```python
# Lista todos os instrumentos conectados via PyVISA
import pyvisa

rm = pyvisa.ResourceManager()
recursos = rm.list_resources()
print("Instrumentos encontrados:", recursos)
```

(Figura 1, Fonte: Autoria própria, 2025)

3.Teste de Fonte e Aplicação Prática

Exemplo básico de controle de uma fonte programável via USB:

(Fonte usada para esse documento foi uma fonte Keysight) [1]

```python
# Abre a fonte pelo endereço VISA, e coloca a corrente como 15A
import pyvisa
import time
ADDR = "USB0::0x0699::0x0346::C010101::INSTR"  # Endereço VISA da fonte
rm = pyvisa.ResourceManager()  # Abre o gerenciador de recursos VISA
psu = rm.open_resource(ADDR, timeout=5000)  # Abre a fonte pelo endereço VISA
psu.write("CURR 15")  # Configura a corrente para 15A
time.sleep(10)  # Espera 10 segundos
psu.write("*RST")  # Reseta a fonte
```

(Figura 2, Fonte: Autoria própria, 2025.)

No exemplo acima, é um teste prático simples para testar a comunicação da fonte.

Colocando o valor de corrente para 15 A, conseguimos ver no visor da tela da fonte, se esse comando foi aceito e comunicado de maneira correta.

(Figura 4, Fonte: Autoria própria, 2025.)

Tendo em vista que a fonte está comunicando corretamente, já podemos utilizar esse o Python para elaboração de testes, ondas, tudo de forma programática e utilizando somente a fonte e a comunicação SCPI. No exemplo abaixo, iremos fazer uma rampa rápida, de 9 V, para 5 V em 5ms.

Isso pode ser feito de várias formas, porém uma maneira simples de fazer, é ajustar o SLEW RATE da fonte. O slew rate é a velocidade com que a tensão (ou corrente) muda com o tempo.Em outras palavras, ele mostra o quão rápido um sinal consegue subir ou descer.

É medido normalmente em volts por segundo (V/s), ou em unidades menores como V/ms ou V/µs.

Iremos fazer o cálculo necessário para essa rampa.

$$\triangle SR = \frac{\Delta V}{\Delta T}$$

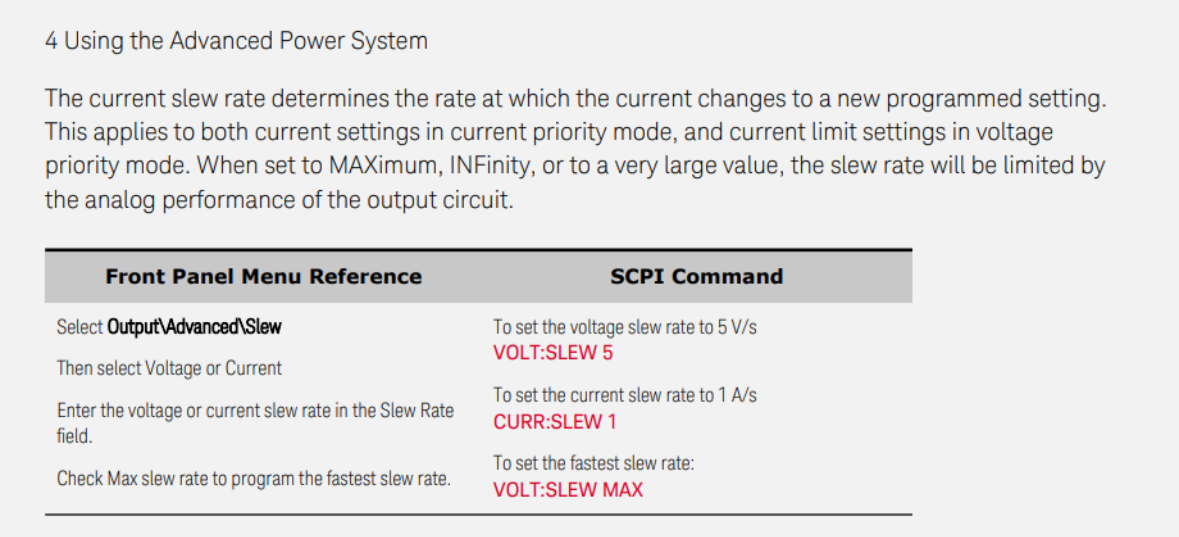
Figura 5, Formula Slew Rate)

Sabendo que a variação que queremos é de 9 V para 5 V, e o tempo necessário para essa rampa é de 5m/s. Basta fazermos essa simples conta.

$$\triangle SR = \frac{(9 - 5)}{0.005} = 800\ V/s$$

Figura 6, (Calculo Slew Rate)

Também é necessário ver o manual da fonte que vai ser utilizada, para saber qual é o comando SCPI correspondente,e se a fonte suporta essa taxa de variação.



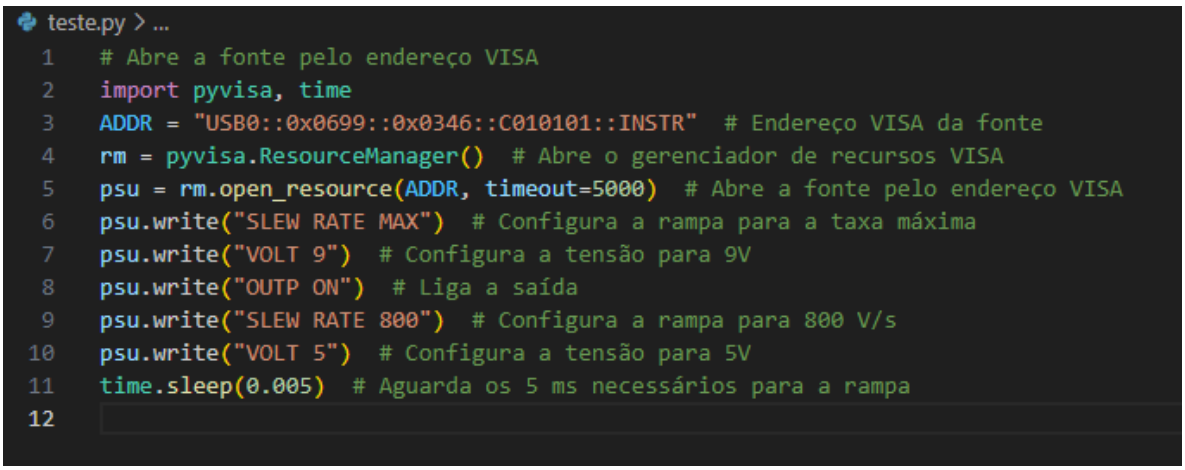
4 Using the Advanced Power System

The current slew rate determines the rate at which the current changes to a new programmed setting. This applies to both current settings in current priority mode, and current limit settings in voltage priority mode. When set to MAXimum, INFinity, or to a very large value, the slew rate will be limited by the analog performance of the output circuit.

Front Panel Menu Reference	SCPI Command
Select **Output\Advanced\Slew** Then select Voltage or Current Enter the voltage or current slew rate in the Slew Rate field. Check Max slew rate to program the fastest slew rate.	To set the voltage slew rate to 5 V/s VOLT:SLEW 5 To set the current slew rate to 1 A/s CURR:SLEW 1 To set the fastest slew rate: VOLT:SLEW MAX

Figura 7(Comando Manual Keysight)

Tendo essas informações, fica fácil de fazer essa rampa com Python.



teste.py > ...

```python
# Abre a fonte pelo endereço VISA
import pyvisa, time
ADDR = "USB0::0x0699::0x0346::C010101::INSTR"  # Endereço VISA da fonte
rm = pyvisa.ResourceManager()  # Abre o gerenciador de recursos VISA
psu = rm.open_resource(ADDR, timeout=5000)  # Abre a fonte pelo endereço VISA
psu.write("SLEW RATE MAX")  # Configura a rampa para a taxa máxima
psu.write("VOLT 9")  # Configura a tensão para 9V
psu.write("OUTP ON")  # Liga a saída
psu.write("SLEW RATE 800")  # Configura a rampa para 800 V/s
psu.write("VOLT 5")  # Configura a tensão para 5V
time.sleep(0.005)  # Aguarda os 5 ms necessários para a rampa

```

Figura 8(Fonte: Autoria própria, 2025)

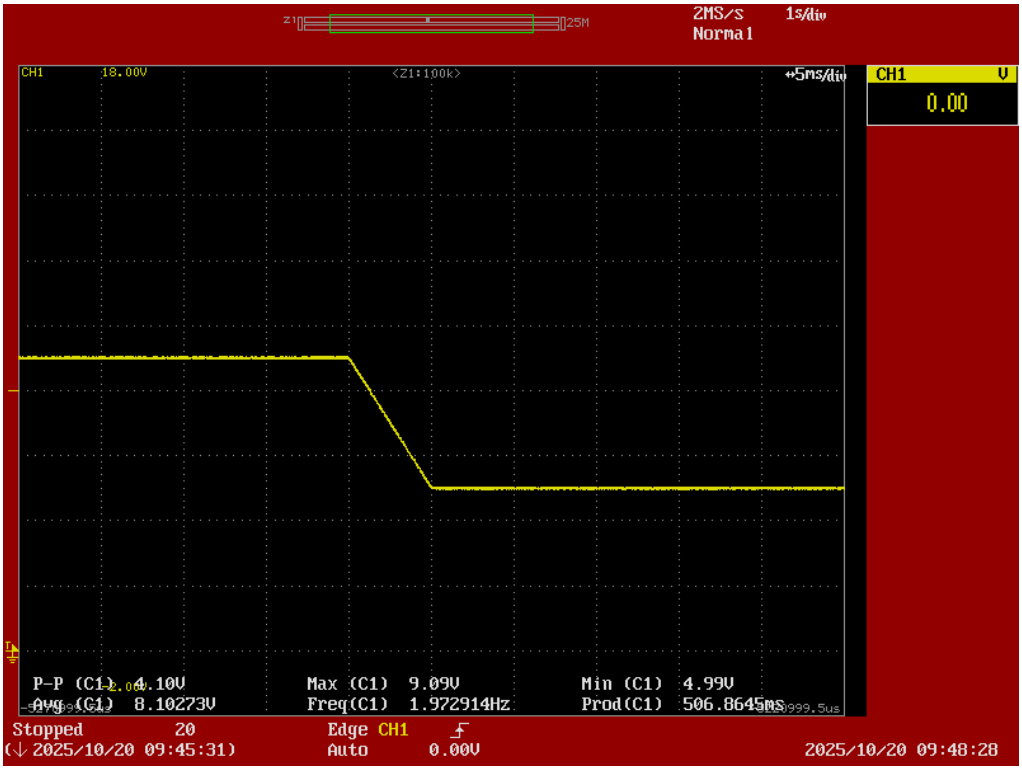


Figura 9 Fonte: Autoria própria, 2025)

Na imagem acima, é possível ver a rampa sendo feita nos exatos 5 m/s, como essa fonte utilizada é um instrumento de precisão, e está dentro do seu periodo de calibração, ele fez a rampa no tempo desejado.

4.Conclusão

A integração entre Python, SCPI e VISA, conforme discutido nas Seções 2.1 e 2.2, estabelece uma fundamentação robusta para automação de testes em ambientes laboratoriais e industriais. A padronização dos comandos SCPI assegura interoperabilidade entre instrumentos de diferentes fabricantes, enquanto o VISA oferece abstração das camadas de comunicação física e mecanismos de gerência de sessão, possibilitando controle confiável, multiplataforma e com elevada rastreabilidade.

No setor automotivo, no qual requisitos como LV124, LV148 e ISO 16750 impõem rigor nos ensaios elétricos aplicados a dispositivos eletrônicos, a metodologia explorada na Seção 4, voltada ao controle programático de fontes de alimentação e instrumentação associada, promove a execução de testes de forma

determinística, repetitiva e alinhada às exigências normativas. Tal abordagem contribui diretamente para a redução de variabilidade experimental, para a conformidade metrológica e para a eficiência geral dos processos de qualificação e validação. [6]

Importa destacar que as técnicas abordadas estendem-se para além da indústria automotiva. Setores como automação industrial, eletrônica de consumo, aeroespacial, telecomunicações e dispositivos biomédicos se beneficiam de métodos automatizados que permitem padronização de resultados, redução de custos operacionais e mitigação de falhas durante o desenvolvimento de produtos.

A utilização de Python e bibliotecas como PyVISA possibilita, adicionalmente, a construção de fluxos completos de teste, aquisição estruturada de dados, análise automatizada e geração de relatórios técnicos integrados, em consonância com os princípios da Indústria 4.0 e com iniciativas de transformação digital nos processos de engenharia.

Como continuidade deste trabalho, propõe-se a expansão da automação para cenários multi-instrumento com sincronização via triggering, a integração com bancos de dados corporativos para gestão centralizada dos resultados e a adoção de técnicas de análise avançada, como detecção automatizada de anomalias e modelos preditivos para aumento da confiabilidade e disponibilidade dos sistemas sob teste. [7]

Dessa forma, conclui-se que a combinação de Python, SCPI e VISA representa uma estratégia versátil, escalável e aderente às necessidades atuais de validação eletrônica, constituindo um recurso transversal a diversos segmentos da indústria moderna e contribuindo significativamente para o avanço dos processos de teste e certificação.

REFERENCES/REFERÊNCIAS

ACEA – European Automobile Manufacturers' Association. LV124: Test Requirements for Electrical and Electronic Components in Road Vehicles. Bruxelas: ACEA, 2016.

IVI FOUNDATION. Standard Commands for Programmable Instruments – SCPI Specification Manual. 1999. Disponível em: https://www.ivifoundation.org/scpi. Acesso em: 26 out. 2025.

KEYSIGHT TECHNOLOGIES. N6900/N7900 Series Operating and Service Guide. Santa Rosa: Keysight Technologies, 2017. Disponível em: https://www.keysight.com. Acesso em: 26 out. 2025.

NATIONAL INSTRUMENTS. VPP-4.3: VISA Specification – Virtual Instrument Software Architecture. Austin: National Instruments, 2004. Disponível em: https://www.ni.com/visa. Acesso em: 26 out. 2025.

PYVISA PROJECT. PyVISA Documentation: Python VISA Library User Guide. 2024. Disponível em: https://pyvisa.readthedocs.io. Acesso em: 26 out. 2025.

ROHDE & SCHWARZ. SCPI Command Reference for Programmable Instruments. Munique: Rohde & Schwarz GmbH & Co. KG, 2017. Disponível em: https://www.rohde-schwarz.com. Acesso em: 26 out. 2025.

WANG, Y.; ZHANG, F. Research on SCPI-Based Automatic Test System for Electronic Modules. IEEE Transactions on Instrumentation and Measurement, v. 68, n. 5, p. 1434–1442, 2019. DOI: 10.1109/TIM.2018.2873584.