スペクトラム・アナライザとネットワーク・アナライザのリモート制御のヒントと秘訣

アプリケーションノート

Products:

- R&S®FSL
- R&S®FSP
- R&S®FSQ
- R&S®FSU
- R&S®FSMR
- R&S®FSUP
- R&S®ZVA
- R&S®ZVB
- R&S®ZVT
- R&S®ZVL

このアプリケーションノートは、スペクトラム・アナライザとネットワーク・アナライザを使用してリモート制御プログラムを実装するためのヒントを提供します。
また、リモート制御のパフォーマンス改善の提案と同期測定の特徴を詳しく説明します。

最後に、生産試験でのリモート制御についていくつかの代表的な問題を説明します。

Application Note
Johannes Ganzert
07.2007 – 1EF62_0J

ROHDE&SCHWARZ

目次

1 概要

このアプリケーションノートは、スペクトラム・アナライザ及びネットワーク・アナライザに共通する、リモート制御の特徴について説明します。特に、測定の同期について、いくつかの方法を説明します。

本ドキュメントは、テスト・ソフトウェアが最高のスループットで設計されると同時に、生産試験ソフトウェアにおいて、テスト中に欠陥があるデバイスを扱わなければならない場合について、いくつかのヒントを示します。

本ドキュメントのプログラムサンプルは特に断りのない限り、Visual Basic 6 で記載されています。

2 共通 I/O の考慮

リモート制御にはさまざまなアプローチがあります。テスト・ソフトウェアは、計測器ドライバを使用する方法、もしくは下位レベルの I/O ライブラリに直接アクセスする方法を選択することができます。

2.1 計測器ドライバ

計測器ドライバは、一般的な開発環境：Labview、LabWindows/CVI、Agilent-VEE、Visual Basic、C++、C#向けに、弊社のウェブサイト http://www.rohde-scwarz.com より無料で利用することができます。

計測器ドライバは I/O 制御の為に VISA ライブラリを使用しています。

2.2 共通 I/O ライブラリ – VISA

もし、計測器ドライバが特別な開発環境で利用できない場合、またドライバが全てのユーザ要求を実現させることができない場合、テスト・ソフトウェアは I/O ライブラリの直接呼出し、または計測器の SCPI コマンドセットを使用することにより、リモート制御を実行することができます。

この場合、標準化された VISA ライブラリの使用を推奨します。
それは、ユーザに分かりやすい I/O チャネルの選択をさせます。従って、テスト・ソフトウェアは一つの I/O チャネルから他の I/O チャネル（例：GPIB → LAN）に容易に移行することができます。

VISA ライブラリの I/O 関数はサポートされている全てのハードウェア・インタフェースで同じです。特定の I/O チャネルを開く為の手順のみ、選択されたインタフェースによって異なります。

次の手順は、GPIB を使用した際の接続のオープン、そして計測器から ID 文字列を読み込んでいます。

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

もし、上記インタフェースが LAN に変更された場合、上記コードのリソース文字列（"GPIB::20::INSTR"の箇所）のみ変更します。

```
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)
```

VISA ライブラリはいろいろなプラットフォームで利用できます。
VISA ライブラリは、ローデ・シュワルツ（付録のオーダー情報をご参照ください）、または他の電子計測器ベンダー（アジレント・テクノロジー、ナショナルインメンツなど）より入手可能です。

3 リモート制御の最適化

リモート経由で計測器を制御することは簡単ですが、全体的なパフォーマンスや、測定終了時間等、いくつかの注意点があります。自動化されたテストシステムのスループットは重要です。次の項では、リモート制御を行う上でいくつかの重要な内容について詳しく説明します。

3.1 表示画面の更新

グラフィック描画は重要な計測器のコンピュータ・リソースを消費します。そのため、自動測定システムでは描画無の結果が必要とされます。ディスプレイアップデート機能は、よりよいパフォーマンスの為、画面表示を OFF にさせることができます。ローデ・シュワルツのスペクトラム・アナライザとネットワーク・アナライザはリモート制御を開始する際、デフォルト設定で画面表示を OFF にします。

テスト・ソフトウェア開発のため、またはアライメントや他の目的で計測器の画面表示が必要な場合、画面表示を手動（ソフトキーの DISP UPD）、もしくは以下のリモート制御コマンドを使用して ON/OFF を切り替えるができます。

```
SYSTem:DISPlay:UPDate ON | OFF or
SYSTem:DISPlay:UPDate  1 | 0
```

画面表示の状態設定（ON/OFF）は、計測器の電源が入っている限り継続します。
そのため、完全な自動制御でベストパフォーマンスを出すには、テスト・ソフトウェアの初期化処理で画面表示を OFF にすることを推奨します。

3.2 連続掃引 vs. シングル掃引

手動操作では、計測器は通常、繰り返し測定を行います。すなわち、連続掃引モードで動作します。これは実行された DUT（device under test）の調整をすぐに画面に反映するので、調整の目的には便利です。

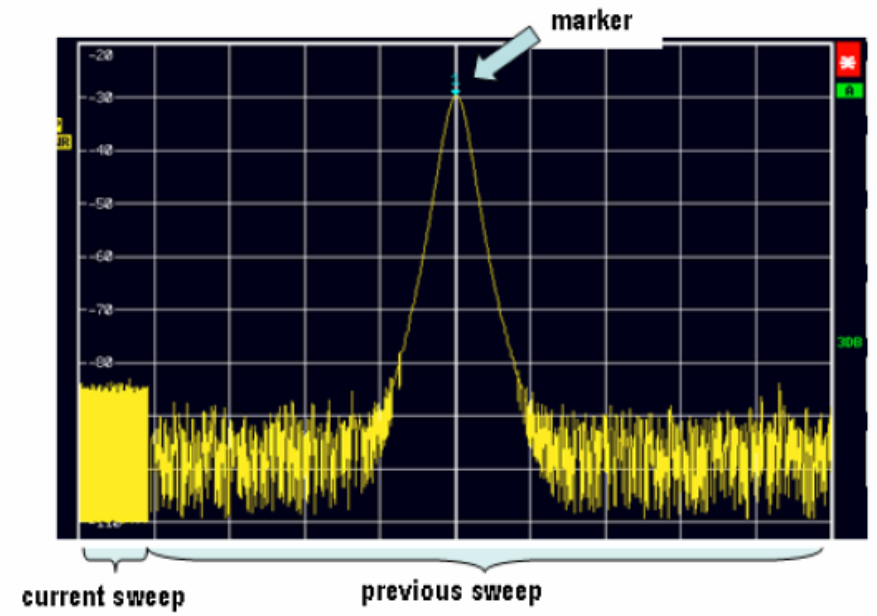
リモート制御では、連続掃引はいくつかの問題を引き起こします。

1. パラメータ変更が遅くなります。
   計測器のファームウェアは、あらゆるリモート制御コマンドの後でハードウェア設定を計算し直さなければなりません。シングル掃引モードでは、リモート制御コマンドは計測器の設定データベースのみに影響します。全てのハードウェア設定は、シングル掃引が開始される前に計算されます。さらに、リモート制御コマンドの計測器応答はシングル掃引モードでより早くなります。それは、計測器のプロセッサが連続した測定タスクに伴うリソースを共有していないためです。

2. さらに重要なことは、計測器の結果は連続掃引では一貫性がありません。それは、測定値が異なった掃引によってもたらされるためです。

3. 掃引終了時の同期は連続掃引では動作しません。”INIT;*OPC?” コマンドはすぐに応答してしまい、掃引終了とは連係しません。

次の図は、掃引時間が変更された 2 つの掃引のスペクトラム測定を示しています。



もし、テスト・ソフトウェアが上図の状態でトレースデータを読み込んだ場合、トレースデータを格納した配列の値は 2 つの測定が混ざったものになります。このデータはあきらかに正しくありません。

また、連続掃引モードでマーカ・サーチを実行すると正しくない結果になります。上図では、マーカ・サーチは掃引中の間に開始されたものです。ピーク・サーチはすぐに実行されてしまうため、結果が実際の測定のピーク値を示すことは保証されません。掃引後のマーカのポジションは、前の掃引か、現在の（不完全な）掃引のどこかにあります。

4 共通の問題と落とし穴

4.1 アドレスの衝突

システム上の固有の GPIB アドレスをチェックします。もし、重複したアドレスが存在している場合、より早く反応する計測器がコマンドを実行するよう試みます。このような状況では、エラーの発見や解析を行うことがとても難しくなります。

4.2 バイナリデータ転送中の不適切な終端文字

バイナリデータ転送における終端文字は適切に設定されなければなりません。デフォルトの終端文字<LF>は、バイナリストリーム上のどこかで発生します。もし、終端文字が使用可能な場合、データ転送は最初に発生するこの文字で終了します。そのため、終端文字は無効化されなければならず、データ転送は終端文字の代わりにバイト数をカウントするようにしなければなりません。バイナリデータを計測器に書き込む時、追加するコマンドは<LF>の発生で計測器のデータ受信が中断されないよう送信されなければなりません。

終端文字<LF>を無効にするコマンド:

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

4.3 自動シリアルポールを無効にする

GPIB ドライバはサービスリクエストが発生した時、自動的にシリアルポールを実行する機能があります。これはテスト・ソフトウェアに問題が生じます。もしドライバが自動でシリアルポールを実行した場合、シリアルポールの結果はテスト・ソフトウェアではすでに利用できません。この問題を避けるため、自動シリアルポールはデフォルトのドライバ設定で無効化されるべきです。

4.4 エラーキューのチェック

″するべきことをするプログラムは必ずしも正しいわけではありません。″
少なくともプログラム開発中、エラーキューはコマンドの後毎にチェックされるべきです。特に既存のコマンドラインに新しいコマンドを挿入した場合、″:″や″;″を忘れる危険があります。これは、プログラム異常の調査で時間を浪費することに繋がる可能性があります。

4.5 長時間のタイムアウト設定を避ける

もし、クエリコマンドのタイムアウト値を十分に長く設定することができない場合、ループ処理を変わりに使用できます。(″短い時間のループを参照″)

4.6 タイムアウトが予期しない場所で発生する

ローデ・シュワルツのスペクトラム・アナライザとネットワーク・アナライザは GPIB コントローラを持っており、複数の短いコマンドを保持できる内部 FIFO を持っています。これは、GPIB コントローラがすでにコマンドを実行したかのように、バスのハンドシェークに応答することを意味します。結果として、FIFO が一杯で、先行しているコマンドが何らかの障害でバスを塞いでいる場合、その後の最初のコマンドでタイムアウトが発生する可能性があります。
この動作は、旧世代の GPIB インタフェースと比較すると異なります。

5 同期測定

計測器から測定結果が取得される前に、計測器の測定が終了しており、結果が有効で計測器の設定と一致していることを確認しなければなりません。計測器から結果を読むことを試みる前に、測定(掃引)終了でテスト・ソフトウェアに同期することは必須です。

掃引終了の同期は、計測器がシングル掃引モードで動作した場合のみ正しく実行されます。以下の同期方法は連続掃引モードでは掃引終了は正しく報告されません。

例えば、計測器が連続的に掃引している時にテスト・ソフトウェアがトレースデータの読み込みを行うと、前の掃引と現在の不完全な掃引が混合した測定値になる可能性があります。

通常、各コマンド(計測器設定)の終了同期は必要とされません。計測器のファームウェアは、実際に測定が始まる前に全てのパラメータが設定されることを確認します。しかし、非常に長い動作の同期に役立つことがあります。(例:キャリブレーション、モード変更等)

IEEE-488.2 は動作終了を確認する3つの方法を定義しています。これらは次の章で説明します。

5.1 *WAI による同期

リモート制御コマンドと同期を取る一番簡単な方法は、他のコマンドに IEEE-488.2 のコモンコマンド(*WAI)を付け加えるです。このコマンドが直接他のコマンドに付け加えられるか、別々に送られるかどうかは関係ありません。次のプログラムコードは同じ動作になります。

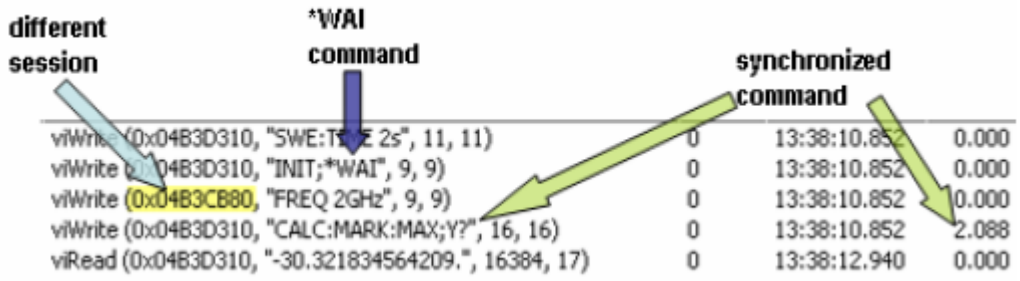
```
InstrumentWrite("INIT:CONT OFF");
InstrumentWrite("INIT;*WAI");
InstrumentQuery("CALC:MARK:MAX;Y?");
```

and

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

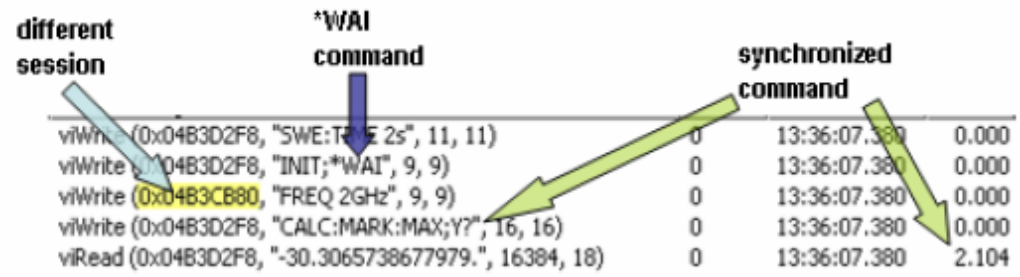
*WAI コマンドの次に続くコマンドは、上記両方のプログラムコード共、掃引終了後のみ実行されます。

次の通信ログはタイミングを説明しています。この例では、通信は VXI-11 プロトコルで LAN 接続を使用して実行されたものです。



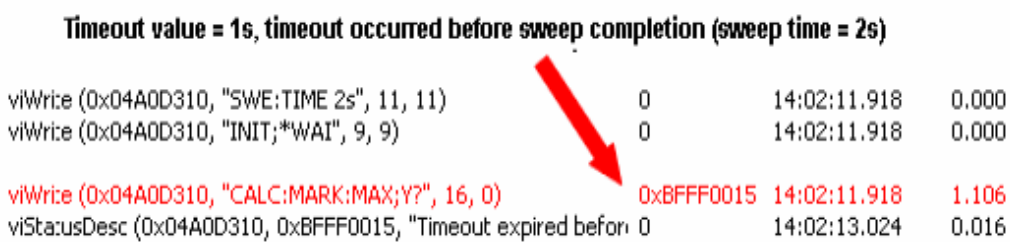
ここで留意すべきことは、*WAI は他の計測器の通信には影響されません。上記の例では、他の測定器（different session）の周波数設定は掃引開始の後にすぐ実行されます。しかし、その次のマーカ・サーチ動作（synchronized command）は 2 秒遅れています。これは、掃引時間が 2 秒に設定されたためです。

遅延がどこで発生するかは選択された I/O チャネルによります。GPIB の場合、読み込み動作の時にメッセージのバッファリングが原因で遅延が発生します。



重要な点は、同期された動作の結果（取得されたマーカ値）が、先に実行している掃引コマンド（INIT;*WAI）の終了（掃引終了）で返されることです。

注記:
*WAI の適切な実行は、各セッションのタイムアウト値が同期されたコマンドの実行時間より長い値を設定する必要があります。もし値がとても小さい場合、プログラムはタイムアウト・エラーになります。

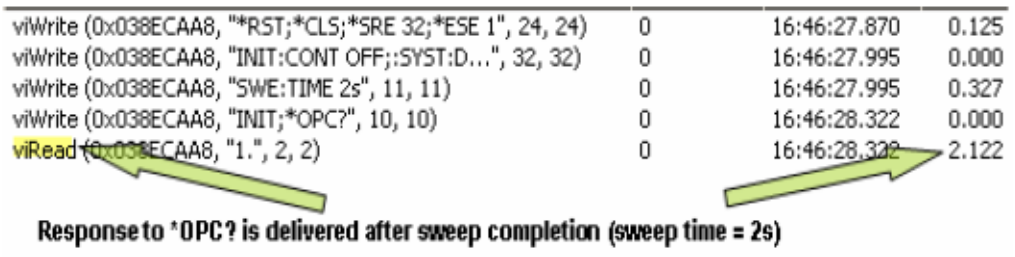


これは、テスト・ソフトウェアが I/O 関数の実行を成功させるためにいつも確認するべき項目の一つです。

5.2 *OPC?による同期

他の簡単な同期方法は IEEE488.2 のコモンコマンド(*OPC?)に基づいています。*WAI コマンドと同様に、このコマンドは同期されたコマンドの後に付け加えられるか、分けて送ることができます。

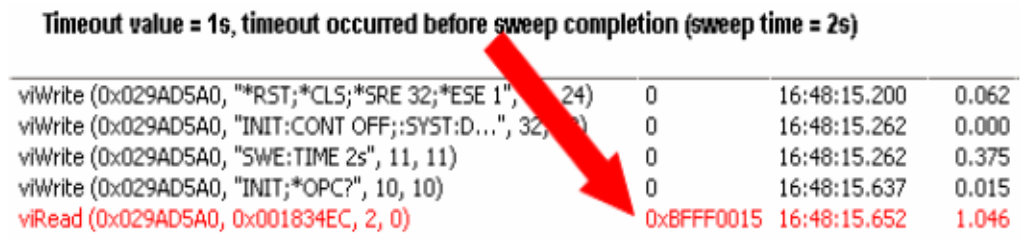
*WAI との大きな違いは、*OPC?クエリの結果は同期で定義されます。計測器による通信は、前のコマンドが終了する迄、読み込み関数の動作の時に中断されます。



もし、読み込み関数の呼出しを除いた場合、同期は発生せず、SCPI エラーが発生します。これは*OPC?クエリの計測器応答がテスト・ソフトウェアによって読み出されないためです。計測器は実行している動作が正常終了した場合、"1"を返します。

タイムアウト値の状態は、*WAI による方法と同様です。例として、タイムアウト値は終了が同期されなければならないコマンドの実行時間より長くなければなりません。

もし、タイムアウトが動作終了の前に発生した場合、読み込み関数はエラーと"0"を含む*OPC?クエリの応答を返します。



注記:
Microsoft Visual Basic 6 では、計測器から読み出しを行う前に、受信用バッファをあらかじめ割り当てることが重要です。これは自動的な文字列長の調整のためで、もし受信用バッファが限られた長さで初期化されなかった場合、読み出し動作は正しく行われません。

標準的なプログラムコード(VB6)

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

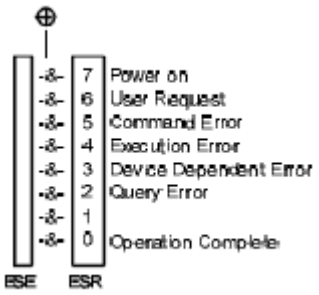
5.3 *OPC による同期

3つ目の同期手段はもっとも複雑ですが、もっとも融通が利きます。

*WAI と*OPC?と同様に、IEEE-488.2 コモンコマンド(*OPC)は同期されたコマンドの後に付け加えられるか、分けて送られます。

注記:
*OPC?はステータス・レジスタにも、サービスリクエストにも影響されません。
*OPC と*OPC?はとても違うので混同しないようにしてください。

IEEE-488.2 に従う全ての計測器はイベント・ステータス・レジスタ(ESR)を実装しています。このレジスタのビット 0 が Operation Complete ビットとして定義されます。



同期はクエリコマンド*ESR?を使用し、Operation Complete ビットのステータスをポーリングすることで実現します。イベント・ステータス・レジスタの読み込み動作はレジスタの内容に影響を与えます。
例として、レジスタは読み込み動作の後にクリアされるので、レジスタの内容は何度も読み出すことができません。

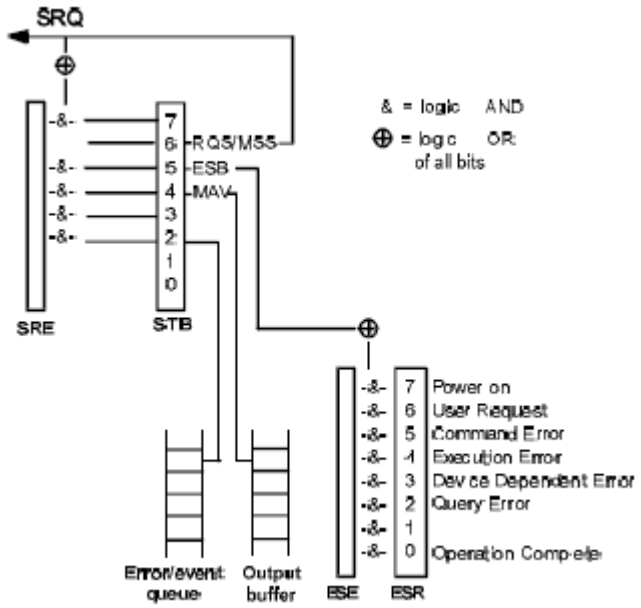
5.4 永久ポーリングよりも短い時間のループ

ESR のポーリングは多くの CPU リソースを占有してしまうため、推奨できません。所望の機能の低パフォーマンス化をもたらします。OPC による永久ループの代わりに、ポーリング処理は予測実行時間の範囲の遅延時間と OPC イベントが発生した後の時間を無駄にしないため、短い遅延時間を入れるべきです。

そのため、*OPC は通常、次の章で説明されているサービスリクエスト(SRQ)と連動して使われます。

6 サービスリクエスト(SRQ)の扱い方

サービスリクエストは、異なった計測器から来る非同期イベントの全体的なハンドリングとさまざまな理由のために割り当てられます。以下の図は、サービスリクエストの発生を定義する原理を示しています。



ステータス・バイト(STB)のビット群はサービス・リクエスト・イネーブル・レジスタ(SRE)でマスクされており、RQS/MSS(Master Status Summary)ビットと論理的 OR で結合されています。もしこのビットがセットされた場合、計測器はサービスリクエストを発行します。

テスト・ソフトウェアは、設定したマスクレジスタ(SRE と ESE)により、サービスリクエストを受けることが可能な情報源を選択することができます。例えば、テスト・ソフトウェアが計測器の出力バッファのデータを含んでいる、もしくは計測器でエラーが発生したかどうか知りたい場合、SRE レジスタのビット 4(=MAV)と 5(=ESB)を有効化しなければなりません。また ESE レジスタはビット 2-5 を有効化しなければなりません。

測定を開始する前に、コントローラは以下のコマンドを計測器に送信します。

```
InstrumentWrite("*SRE 64");
InstrumentWrite("*ESE 60");
```

上記の値はマスクレジスタの十進表現を表しています。

この動作は ESE のビット 0 を有効化することにより、動作完了の同期に使用することができます。同期するためにコマンドに付け足した*OPC は Operation Complete ビットがセットされると発生します。結果として、ESB(=event status sum bit)はステータス・バイトにセットされ、SRE のビット 5 が有効の場合、サービスリクエストを発生します。

注記:
サービスリクエスト・イベントはその発信元についての情報は含んでいません。そのため、テスト・ソフトウェアは次のアクションの前に、発信元とサービスリクエストの理由を見つける必要があります。

以下のプログラムコードは、サービスリクエストを使用した測定の同期についてプログラムする方法を示しています。

ステータスレジスタ群を初期化(関連するマスクレジスタを有効化にする)し、シングル掃引モードを選択します。

```
InstrumentWrite("*CLS"); //Clear status registers
InstrumentWrite("*SRE 32;*ESE 1"); //Enable masks
InstrumentWrite("INIT:CONT OFF"); //Single sweep
```

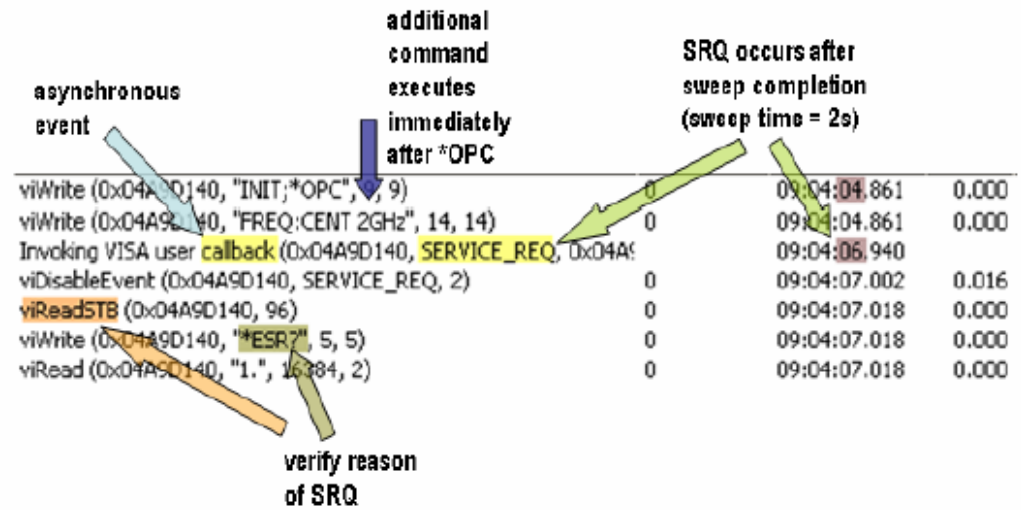
*OPC と共に測定を開始します。

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

測定開始の後、テスト・ソフトウェアはサービスリクエスト・イベントが発生するまで、待たなければなりません。タイムアウトは以下のように設定することができます。

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

テスト・ソフトウェアの設計次第で、プログラムの処理は測定が開始された後に続けることができます。サービスリクエストは非同期に発生し、プログラムの通常実行中に割り込みます。



もし、サービスリクエストを使用する場合、プログラムはイベントが発生した理由を確認しなければなりません。

以下にサービスリクエスト処理の例を示します。この例は C#で記載されています。

```
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 生産試験の考慮

生産環境でのテスト・ソフトウェアは、生産フローを中断することなく、DUT の欠陥に対処しなければなりません。これらの状況下では、テスト・ソフトウェアは通常の pass／fail のリミットチェックの他、以下の 2 つの状況に対処しなければなりません。

1. トリガ測定はトリガ条件（例：低い信号レベル）を満足しなかった場合、測定が終了しません。
2. 測定結果は信号が一定の基準に満たない場合、利用できない可能性があります。

上記の状況はデジタル通信（例 ： IEEE-802.11 信号）向けデバイスの生産試験で発生します。

最初のケース（トリガの失敗）では測定（掃引）が完了しません。同期の方法により、この状態はタイムアウト（*WAI or *OPC?）、もしくはオペレーション・コンプリート・イベント（もしサービスリクエストが設定された場合）が一度も発生しない結果となります。

その結果として、テストソフトウェアは結果を取得（例：トレースデータ）する前にオペレーション・コンプリート（掃引の完了）の確認をしなければなりません。

*OPC?の場合、この処理は以下のプログラムコードで確認することができます。

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

2 番目の状況はより複雑です。
変調特性がバースト信号で測定されると仮定すると、信号変調が有効ではなくても確実にトリガを発生させることができます。
例として、ユーザが EVM データの読み込みを試みようとした場合、計測器は利用できるデータを持っていないので、クエリ処理はタイムアウトします。

タイムアウトを避けるため、ソフトウェアは値の読み込みを試みる前にステータスレジスタ（STB）上の MAV ビット（message available）のステータスを確認します。

ステータス・バイトの読み出しはシリアルポールによって実行されなれけばなりません。標準のクエリコマンド（*STB?）を使用すると、測定器のアウトプット・バッファの内容を上書きしてしまいます。クエリコマンドの後にシリアルポールを計測器に実行させる為、読み込み関数の前に設定します。次のプログラムはこの手順を示しています。

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

8 追加情報

このアプリケーションに関しましてご意見やご提案がございましたら、以下のアドレスまでご連絡ください。

TM_Applications@rohde-schwarz.com

追加情報の為の製品カタログとデータシートはローデ・シュワルツの以下のウェブサイトをご参照ください。

www.rohde-schwazt.com

9 オーダー情報

VISA ライブラリ

VISA IO Libraries	Verson 14.2	1308.9464.00

ローデ・シュワルツについて
ローデ・シュワルツ・グループ（本社：ドイツ・ミュンヘン）は、エレクトロニクス分野に特化し、電子計測、放送、無線通信の監視・探知および高品質な通信システムなどで世界をリードしています。
75 年以上前に創業し、世界 70 カ国以上で販売と保守・修理を展開している会社です。

ローデ・シュワルツ・ジャパン株式会社
本社／東京オフィス
〒160-0023 東京都新宿区西新宿 7-20-1
住友不動産西新宿ビル 27 階
TEL:03-5925-1288/1287 FAX:03-5925-1290/1285

神奈川オフィス
〒222-0033 神奈川県横浜市港北区新横浜 2-8-12
Attend on Tower 16 階
TEL:045-477-3570（代） FAX:045-471-7678

大阪オフィス
〒564-0063 大阪府吹田市江坂町 1-23-20
TEK 第 2 ビル 8 階
TEL:06-6310-9651（代） FAX:06-6330-9651

サービスセンター
〒330-0075 埼玉県さいたま市浦和区針ヶ谷 4-2-20
浦和テクノシティビル 3 階
TEL:048-829-8061 FAX:048-822-3156

E-mail: info.rsjp@rohde-schwarz.com
URL: http://www.rohde-schwarz.co.jp

Certified Quality System
ISO 9001

ローデ・シュワルツ・ジャパン株式会社
〒160-0023 東京都新宿区西新宿 7-20-1 住友不動産西新宿ビル 27 階
TEL:03-5925-1288/1287 FAX:03-5925-1290/1285

www.rohde-schwarz.co.jp