LibreVNA Vector Network Analyzer

Function & Performance Validation Test Assignment

(R&D Validation Purpose — Not a Tutorial Document)

Assignment Date	___/___/______
Assigned To	
Expected Completion	
Purpose	Pre-validation data for vendor collaboration project

1. Background & Objectives

The lab has recently acquired a LibreVNA Vector Network Analyzer (VNA) as evaluation equipment for a vendor collaboration project. The core objective of this assignment is to conduct a comprehensive functional verification and performance characterization of the device from an R&D perspective, producing objective test data and an evaluation report to support subsequent discussions on specifications, optimization directions, and system integration feasibility.

Positioning Note: This is an R&D validation task. The report audience is lab members and vendor technical staff. Content should emphasize precise recording of test conditions, objective data presentation, and analysis of device capability boundaries.

2. Background Knowledge: What Are VNA and S-Parameters?

Before proceeding to the device overview and test tasks, this section briefly explains the core concepts involved. If you already have relevant background, you may skim through and skip to the next section.

2.1 What Is a VNA (Vector Network Analyzer)?

VNA stands for Vector Network Analyzer, one of the most important measurement instruments in RF (Radio Frequency) and microwave engineering. Its core function is to send a known RF signal to a Device Under Test (DUT) across a specified frequency range, then precisely measure both the reflected and transmitted signals, capturing both amplitude (magnitude) and phase information simultaneously.

Why "vector"? Because it measures not only "how strong the signal is" (scalar) but also "how much the signal's phase has shifted." With both amplitude and phase, you obtain complete complex-valued information, enabling impedance analysis on a Smith Chart, Time Domain Reflectometry (TDR), or precise equivalent circuit modeling. In contrast, a Scalar Network Analyzer (SNA) measures only amplitude and cannot provide complete information.

Simple Analogy: Think of an RF circuit as a plumbing system. A VNA is like a precision flow meter that simultaneously measures "how much water bounces back from the pipe inlet" and "how much water successfully passes through," and it can even distinguish the "rhythm" (phase) of the water flow.

Common applications include: antenna design (verifying an antenna works at target frequencies), filter characterization (passband and stopband performance), impedance matching analysis, cable quality testing, and PCB trace characteristic impedance verification.

2.2 What Are S-Parameters (Scattering Parameters)?

S-parameters (Scattering Parameters) are the primary measurement output of a VNA. For an RF component with multiple ports, S-parameters describe the reflection and transmission behavior observed at each port when a signal is injected into a given port.

The naming convention is Sxy, where x is the "observing (receiving) port" and y is the "signal input port." For a 2-port VNA like the LibreVNA, there are four S-parameters:

Parameter	Physical Meaning	Plain Explanation
S11	Port 1 Reflection Coefficient	How much of the signal sent into Port 1 is reflected back to Port 1. Reflects the impedance match quality at Port 1. Lower values (more negative dB) indicate better matching and less reflection.
S21	Port 1→2 Transmission Coefficient	How much of the signal sent into Port 1 is successfully transmitted to Port 2. Reflects insertion loss or gain of the DUT.
S12	Port 2→1 Transmission Coefficient	The proportion of signal sent into Port 2 that is transmitted to Port 1. Used to measure reverse isolation.
S22	Port 2 Reflection Coefficient	How much of the signal sent into Port 2 is reflected back. Similar to S11 but measured from the other end.

S11 and Its Relevance to This Test

In this test, we primarily focus on S11 measurements. When you connect an antenna to the VNA's Port 1, S11 tells you at each frequency how much energy the antenna "absorbs" (i.e., how much is reflected). A well-designed antenna will show a noticeable dip in S11 near its operating frequency (called a resonance dip), typically targeting S11 < -10 dB (meaning less than 10% of power is reflected).

S11 is usually expressed in dB using the formula: S11 (dB) = $20 \times \log_{10}(|S11|)$, where |S11| is the reflection coefficient magnitude (between 0 and 1). For example: |S11| = 0.1 corresponds to -20 dB (only 1% power reflected, good match); |S11| = 0.5 corresponds to -6 dB (25% power reflected, poor match).

2.3 Key Terminology Quick Reference

Term	Description
SOLT Calibration	Short-Open-Load-Through calibration method. Before measurement, known calibration standards are connected to VNA ports so the VNA can mathematically remove systematic errors (cable loss, connector imperfections, etc.), improving measurement accuracy.
IFBW	Intermediate Frequency Bandwidth. The VNA internally down-converts RF signals for digitization; IFBW determines the IF filter bandwidth. Narrower = lower noise (greater dynamic range) but slower sweep; wider = faster but noisier. This is the core trade-off to be quantified in this test.
Sweep	The process where the VNA sequentially changes the excitation signal frequency and measures S-parameters at each frequency point. One complete frequency scan is called a sweep.
Sweep Time	Time required to complete one sweep. Update rate = 1 / sweep time.
Span	The width of the frequency sweep range. E.g., center frequency 2.44 GHz with 20 MHz span means sweeping 2.43–2.45 GHz.
Dynamic Range	The difference between the largest and smallest signals the VNA can measure (in dB). Greater dynamic range means weaker signals can be measured.
DUT	Device Under Test. In this test, either a 50 Ω load or an antenna.
SMA Connector	SubMiniature version A, a common RF coaxial connector usable up to 18 GHz, 50 Ω impedance.
Return Loss	Equals -S11 (dB). S11 = -20 dB means return loss = 20 dB. Higher values indicate better match.
SCPI	Standard Commands for Programmable Instruments. A standard text command protocol for programmatic instrument control. E.g., ":VNA:FREQuency:START 100000" sets the start frequency to 100 kHz.

3. Device Overview: LibreVNA

LibreVNA is an open-source USB Vector Network Analyzer designed by Jan Käberich (jankae), with hardware and firmware published on GitHub. It is primarily used to measure S-parameters of RF circuits and antennas.

Item	Specification
Frequency Range	100 kHz – 6 GHz
Measurement Ports	2 × 50 Ω SMA (female), full S-parameters (S11/S21/S12/S22)
Max Points	2 – 4,501 points
Sweep Speed (2-port, 4000 pts)	< 500 ms (IFBW = 50 kHz)
IFBW Range	10 Hz – 50 kHz
Dynamic Range (IFBW 10 Hz)	> 95 dB (to 3 GHz); > 50 dB (to 6 GHz)
Frequency Accuracy	< 2 ppm
ADC	16-bit, 3 synchronous ADCs
Signal Processing	Spartan-6 FPGA real-time processing
Interface	USB Type-C (USB 2.0 Full Speed); 5 V / max 1.2 A
External Reference	10 MHz input/output (SMA)
Control	LibreVNA-GUI (Win/Linux/macOS) + TCP SCPI (default Port 19542)

Known Limitations: Performance degrades above 3 GHz (port isolation drops to 50 dB@6 GHz); dynamic range drops to 70 dB@100 kHz below 1 MHz; S12 isolation ~10 dB worse than S21 due to layout. Max RF port input power +10 dBm — exceeding this may cause damage.

4. Task Description

Part 1: Basic Installation & Operation Verification

Objective: Confirm the instrument hardware has no obvious anomalies and establish a valid calibration baseline.

1. **Software Installation & Connection:** Install LibreVNA-GUI on the lab computer (download the latest stable release from GitHub Releases). On Linux, additionally install the udev rules file (see Appendix A). Confirm USB connection is working and enter VNA mode. Record the GUI version and device firmware version.
2. **SOLT Calibration:** Perform a full SOLT calibration on Port 1 using the included SMA calibration kit (Short, Open, Load, Through). The calibration frequency range should cover subsequent test requirements. Export the calibration file with date and conditions noted.
3. **Calibration Verification:** Connect a 50 Ω load to Port 1 and confirm S11 return loss > 30 dB across the calibrated frequency range. Capture a screenshot as documentation.

Part 2: Sweep Speed & Measurement Quality Testing

Objective: Quantify the trade-off between sweep speed and measurement quality, producing objective data for vendor discussion.

(A) Sweep Speed Baseline Test

Baseline conditions: S11 only, 20 MHz span, 300 points, IFBW 50 kHz, target update rate ≥ 25 Hz. Record at least 30 consecutive sweep times, computing mean, standard deviation, and extremes.

(B) IFBW Parameter Sweep

Using the same span/points, set IFBW to 50 kHz, 10 kHz, and 1 kHz respectively, recording sweep time, equivalent update rate, S11 noise floor, and trace jitter.

(C) DUT Measurement Validation

- **50 Ω Load:** Observe noise floor behavior under different IFBW settings.
- **Known-band Antenna (record model and nominal band):** Observe resonance dip depth, position, and trace jitter.

Part 3: Python Automated Data Acquisition

Objective: Develop a reusable Python measurement script template that can later be ported to Raspberry Pi or other embedded platforms.

Complete technical details for this part (system architecture, SCPI command reference, full example code, troubleshooting) are in Appendices B–E. Below are the core requirement specifications only:

4. Control LibreVNA via SCPI protocol (TCP socket connection to LibreVNA-GUI's SCPI server, default `localhost:19542`)
5. Configure measurement parameters: S11, specified center frequency, 20 MHz span, 300 points, IFBW 50 kHz
6. Trigger sweep and use `:VNA:ACQ:FIN?` polling to confirm completion (do not use fixed delays)
7. Read back full frequency axis and S11 data (dB), save as CSV (columns: Frequency_Hz, S11_dB), filename includes timestamp
8. Code should have clear function decomposition, type hints, and comments

⚠ LibreVNA's SCPI control requires the LibreVNA-GUI background process. Even if the graphical interface is not needed, the GUI must be launched with the --no-gui argument. See Appendix B for architecture details.

5. Expected Deliverables

- **Validation Test Report:** Including test environment (software/hardware versions, calibration conditions), all test data, representative S11 plots, preliminary assessment and recommendations.

- **Python Source Code:** Complete code with README (including environment requirements and known limitations).
- **Raw Data:** All CSV files and calibration files, archived with meaningful filenames and folder structure.

6. Reference Resources

Resource	Location
Official GitHub	`https://github.com/jankae/LibreVNA`
GUI Release Downloads	`https://github.com/jankae/LibreVNA/releases`
SCPI Programming Guide	`Documentation/UserManual/ProgrammingGuide.pdf`
Python Wrapper libreVNA.py	`Documentation/UserManual/SCPI_Examples/libreVNA.py`
Official Example: retrieve_trace_data.py	`Documentation/UserManual/SCPI_Examples/retrieve_trace_data.py`
User Manual (PDF)	`Documentation/UserManual/manual.pdf`
Measurement Examples	`Documentation/Measurements/Measurements.md`
udev Rules (Linux)	`Software/PC_Application/51-vna.rules`
Community Discussions	`https://github.com/jankae/LibreVNA/discussions`
LibreVNA Mailing List	`https://groups.io/g/LibreVNA`

Appendix A: Linux Installation Steps

To use LibreVNA on Linux, you need to install udev rules to allow non-root users to access the USB device:

A-1 Install udev Rules

```
# Download udev rules
wget https://raw.githubusercontent.com/jankae/LibreVNA/master/\
     Software/PC_Application/51-vna.rules


# Copy to system udev rules directory
sudo cp 51-vna.rules /etc/udev/rules.d/


# Reload udev rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

A-2 Install LibreVNA-GUI

```
# Download from GitHub Releases
# https://github.com/jankae/LibreVNA/releases


chmod +x LibreVNA-GUI


# Normal mode
./LibreVNA-GUI


# Headless mode (for SCPI automation)
./LibreVNA-GUI --no-gui
```

A-3 Verify Connection

After starting the GUI, if the LibreVNA is connected via USB, the GUI should automatically detect and display the device serial number. You can verify via the following SCPI commands:

```
# Test SCPI connection (default port 19542)
telnet localhost 19542


# Query ID
*IDN?
# Expected: LibreVNA,LibreVNA-GUI,<serial>,<version>
```

```
:DEV:CONN?
# Returns device serial or "Not connected"
```

⚠ LibreVNA's USB power draw is up to 1.2 A, which may exceed the capacity of some USB 2.0 ports. If the device is unstable at startup, use external DC power (5 V / 1.5 A). Normal operating case temperature is approximately 60°C.

Appendix B: LibreVNA SCPI Control Architecture

B-1 System Architecture

LibreVNA's programmatic control architecture differs from commercial VNAs (e.g., Keysight, R&S) that use direct VISA connections. The architecture is as follows:

```
┌─────────────────┐
│  Python Script   │  ← Your automation program
│  (TCP Client)    │
└───────┬─────────┘
        │ TCP Socket (localhost:19542)
        ▼
┌─────────────────┐
│ LibreVNA-GUI     │  ← SCPI Server (must be running)
│ (SCPI Server)    │
└───────┬─────────┘
        │ USB (proprietary protocol)
        ▼
┌─────────────────┐
│  LibreVNA HW     │  ← FPGA + MCU + RF Frontend
└─────────────────┘
```

Key Point: The Python script does not communicate directly with the LibreVNA hardware via USB. Instead, it connects via TCP socket to the SCPI server built into LibreVNA-GUI. Even if the graphical interface is not needed, the GUI program must still be running in the background.

B-2 Two Ways to Start the SCPI Server

Method 1: Open the GUI directly; the SCPI server starts automatically.

Method 2 (recommended for automation): Start in headless mode:

```
# Linux / macOS
./LibreVNA-GUI --no-gui &


# Windows
LibreVNA-GUI.exe --no-gui
```

Tip: When launching the GUI from a Python script, use subprocess.Popen() instead of os.system() to avoid blocking the main program. See Appendix D code examples.

B-3 SCPI Communication Characteristics (Differences from Standard SCPI)

LibreVNA's SCPI implementation has some differences from the IEEE 488.2/SCPI-99 standard. Important notes:

- **Every command (including non-query commands) generates a response** (possibly an empty string with newline). Your program must read each command's response before sending the next, otherwise residual data will corrupt subsequent query results.
- ***OPC? does not wait for sweep completion:** It only confirms that the "start sweep" action has completed, not that sweep data is ready. Use :VNA:ACQ:FIN? polling instead.
- **Strict version compatibility:** The libreVNA.py wrapper class must match the GUI version. Mixing versions may cause errors such as *ESR? returning empty values.
- **Leading colon in commands:** It is recommended to start every command with a colon (e.g., :VNA:FREQuency:START) to avoid unexpected behavior from the SCPI parser remembering the previous command's branch.

Appendix C: LibreVNA SCPI Command Reference (For This Test)

Below are the main SCPI commands needed for this test. For the complete command set, refer to the official ProgrammingGuide.pdf.

C-1 System & Connection Commands

Command	Description
`*IDN?`	Query device identification string (format: LibreVNA,LibreVNA-GUI,<serial>,<version>)
`:DEV:CONN?`	Query connected device serial (returns "Not connected" if none)
`:DEV:CONN <serial>`	Connect to specified device serial
`:DEV:MODE VNA`	Switch to VNA mode
`:DEV:MODE?`	Query current operating mode

C-2 Frequency & Measurement Settings

Command	Description
`:VNA:FREQuency:START <freq_Hz>`	Set start frequency (Hz)
`:VNA:FREQuency:STOP <freq_Hz>`	Set stop frequency
`:VNA:FREQuency:CENTer <freq_Hz>`	Set center frequency
`:VNA:FREQuency:SPAN <span_Hz>`	Set frequency span
`:VNA:ACQ:IFBW <bw_Hz>`	Set IF bandwidth (10–50000 Hz)
`:VNA:ACQ:POINTS <N>`	Set number of sweep points (2–4501)
`:VNA:ACQ:AVG <N>`	Set averaging count
`:VNA:STIM:LVL <dBm>`	Set stimulus power level
`:VNA:SWEEP FREQUENCY`	Set to frequency sweep mode

C-3 Sweep Control & Data Retrieval

Command	Description

`:VNA:ACQ:SINGLE`	Trigger single sweep
`:VNA:ACQ:FIN?`	Query sweep completion (TRUE/FALSE)
`:VNA:ACQ:AVGLEV?`	Query current averaging level
`:VNA:TRAC:LIST?`	List available trace names
`:VNA:TRAC:DATA? <trace>`	Read trace data (freq,real,imag CSV)
`:VNA:TRAC:AT? <trace> <freq>`	Query trace data at specific frequency

Data Format: :VNA:TRAC:DATA? S11 returns lines in "frequency,real,imag" format (comma-separated), where real and imag are linear-scale complex S-parameter values. Conversion: S11_dB = 20 * log10(sqrt(real^2 + imag^2)).

Appendix D: Python Example Code

Below is a complete Python automated measurement example that can serve as a development template. This program uses the official libreVNA.py wrapper class.

D-1 Prerequisites

First download libreVNA.py matching your GUI version from the official GitHub:

```
# Download libreVNA.py matching your GUI version
# Path: Documentation/UserManual/SCPI_Examples/libreVNA.py
#
# For v1.5.1:
# https://github.com/jankae/LibreVNA/blob/v1.5.1/
#     Documentation/UserManual/SCPI_Examples/libreVNA.py
#
# Place libreVNA.py in the same directory as your script
```

D-2 Complete Example: Single S11 Measurement & Save

```
#!/usr/bin/env python3
"""
LibreVNA Automated S11 Measurement Example
"""


import math, time, csv
from datetime import datetime
from libreVNA import libreVNA




CENTER_FREQ_HZ = 2_440_000_000
SPAN_HZ        = 20_000_000
NUM_POINTS      = 300
IFBW_HZ         = 50_000
STIM_LEVEL_DBM  = -10




def connect_vna(host="localhost", port=19542) -> libreVNA:
    vna = libreVNA(host, port)
    idn = vna.query("*IDN?")
    print(f"Connected: {idn}")
    dev = vna.query(":DEV:CONN?")
    if "Not connected" in dev:
        raise ConnectionError("LibreVNA hardware not connected")
    print(f"Device serial: {dev}")
    return vna
```

```
def configure_sweep(vna: libreVNA) -> None:
    start = CENTER_FREQ_HZ - SPAN_HZ // 2
    stop  = CENTER_FREQ_HZ + SPAN_HZ // 2
    vna.cmd(":DEV:MODE VNA")
    vna.cmd(":VNA:SWEEP FREQUENCY")
    vna.cmd(f":VNA:FREQuency:START {start}")
    vna.cmd(f":VNA:FREQuency:STOP {stop}")
    vna.cmd(f":VNA:ACQ:IFBW {IFBW_HZ}")
    vna.cmd(f":VNA:ACQ:POINTS {NUM_POINTS}")
    vna.cmd(f":VNA:STIM:LVL {STIM_LEVEL_DBM}")



def trigger_and_wait(vna, timeout_s=10.0,
                     poll_interval=0.05) -> float:
    vna.cmd(":VNA:ACQ:SINGLE")
    t0 = time.perf_counter()
    while True:
        if "TRUE" in vna.query(":VNA:ACQ:FIN?").upper():
            break
        if time.perf_counter() - t0 > timeout_s:
            raise TimeoutError("Sweep timeout")
        time.sleep(poll_interval)
    return time.perf_counter() - t0



def read_s11_data(vna) -> list[tuple[float, float]]:
    raw = vna.query(":VNA:TRAC:DATA? S11")
    data = []
    for line in raw.strip().split("\n"):
        parts = line.split(",")
        if len(parts) >= 3:
            freq = float(parts[0])
            r, i = float(parts[1]), float(parts[2])
            mag = math.sqrt(r**2 + i**2)
            db = 20*math.log10(mag) if mag > 0 else -999
            data.append((freq, db))
    return data



def save_csv(data, prefix="S11") -> str:
    ts = datetime.now().strftime("%Y%m%d_%H%M%S")
    fn = f"{prefix}_{ts}.csv"
    with open(fn, "w", newline="") as f:
        w = csv.writer(f)
        w.writerow(["Frequency_Hz", "S11_dB"])
        for freq, db in data:
            w.writerow([f"{freq:.0f}", f"{db:.4f}"])
    print(f"Saved: {fn}")
    return fn
```

```
def main():
    vna = connect_vna()
    configure_sweep(vna)
    st = trigger_and_wait(vna)
    print(f"Sweep: {st:.4f}s ({1/st:.1f} Hz)")
    data = read_s11_data(vna)
    save_csv(data)
    print(f"{len(data)} points, "
          f"{data[0][0]/1e6:.2f}-{data[-1][0]/1e6:.2f} MHz")



if __name__ == "__main__":
    main()
```

D-3 Advanced: Batch IFBW Sweep Test Script

```
"""Batch IFBW sweep test."""


IFBW_LIST = [50_000, 10_000, 1_000]
N_REPEATS = 30




def batch_ifbw_test(vna):
    import statistics
    for ifbw in IFBW_LIST:
        vna.cmd(f":VNA:ACQ:IFBW {ifbw}")
        times = []
        for _ in range(N_REPEATS):
            times.append(trigger_and_wait(vna))
        avg = statistics.mean(times)
        std = statistics.stdev(times)
        print(f"\nIFBW={ifbw/1000:.0f}kHz:"
              f" {avg:.4f}s ({1/avg:.1f}Hz)"
              f" std={std:.4f}")
        data = read_s11_data(vna)
        save_csv(data, f"S11_IFBW{ifbw//1000}kHz")
```

Extension Suggestion: This script can be extended with matplotlib plotting to automatically generate S11 curves for each IFBW setting. For Raspberry Pi deployment, use headless matplotlib backend or output PNG directly.

Appendix E: Troubleshooting Guide

E-1 Connection Issues

Symptom	Possible Cause & Solution
Connection refused (port 19542)	LibreVNA-GUI is not running. Start the GUI or launch with --no-gui. Verify TCP port 19542 is not blocked by firewall.
"Not connected" from :DEV:CONN?	GUI is running but hardware not detected. Check USB connection, udev rules (Linux), drivers (Windows). Try re-plugging USB.
Device unstable at startup	Insufficient USB port power (requires 1.2 A). Use external DC power supply or powered USB hub.

E-2 SCPI Communication Issues

Symptom	Possible Cause & Solution
"Expected numeric response from *ESR?"	libreVNA.py version does not match GUI version. Download from the matching Git tag.
Query returns "ERROR"	Command syntax error or missing leading colon. Verify spelling; recommend starting with colon.
Subsequent query results corrupted	Previous command's response not read. Ensure every cmd()/query() response is properly consumed.
Sweep data is empty	Data read before sweep completion. Use :VNA:ACQ:FIN? polling before reading.

E-3 Measurement Quality Issues

Symptom	Possible Cause & Solution
S11 noise floor too high	IFBW set too wide. Reduce IFBW to improve dynamic range (at the cost of slower sweeps).
Resonance dip shallower than expected	Calibration may have drifted. Re-perform SOLT calibration; ensure connections are secure.
Unstable measurements above 3 GHz	Known LibreVNA limitation above 3 GHz. Try increasing averaging or reducing IFBW.

Spikes in sweep trace	Known firmware issue (some versions). Update to latest firmware.

E-4 Firmware Update Notes

⚠ Firmware 1.6.3 has a known Flash CS pin timing issue that may cause subsequent firmware updates to fail. If you have already updated to this version and encounter problems, try cooling or heating the LibreVNA before updating. Otherwise, a JTAG programmer may be needed for recovery. Always check Release Notes before updating.

— End of Document —