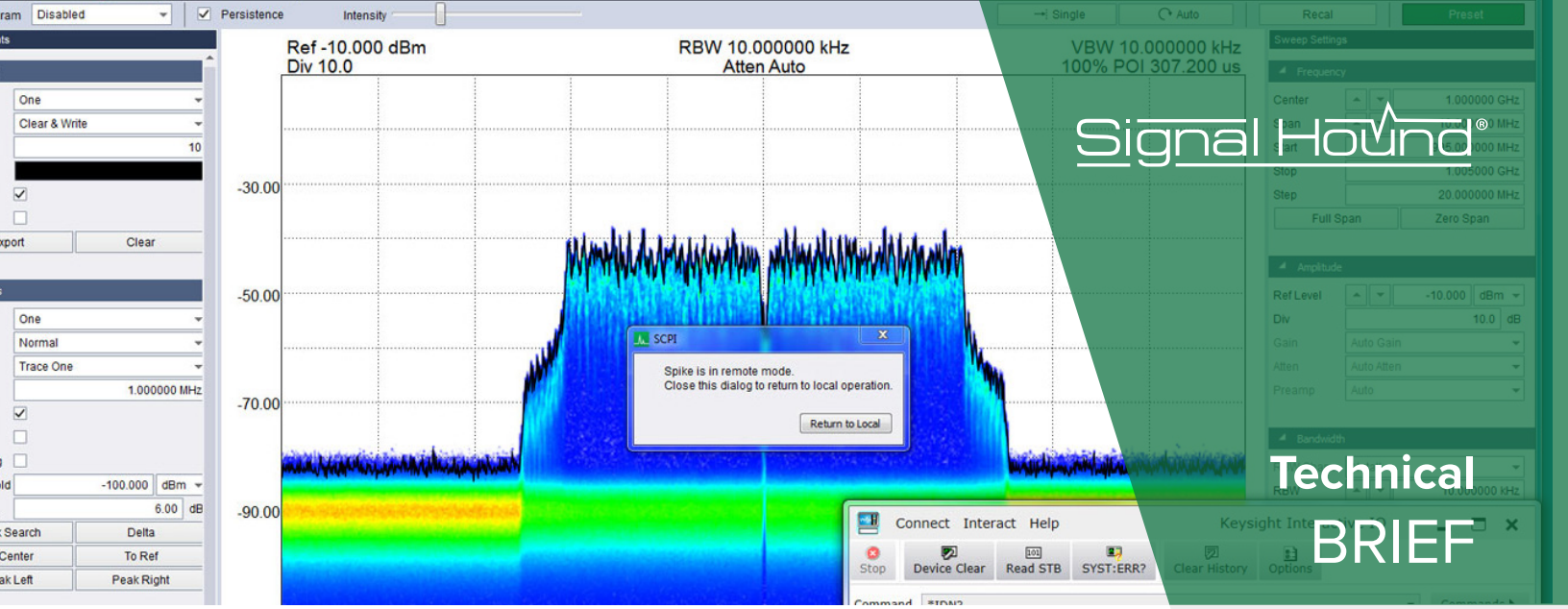

Signal Hound®

Technical BRIEF

SCPI Automation

of Signal Hound Spectrum Analyzers

We often get asked whether we support SCPI programming with our spectrum analyzers.
The short answer is yes!

What is SCPI?

SCPI, which stands for “Standard Commands for Programmable Instruments”, specifies a programming “language” that is designed for controlling test instruments. It provides a common syntax, command structure, and data interchange format which can be used across multiple SCPI compatible test and measurement instruments. Commands are hierarchical using easily understood English based instructions. SCPI commands are sent to instruments over many interfaces, commonly GPIB, VXI, USB, Ethernet, etc. SCPI is intended to give the test system programmer a standardized environment for writing test instrument automation scripts, reducing implementation time by eliminating the need to learn a new proprietary software syntax.

Signal Hound offers remote interface and control capabilities using SCPI compatibility commands for its spectrum analyzers via its Spike software. Our Spike software provides control of all Signal Hound spectrum analyzers using a common Graphical User Interface (GUI) to offer advanced signal analysis measurements and display. They can be remotely operated by sending SCPI commands to Spike through a TCP/IP link. You can connect and interface the Spike software through any VISA implementation or any programming language that allows SOCKET programming.

Setting up a VISA Socket Connection

The Spike software will accept a single network connection in which it can receive SCPI commands and send responses. Instrument control is performed by connecting to the Spike software on a TCP/IP port. On this port, a user can send and receive raw SCPI commands. It is not necessary to use an I/O library like VISA to communicate with the Spike software, but it can simplify several operations. It is possible to communicate directly over the socket with socket programming. The computer that is communicating with the Spike software does not have to be the same computer running the Spike software and does not have to be a Windows platform.

```
    // Phase noise measurements can take awhile, rather than spin a loop waiting for
    //  a non-timeout, lets just increase our timeout value for the measurements.
    // The sweep time is measured in Spike first to determine a reasonable timeout val.
    viSetAttribute(inst, VI_ATTR_TMO_VALUE, 10e3);

    // Setup the traces
    viPrintf(inst, "TRAC:PN:SEL 1; TYPE NORMAL\n");
    viPrintf(inst, "TRAC:PN:SEL 2; TYPE AVERAGE; AVER:COUNT 5\n");

    // Do a single sweep and set it as the reference
    // For instructive purposes only
    int opc;
    viQueryf(inst, "INIT; *OPC?\n", "%d", &opc);
    viPrintf(inst, "TRAC:PN:SEL 1; TO 3\n"); // Store the reference
    viPrintf(inst, "TRAC:PN:SEL 2; CLEAR\n"); // Clear the average trace

    // Do the sweeps, wait for each one to complete
    for(int i = 0; i < averageCount; i++) {
        int opc;
        viQueryf(inst, "INIT; *OPC?\n", "%d", &opc);
    }

    // Reset our timeout time
    viSetAttribute(inst, VI_ATTR_TMO_VALUE, 2e3);

    // Get decade marker tables
    double offsets[5] = {100, 1e3, 10e3, 100e3, 1e6};
    double table[5];

    viPrintf(inst, "CALC:PN:MARK ON\n");

    for(int m = 0; m < 3; m++) {
        viPrintf(inst, "CALCULATE:PNOISE:MARKER:TRACE %d\n", m+1);
        for(int i = 0; i < 5; i++) {
            viPrintf(inst, "CALC:PN:MARK:X %fHz\n", offsets[i]);
            viQueryf(inst, "CALC:PN:MARK:Y?\n", "%lf", table + i);
        }

        // Print it off to the console
        printf("Decade table for trace %d\n", m);
        for(int i = 0; i < 5; i++) {
            printf("%g Offset: %f dBc\n", offsets[i], table[i]);
        }
    }
```

Figure 1—Automated phase noise measurements using C++ and VISA

It is recommended to use a VISA library if available. Several implementations of VISA exist. Commonly used options include Keysight's I/O libraries and NI's VISA libraries. You can also use VISA implementations that exist in other languages/environments such as MATLAB, LabVIEW, and Python.

Connecting to the socket interface using VISA looks like this:

```
viOpen(rm, "TCPIP::localhost::5025:SOCKET", VI_NULL, VI_NULL, &inst);
```

Additionally, when using a VISA library, it is necessary to set the VI_ATTR_TERMCHAR_EN attribute to true. This will terminate the read operation when the termination character is received. The termination character should be set to the newline ('\n') character if it is not set by default. The code for this is:

```
viSetAttribute(inst, VI_ATTR_TERMCHAR_EN, VI_TRUE);
viSetAttribute(inst, VI_ATTR_TERMCHAR, '\n');
```

Only one connection to the Spike software can be active at a time. The connection can be terminated by closing the socket connection, either through the socket library you are using, the viClose function if you are using a VISA library, or by closing your application. Spike will immediately begin waiting for another socket connection when the previous one has ended.

Supported SCPI Commands

Spike's current set of SCPI commands cover the most common spectrum analyzer/receiver functions within the Spike software.

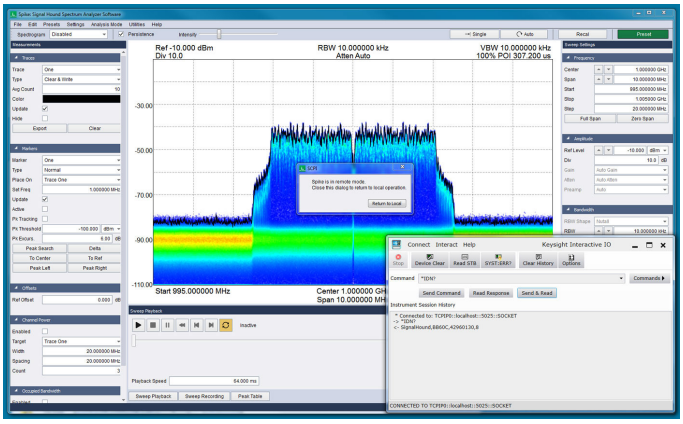

Figure 2—Remote commands can be sent to a device, such as the *IDN? command which identifies the currently connected spectrum analyzer

The table below details which functionality is covered under the current SCPI command set. Additional functionality will be added over time. If the functionality you need is not available, please contact us at aj@signalhound.com to make requests.

Functionality	Supported
Swept Analysis – Sweep Settings	Yes
Swept Analysis – Trace controls	Yes
Swept Analysis – Marker controls	Yes
Swept Analysis – Channel power, occupied bandwidth	Yes
Swept Analysis – Peak table	No
Swept Analysis – Sweep recording/playback	No
Path Loss Tables	No
Limit Lines	Yes
Spectrogram/Waterfall plot controls	No
Persistence display controls	No
Real-Time (Since real-time shares several controls with swept analysis, any functionality provided for swept analysis will be available for real-time measurement mode)	Partial
Zero-Span	No
Harmonic Measurements	Yes
Scalar Network Analysis	Yes
Phase Noise Measurements	Yes
Digital Modulation Analysis	Yes
EMC Precompliance	No
Analog Demodulation	Yes
Interference Hunting	No

Table 1—Current Signal Hound SCPI commands (Fall 2018)

Programming Alternatives

All Signal Hound spectrum analyzers, including the SM200, BB60, SA44, and SA124, can be programmed using three methods (Figure 3). The first two employ SCPI commands via Spike software either locally or remotely over the internet. The third is through fast, direct API programming using a device-specific, local API. API's are available at no cost for all Signal Hound spectrum analyzers.

Since a TCP/IP SOCKET link is used for the SCPI commands, you can control the Spike software from any PC/operating system. For example, a Windows PC runs Spike, but remote control of Spike on the Windows PC can occur on a Linux or Apple system. Traditional programming using Signal Hound-supplied, device-specific API's remains available for fast, direct device control. The device specific API's use a C interface, and the functions can be called from most modern programming languages and environments such as C/C++, C#, Python, Java, LabVIEW, and MATLAB.

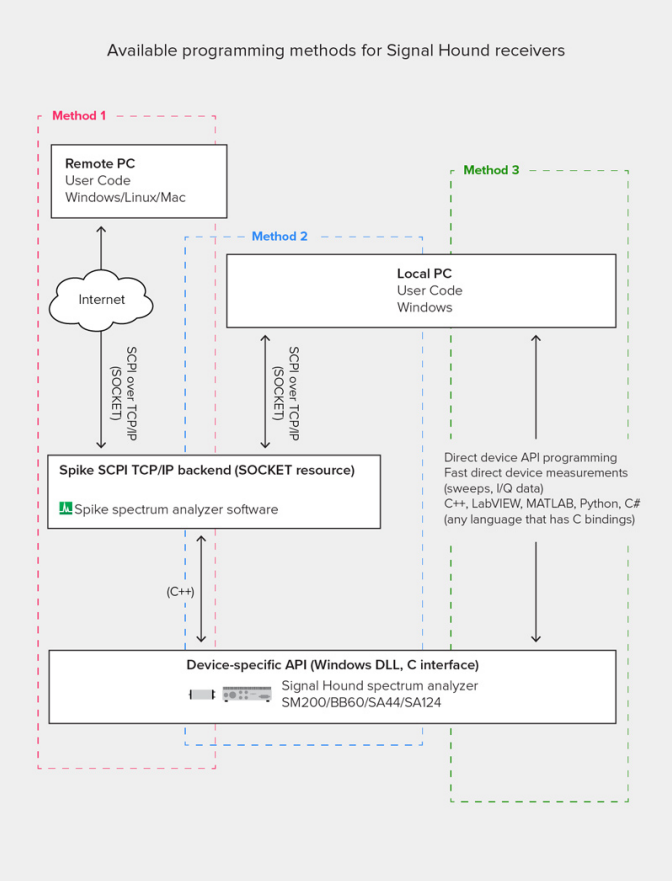

Figure 3—Signal Hound spectrum analyzers can be programmed using three methods: two employ SCPI commands via Spike software either locally or remotely over the internet; the third is through fast, direct API programming using a device-specific, local API.

For More Info

The Signal Hound Software Development Kit (SDK) includes example programs using SCPI to automate the Spike software for several measurement procedures. These programs can provide a basis on which you can build custom programs for automating Signal Hound spectrum analyzers.

The SCPI functionality for Spike is available immediately, at no cost, as part of the Spike software download. The Spike SCPI programming manual and examples are available as a part of the Signal Hound SDK. The manual covers the basics of SCPI commands, how to get started programming the Spike software, and will cover the full SCPI command set implemented by the Spike software. SCPI commands can and will change as the Spike software evolves. It is recommended that when you update Spike in an installation that is controlled via SCPI, you review the version notes and determine if any functionality needs to be updated.

Further Reading

Learn more about Signal Hound's robust, real-time USB-powered spectrum analyzers at signalhound.com/learn.

About Signal Hound

Signal Hound designs and builds powerful, affordable spectrum analyzers and signal generators for engineers and RF professionals around the globe. Whether you're needing EMC precompliance capabilities in a small two-person shop or spectrum monitoring on a national scale, our test equipment is designed with you in mind. Accurate and powerful enough for mission-critical RF analysis, priced at a point accessible to most, and supported by a talented group of engineers committed to what they do – we truly believe that our devices offer unrivaled value in the test equipment industry.

In business since 1996 and selling our own line of Signal Hound test equipment since 2010, we've built the foundation of our company on years of test equipment repair, service, hardware and software development, and manufacturing experience. Signal Hound is a small company with big goals – and an even bigger commitment to providing our customers with an outstanding experience when purchasing and using our products.

Signal Hound®

1502 SE COMMERCE AVE, STE 101 // BATTLE GROUND, WA 98604 // (360) 313-7997 // SIGNALHOUND.COM //