Hints and Tricks for Remote Control of Spectrum and Network Analyzers

Application Note

Products:

- R&S®FSW, FSV, FPS, FSL, FSU, FSQ, FSP
- R&S®ZVA, ZNB, ZNC, ZVL, ZVB
- R&S®FSUP, FSMR

This application note provides hints for implementing remote control programs using Rohde & Schwarz spectrum and network analyzers. The document makes suggestions for improved remote control performance and describes aspects of measurement synchronization in detail.

Finally the document discusses some typical challenges of remote control in production test.

Application Note
Johannes Ganzert
8.2014 - 1EF62_1E

ROHDE&SCHWARZ

Table of Contents

1 Overview 3

2 General Input/Output Considerations 4

2.1 Instrument Drivers 4

2.2 Generic IO Library – VISA 4

3 Optimizing Remote Control Operation 6

3.1 Display Update 6

3.2 Continuous vs. Single Sweep 6

4 Common Issues and Pitfalls 8

4.1 Address Conflicts 8

4.2 Wrong Termination Character During Binary Data Transfer 8

4.3 Disable Auto Serial Poll 8

4.4 Check the Error Queue 8

4.5 Avoiding Timeouts on Long Execution Times 8

4.6 Timeout Occurs in Unexpected Place 9

5 Measurement Synchronization 10

5.1 Synchronization with *WAI 10

5.2 Synchronization with *OPC? 11

5.3 Synchronization with *OPC 12

5.4 Loops with Short Delay 13

6 Service Request (SRQ) Handling 14

7 Production Test Considerations 17

8 Ordering Information 18

1 Overview

This application note describes common aspects of remote control sequences for R&S spectrum and network analyzers. In particular, several synchronization mechanisms are explained.

The document concludes with some hints for production test software, where the test application is designed for maximum throughput and at the same time must be capable of handling defective devices under test.

Programming examples in this document are written in Visual Basic 6 unless noted otherwise.

2 General Input/Output Considerations

There are various approaches to remote control. Test program software can use instrument drivers or choose a lower level approaches by accessing the IO library directly.

2.1 Instrument Drivers

Instrument drivers are available free of charge from the web site http://www.rohde-schwarz.com for common T&M development environments: LabVIEW, LabWindows/CVI, Agilent-VEE, Visual Basic, C++, C#.

Instrument drivers use the VISA library for IO operations.

2.2 Generic IO Library – VISA

If instrument drivers are not available for a specific development environment or if the drivers do not fulfill all user requirements, the client application can perform remote control by means of direct calls to an IO library and use of the SCPI command set of the instrument.

In this case, the use of the standardized VISA library is recommended, because it makes the selections of the actual IO channel almost transparent to the user. Thus test software can be easily migrated from one IO channel to another (e.g. GPIB to LAN).

In the VISA library, the IO functions are the same for all supported hardware interfaces. Only the sequence for opening a particular IO channel varies depending on the chosen bus.

The following sequence opens a connection using GPIB and reads the ID string from the instrument:

```
viOpenDefaultRM(defaultRM)
viOpen(defaultRM, "GPIB::20::INSTR", VI_NULL, 1000, hdl)
viWrite(hdl, "*IDN?", 5, retCount)
response$ = Space$(100)
viRead(hdl, response$, 100, retCount)
```

Now, if the interface is changed to LAN, only the resource string changes in the above code:

```
viOpenDefaultRM(defaultRM)
viOpen(defaultRM, "TCPIP::192.168.1.100::INSTR", VI_NULL, 1000,
hdl)
viWrite(hdl, "*IDN?", 5, retCount)
response$ = Space$(100)
viRead(hdl, response$, 100, retCount)
```

Generic IO Library – VISA

The VISA library is available for a variety of platforms. It can be obtained from Rohde & Schwarz (see order number in the appendix) and several other vendors in the T&M market – Agilent Technologies, National Instruments and others.

3 Optimizing Remote Control Operation

While it is easy to control an instrument remotely, there are some aspects which affect the overall performance and in the end test time. In automated test systems throughput is a key figure. The following paragraphs elaborate on some important details in remote control operation.

3.1 Display Update

Graphical operations consume significant instrument computing resources. Therefore in automated measurements systems, where no visual display of the results is required, the display update can be turned off for better performance. R&S spectrum and network analyzers switch off display update by default, when the instrument goes into remote control operation.

For development of test programs, or if the instrument display update is required for alignment or other purposes, the display update can be turned on or off manually (Softkey DISP UPD) or by using the remote control command:

`SYSTem:DISPlay:UPDate ON | OFF` or

`SYSTem:DISPlay:UPDate  1 | 0`

The state of display update (ON or OFF) is persistent as long as the instrument is powered up. Therefore, for best performance of fully automated operation it is recommended to switch display update OFF at initialization of the test program.

3.2 Continuous vs. Single Sweep

In manual operation, the instrument typically performs measurements repeatedly, i.e. it operates in continuous sweep mode. This is useful for alignment tasks, where the instrument display immediately reflects adjustments performed on the device under test (DUT).

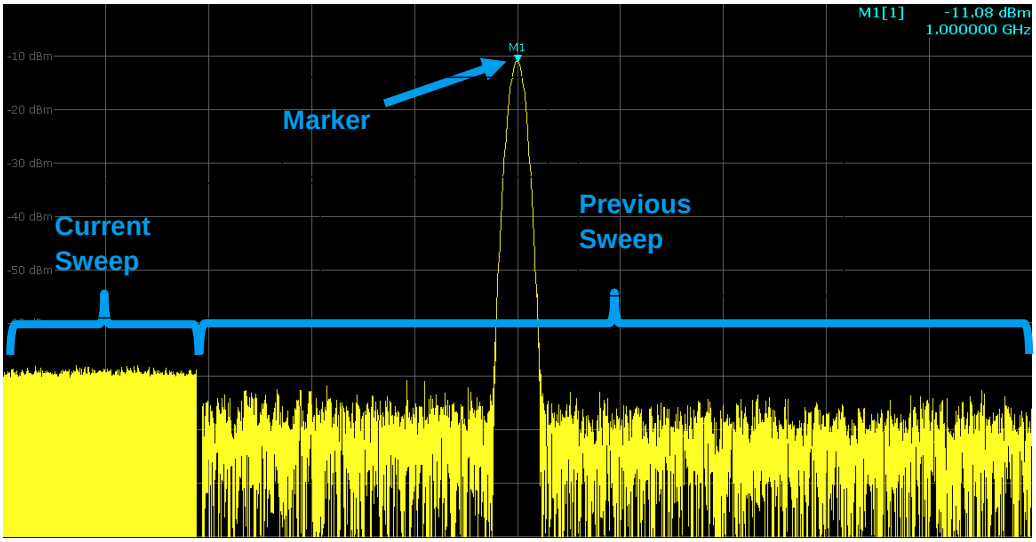
For remote control operations, however, continuous sweep presents several issues:

Parameter changes are slower, because the instrument firmware must recalculate hardware settings after every single remote control command. In single sweep mode, the remote control command affects only the instrument's settings database. All hardware settings are calculated before the single sweep is initiated. In addition, instrument response to remote control commands is faster in single sweep mode because the instrument processor does not share resources with the measurement task continuously.

More importantly, instrument results can be inconsistent in continuous sweep operation, because measurements values can come from different sweeps.

Synchronization to the end of the sweep does not work in continuous sweep operation. `INIT;*OPC?` will return immediately and not related to sweep completion.

The following figure shows a spectrum measurement, where the sweep time was changed between two sweeps.



If the test software reads the trace data under such conditions, the resulting array of values consists of a mixture from two subsequent measurements. The data is obviously invalid.

Also, performing marker search operations in continuous sweep mode can lead to invalid results. In the above figure, the marker search was initiated during a running sweep. The peak search operation is performed instantaneously and therefore it is not guaranteed that the result leads to the actual maximum value of the measurement. The marker position after the search could be somewhere in the previous sweep or on the current (incomplete) sweep.

4 Common Issues and Pitfalls

4.1 Address Conflicts

Check that GPIB addresses are unique in the system. If duplicate addresses are present, the instrument which reacts faster will try to execute the commands. Errors in such scenarios are very difficult to diagnose and locate.

4.2 Termination Character for Binary Data Transfer

For binary data transfer the termination character must be set properly. The default termination character <LF> can occur anywhere in a binary stream. If the termination character is enabled, the data transfer terminates at the first occurrence of this character. Therefore, the termination character must be disabled and data transfer must be based on byte count instead of termination character. When writing binary data to the instrument, an additional command must be sent to the instrument such that it doesn't abort data reception upon occurrence of <LF>. This command is:

```
SYST:COMM:RTER EOI
```

4.3 Disable Auto Serial Poll

GPIB drivers include the capability to perform a serial poll automatically when a service request occurs. This causes problems for the test software because the result of the serial poll is no longer available to the application software if the driver performed the serial poll automatically. To avoid this issue, automatic serial poll should be disabled in the default driver configuration.

4.4 Check the Error Queue

"A program that does what it should do, is not necessarily correct". At least during program development the error queue should be checked after each command. Especially if new commands are inserted in an existing command line the risk is to forget a ":" or ";", which might lead to time consuming investigations on malfunctions of the program.

4.5 Avoiding Timeouts on Long Execution Times

If the timeout value for a query command cannot be set sufficiently large, a loop may be used instead (see "loops with short delay").

4.6 Timeout Occurs in Unexpected Place

R&S spectrum and network analyzers have a GPIB controller, which has an internal FIFO that can store multiple short commands. This means, that the GPIB controller responds to the bus handshake as if commands were already executed. As a consequence a timeout may occur at the first command after the FIFO is full and one of the preceding commands is blocking. This behavior is different compared to earlier generations of GPIB interfaces.

5 Measurement Synchronization

Before a measurement result is retrieved from the instrument, one must ensure that the instrument has completed the measurement and the result is valid and consistent with the instrument settings. It is therefore mandatory to synchronize the test program flow with the measurement (sweep) completion before attempting to read any results from the instrument.

Sweep end synchronization can only be done properly if the instrument operates in single sweep mode. None of the subsequent synchronization mechanism reports the sweep end correctly in continuous sweep mode. If, for example the instrument sweeps continuously and the client application attempts to read trace data, the resulting values might be a mixture from different measurements.

It is generally not required to synchronize the completion of each command (instrument setting), because the instrument firmware ensures that all parameters are set before actually starting the measurement. However, it may be useful to synchronize lengthy operations (e.g. calibration, mode changes, etc.).

IEEE-488.2 defines 3 mechanisms ensure operation completion. These are explained in the following paragraphs.

5.1 Synchronization with *WAI

The simplest way to synchronize remote control commands is by appending the IEEE-488.2 Common Command `*WAI` to other commands. It is irrelevant, whether this command is directly appended to another command or sent separately. The following pseudo-code sequences are equivalent:

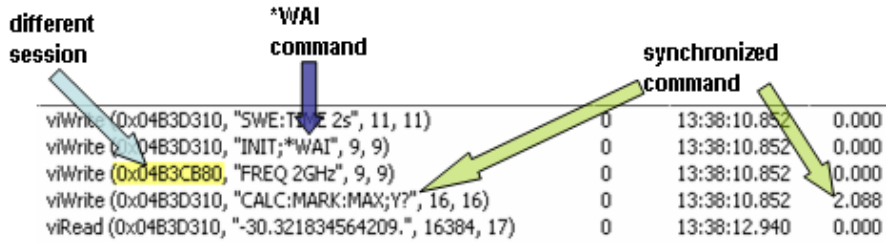
```
InstrumentWrite("INIT:CONT OFF");
InstrumentWrite("INIT;*WAI");
InstrumentQuery("CALC:MARK:MAX;Y?");
```

and

```
InstrumentWrite("INIT:CONT OFF");
InstrumentWrite("INIT");
InstrumentWrite("*WAI");
InstrumentQuery("CALC:MARK:MAX;Y?");
```

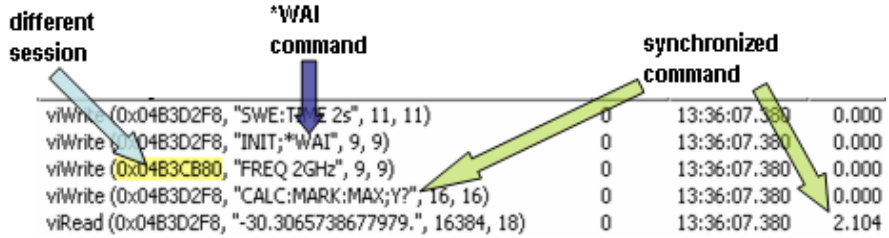
In both cases the command subsequent to `*WAI` is executed only after sweep completion.

The following communication log illustrates the timing. In this example, communication was performed using a LAN connection with the VXI-11 protocol:



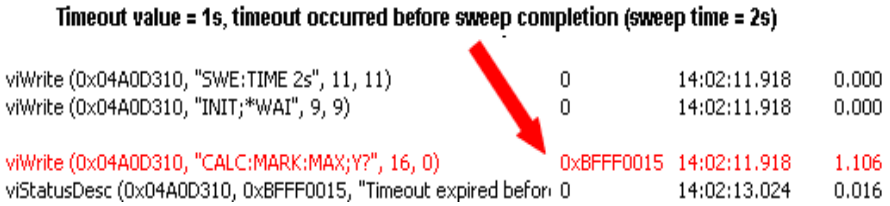
Note that `*WAI` does not affect the communication with other instruments. In the example above, the frequency setting of a second instrument is performed immediately after the initiation of the sweep. The marker search operation however, is delayed for 2 seconds, because the sweep time was set to this value.

Where exactly the delay occurs, may depend on the chosen IO channel. Due to message buffering in the case of some GPIB architectures, the delay occurs at the read operation:



Important is the fact that the result of the synchronized operation (here marker value) is returned upon completion of the preceding commands (here sweep completed).

Note: Proper execution of `*WAI` requires that the timeout value of the session is set to a value larger than the execution time of commands which are to be synchronized. If the value is too small, the program flow continues with a timeout error.



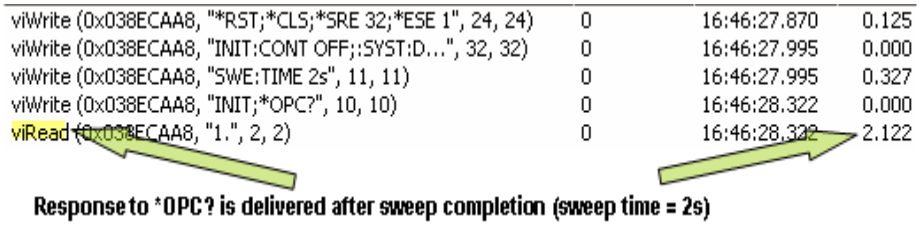
This is one of many reasons, why the test software should always test IO function status codes for success.

5.2 Synchronization with *OPC?

Another simple synchronization mechanism is based on the IEEE-488.2 Common Command `*OPC?` . Similar to `*WAI` this command can be appended or sent separately after the command to be synchronized with.

The major difference is the fact that the result of the `*OPC?` query determines the synchronization. The communication with the instrument is halted at the time of the

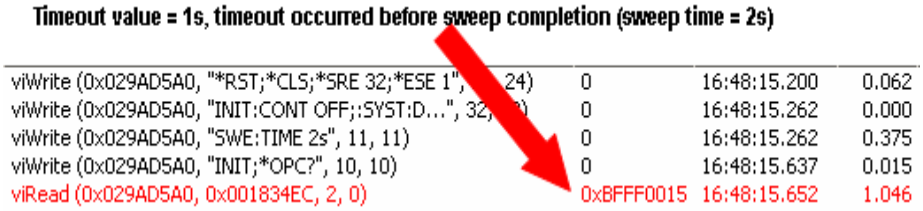
`Read()` operation until previous commands are completed.



If the call to the read function is omitted, no synchronization occurs and a SCPI error will be generated, because the instrument response to the `*OPC?` query was not retrieved by the client application. The instrument returns “1” for successful completion of the preceding operation.

Conditions for the timeout value are the same like for the `*WAI` mechanism, i.e. the timeout value must be larger than the execution time of the command whose completion must be synchronized.

If a timeout occurs before operation completion, the `Read()` function will return an error and the response to the `*OPC?` query contains “0”.



Note: In Microsoft Visual Basic 6 it is important to pre-allocate the response buffer before reading from the instrument. Due to automatic string length adjustment, read operations are not performed correctly if the response buffer is not initialized to a finite length.

Typical code sequence (VB6):

```
response$ = Space$(100)
viRead(hdl, response$, 100, retCount)
```

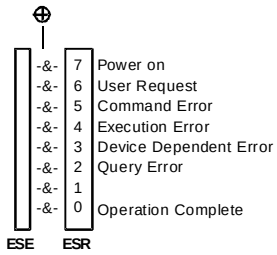
5.3 Synchronization with *OPC

The third alternative for synchronization is the most complex and also the most flexible one.

Similar to `*WAI` and `*OPC?` the IEEE-488.2 Common Command *OPC is appended or sent separately after the command to be synchronized with.

Note: `*OPC?` does not influence the status registers, nor does it affect the service request mechanism. In this context `*OPC` and `*OPC?` are very different and must not be confused.

All instruments that comply to IEEE-488.2, implement the Event Status Register (ESR). Bit 0 of this register is defined as Operation Complete bit.



Synchronization can be achieved by polling the status of the Operation Complete bit using the query command `*ESR?`. A read operation on the Event Status Register is destructive, i.e. the register is cleared after the read operation and thus the contents cannot be retrieved multiple times.

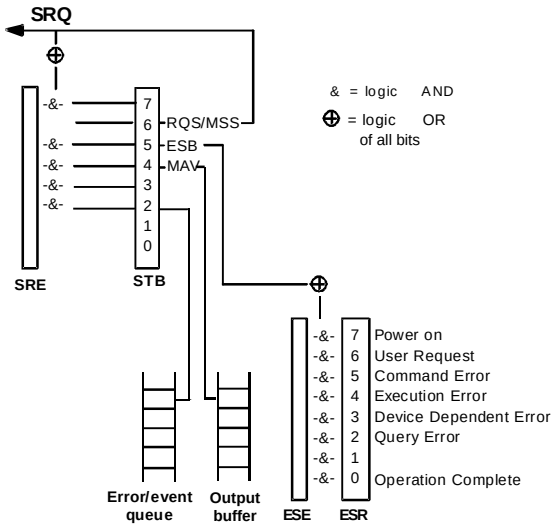
5.4 Loops with Short Delay

Polling the ESR is not recommended, because it will occupy a lot of CPU resources, resulting in slower performance of the desired function. Instead of permanently polling for OPC in a loop, the poll routine should include a delay in the range of the estimated execution time, and after that short delays in order to not waste time after the OPC event occurred.

Therefore `*OPC` is mostly used in conjunction with the Service Request (SRQ) mechanism, which is described in the next chapter.

6 Service Request (SRQ) Handling

The Service Request mechanism allows generic handling of asynchronous events coming from different sources and for different reasons. The figure below shows the principle that determines the generation of a service request.



Bits of the Status Byte (STB) are masked with the contents of the Service Request Enable (SRE) register and combined with a logical OR operation into the RQS/MSS (Master Status Summary) bit. If this bit is set, the instrument issues a service request.

The controlling test software can select the potential sources for service requests by setting the mask registers SRE and ESE accordingly. For example, if the test program wants to be notified when the instrument output buffer contains data or an error occurred in the instrument, the bits 4 (=MAV) and 5 (=ESB) must be enabled in the SRE register. Bits 2-5 must be enabled in the ESE.

Before the measurement is initiated, the controller sends these commands to the instrument:

```
InstrumentWrite("*SRE 64");

InstrumentWrite("*ESE 60");
```

The values are the decimal representation of the bits in the mask registers.

This mechanism can be used for operation complete synchronization by enabling bit 0 in the ESE. Appending `*OPC` to the command to be synchronized will cause the Operation Complete bit to be set. As a consequence, the ESB (=event status sum bit) is set in the status byte and if bit 5 in the SRE is enabled, it leads to a service request.

Note: The service request event does not contain information about its source. Therefore the controller software is responsible for determining the source and reason of the service request, before taking further action.

The subsequent pseudo-code below shows how measurement synchronization can be implemented using the service request mechanism.

Initialize status registers (enable the corresponding mask registers) and select single sweep mode:

```
InstrumentWrite("*CLS"); //Clear status registers

InstrumentWrite("*SRE 32;*ESE 1"); //Enable masks

InstrumentWrite("INIT:CONT OFF"); //Single sweep
```

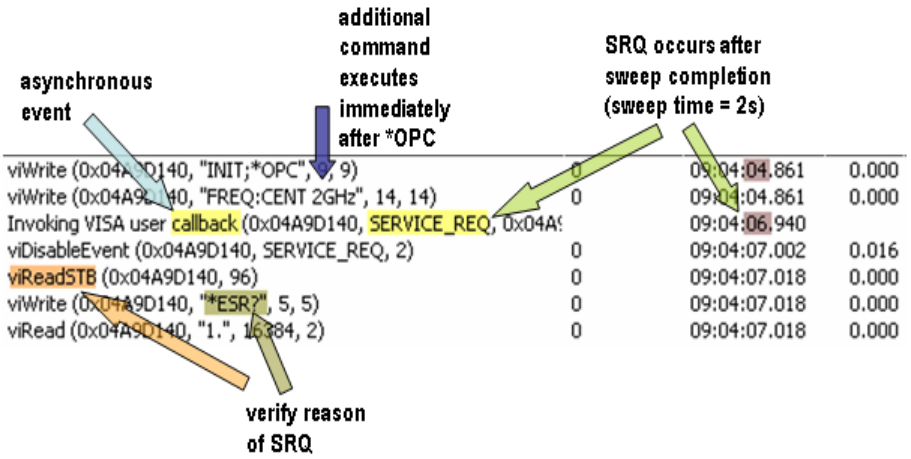
Start measurement with `*OPC`:

```
InstrumentWrite("INIT;*OPC");
```

After initiating the measurement, the controller software must wait for the service request event to occur. A timeout can be specified.

```
WaitOnEvent(ServiceRequest, 3000);
```

Depending on the test software design, the program flow can continue after the measurement was started. The service request will occur asynchronously and interrupt the normal execution of the program.



If a generic service request handler is used, the routine must determine the reason for the event.

Here is an example of a generic service request handler. This example is written in C#:

```
private void OnServiceRequest(object sender, MessageBasedSessionEventArgs e)
{
    MessageBasedSession session = sender as MessageBasedSession;
    if (session != null)
    {
        try
        {
            // Disable event
            session.DisableEvent(MessageBasedSessionEventType.ServiceRequest,
                EventMechanism.Handler);

            // Get status byte by performing a serial poll
            StatusByteFlags sb = session.ReadStatusByte();

            if ((sb & StatusByteFlags.MessageAvailable) != 0)
            {
                MessageBox.Show("MAV in status register is set.");
            }
            else if ((sb & StatusByteFlags.EventStatusRegister) != 0)
            {
                MessageBox.Show("ESB in status register is set.");
                // Get content of event status register
                string esrValue = session.Query("*ESR?");
            }

            // Clear status registers
            session.Write("*CLS");
        }
        catch (Exception exp)
        {
            MessageBox.Show(exp.Message);
        }
    }
    else
    {
        MessageBox.Show("Sender is not a valid session");
    }
}
```

7 Production Test Considerations

In production environments the test software must cope with failing devices under test without significantly interrupting the production flow. Under these conditions the test program must handle typically 2 scenarios that fall outside of the regular pass/fail limit check:

1. A triggered measurement doesn't complete if the trigger condition (e.g. low signal level) is not satisfied.
2. A measurement result may not be available if the signal doesn't comply with certain standards.

Both of the above topics occur in production test of devices for digital communications (e.g. IEEE-802.11 signals).

In the first case (trigger failed) the measurement (sweep) will not complete. Depending on the synchronization mechanism, this condition results in a timeout (`*WAI` or `*OPC?`) or the operation complete event (if service request is configured for that) never occurs.

Consequently the test software must verify for operation completion (sweep finished) before retrieving results (e.g. trace data). In the case of `*OPC?` this task can be accomplished with the subsequent pseudo-code:

```
InstrumentWrite("INIT;*OPC?");
InstrumentRead(response);
If (response == '1')
    traceData = InstrumentQuery("TRAC? TRACE1");
End if
```

The second scenario is more complex. Assuming that modulation characteristics are to be measured on a bursted signal, it can very well be that the trigger condition occurs but the signal modulation is not valid. If the user tries to read for example EVM data, the query function times out because the instrument has no data available.

In order to avoid the timeout, the software can test the status of the MAV bit (message available) in the status register (STB) before attempting to read the value.

Retrieving the contents of the status byte must be done by means of a serial poll operation. Using the standard `*STB?` query command would overwrite the contents of the instrument output buffer. Furthermore, the serial poll must be done after the query command was sent to the instrument and before the `Read()` function is called. The following pseudo-code illustrates this sequence:

```
InstrumentWrite("FETC:BURS:EVM:ALL:MAX?");
ReadSTB(statusByte);  // Perform serial poll
If (statusByte & MAV) // Check for message available
    InstrumentRead(response);
End if
```

8 Ordering Information

Designation	Type	Order No.
NI-VISA IO library for instrument control		1310.0054.02

About Rohde & Schwarz

Rohde & Schwarz is an independent group of companies specializing in electronics. It is a leading supplier of solutions in the fields of test and measurement, broadcasting, radiomonitoring and radiolocation, as well as secure communications. Established more than 75 years ago, Rohde & Schwarz has a global presence and a dedicated service network in over 70 countries. Company headquarters are in Munich, Germany.

Regional contact

Europe, Africa, Middle East
+49 89 4129 12345
customersupport@rohde-schwarz.com

North America
1-888-TEST-RSA (1-888-837-8772)
customer.support@rsa.rohde-schwarz.com

Latin America
+1-410-910-7988
customersupport.la@rohde-schwarz.com

Asia/Pacific
+65 65 13 04 88
customersupport.asia@rohde-schwarz.com

China
+86-800-810-8228 /+86-400-650-5896
customersupport.china@rohde-schwarz.com

Environmental commitment

- Energy-efficient products
- Continuous improvement in environmental sustainability
- ISO 14001-certified environmental management system

Certified Quality System
ISO 9001

PAD-T-M: 3573.7380.02/02.01/EN/

Rohde & Schwarz GmbH & Co. KG
Mühldorfstraße 15 | D - 81671 München
Phone + 49 89 4129 - 0 | Fax + 49 89 4129 – 13777

www.rohde-schwarz.com