Software-Interface

Andreas Bauer
mb-Technologies, Technisches Büro für technische Physik, GmbH

mbInterface.doc v1.8-de 2009-04-28

Inhaltsverzeichnis

Allgemeines

Einführung

Testsysteme von *mb-Technologies* werden meistens mit Hilfe der mitgelieferten Softwareumgebung *mbTester* verwendet. Für den Fall, dass Kunden ihre eigene Software einsetzen wollen, wird die Schnittstellenbibliothek *mbInterface.dll* mitgeliefert.

Die Schnittstelle ist zu Programmiersprachen wie Microsoft© Visual Basic® oder C++ und zu Mess- und Programmierumgebungen wie LabView® von National Instruments© kompatibel.

Installation

Die Bibliothek wird zusammen mit der Standardsoftware von *mb-Technologies* installiert und ist danach im Installationsordner der *mbTester*-Software (normalerweise *c:\Programme\mbTester\)* zu finden. Die aktuelle Dokumentation liegt im Unterverzeichnis *Help*, die Beispielprogramme im Verzeichnis *Example*.

Um die Schnittstellenbibliothek zu verwenden, muss sie in der Programmierumgebung ausgewählt werden. In Microsoft© Visual Studio® wird dies über *Projekte/Referenzen* eingestellt. Wählen sie den Eintrag „mb-Technologies Software Interface“ in der Liste aus oder suchen sie *mbInterface.dll* im Dateisystem und wählen sie die Datei dann aus.

Die Bibliothek ist multitaskingfähig. Hardware-Ressourcen und die Kommunikationsschnittstelle werden automatisch und nach Notwendigkeit gesperrt und wieder freigegeben.

Struktur von Testprogrammen

Jedes Testprogramm benötigt ein *mbInterface* Objekt, das die entsprechenden Methoden und Funktionen bereitstellt. Es ist möglich, mehrere Testprogramme parallel laufen zu lassen. Es werden dann entsprechend viele *mbInterface* Objekte benötigt, die auch in verschiedenen Tasks oder Programmen ausgeführt werden können.

Mit den ersten Befehlen im Testprogramm wird die Hardware konfiguriert: Die Art des Testsystems wird mit dem **Configuration**-Befehl angegeben. Die Default-Einstellung ist ein Parameter-Analyzer. Bei diesem ist dieser Befehl daher nicht notwendig. Bei einem Reliability-Testsystem muss zunächst mit dem **Board**-Befehl das verwendete DUT-Board angegeben werden.

Wenn bei einem Parameter-Analyzer ein Multiplexer verwendet wird, müssen die Verbindungen definiert werden, bevor der erste Messbefehl ausgeführt wird. Das gleiche gilt für Hot-Carrier-Testsysteme. Für jede einzelne Verbindung ist ein **Relay**-Befehl erforderlich. Sobald die Messungen beendet sind, werden alle Source-Measure-Units mit dem **Reset**-Befehl abgeschaltet, der auch alle Relais öffnet. Danach kann eine neue Messung mit geänderten Verbindungen gestartet werden.

Bei Package-Level-Testern ist es möglich, die Multiplexer auch unter Spannung zu schalten. Außer mit **Reset**-Befehl ist es hier auch möglich, einzelne Relais mit dem **RelayOpen**-Befehl

zu öffnen. Die Relais haben allerdings auch bei diesem Testsystem eine begrenzte Schaltleistung. Die Software stellt allerdings sicher, dass vor dem Schalten von Relais-Kontakten bei alle beteiligten SMUs die Strombegrenzung auf einen sicheren Wert reduziert wird. Nach dem Schaltvorgang wird die Strombegrenzung gegebenenfalls wieder auf den vorherigen Wert eingestellt. Dieser Ablauf erfolgt in der Software automatisch, kann jedoch unter Umständen das Messprogramm deutlich verlangsamen. Um das zu verhindern sollte man bei Messprogrammen, die mitten im Messablauf Multiplexer-Relais schalten, bei den SMUs, wo dies möglich ist, eine Strombegrenzung von maximal 1 mA einstellen.

Für Messungen wählt man zuerst eine Source-Measure-Unit mit dem **Source**-Befehl aus. Diese SMU bleibt bis zu einem neuen **Source**-Befehl bzw. bis zu einem **Reset** eingestellt. Bei einem Parameter-Analyzer können alle Source-Measure-Units verwendet werden, die im Testsystem vorhanden sind. Bei einem Reliability-Tester sind die SMUs durch das eingestellte DUT-Board fest vorgegeben und haben immer die logischen Nummern 1 bis 4. Bei Package-Level-Testsystemen werden die SMUs 1 bis 5 und 7 bis 11 für den Mess-Bus, und die SMUs 13 bis 17 und 19 bis 23 für den Stress-Bus verwendet. Bei einem Messprogramm, das mehrere SMUs benötigt, wird einfach vor jede Gruppe von Messbefehlen, die für die selbe Source-Measure-Unit gelten sollen, ein entsprechender **Source**-Befehl gestellt. Die Synchronisierung der SMUs erfolgt automatisch.

Nach dem Auswählen einer Source-Measure-Unit werden üblicherweise die Spannungs- und Strombereiche eingestellt (**VoltageRange**, **CurrentRange, LowestVoltageRange**, **LowestCurrentRange**) sowie mit **Average** die gewünschte Messgeschwindigkeit bzw. Messgenauigkeit.

Die Befehlen **ForceVoltage** bzw. **ForceCurrent** stellen Spannungen bzw. Ströme ein. Die Source-Measure-Unit wird dazu als Spannungsquelle bzw. Stromquelle konfiguriert. Beim Betrieb als Spannungsquelle kann eine Strombegrenzung mit **CurrentLimit** eingestellt werden. Source-Measure-Units von *mb-Technologies* sind vierquadranten-fähig, d.h. sie können unabhängig von der Polarität der Ausgangsspannung Strom liefern oder auch aufnehmen. Die Strombegrenzung gilt dann für beide Polaritäten. Analog kann mit **VoltageLimit** eine Spannungsbegrenzung für den Betrieb als Stromquelle angegeben werden.

Messungen werden mit den Befehlen **MeasureVoltage** und **MeasureCurrent** durchgeführt. Es ist in den meisten Fällen nicht notwendig, eine Verzögerung zwischen dem Anlegen einer Spannung und dem nachfolgenden Messbefehl anzugeben. Die SMU wartet automatisch, bis die eingestellten Werte stabil sind. Dies gilt auch für den Fall, dass mehrere SMUs an einer Messung beteiligt sind. Wenn eine zusätzliche Verzögerung gewünscht wird, steht dafür der Befehl **Delay** zur Verfügung. Dies kann z.B. notwendig sein, um Kapazitäten umzuladen, wenn sich die SMU in einem der unteren Strombereiche befindet.

Für Zeitinformationen besitzt jede SMU einen genauen Timer mit Millisekundenauflösung. Der aktuelle Timerstand kann mit dem **MeasureTime**-Befehl ausgelesen. **ResetTime** setzt den Zähler zurück.

Bei allen Messungen wird gleichzeitig mit dem Messwert der jeweilige Betriebszustand der Source-Measure-Unit gespeichert, also z.B. Spannungsquelle, Strombegrenzung usw. Diese Information kann mit der **Status**-Funktion abgefragt werden.

Wenn eine Source-Measure-Unit verwendet wird, wird diese für andere Testprogramme automatisch gesperrt. Die Sperre wird erst durch den **Reset**-Befehl oder durch das Beenden des Programms aufgehoben.

Mit den Sweep-Befehlen (siehe unten) können auf einfach Kennlinien oder Kennlinienfelder gemessen werden.

Mit den Rampenbefehlen (siehe unten) können automatische Spannungs- und Stromrampen und komplexe Ausgangsfunktionen definiert werden. Während der Rampen sind auch automatische Messungen zu vorgegebenen Zeiten möglich.

Zwischenspeichern von Programmen und Messergebnissen

Aus Performancegründen werden die Messbefehle nicht zum Zeitpunkt der Definition ausgeführt, sondern zunächst zwischengespeichert. Erst mit den Befehlen **Start** oder **Execute** wird das Messprogramm zum Testsystem geschickt und dort lokal abgearbeitet. Die Messergebnisse werden dort zwischengespeichert und asynchron zur Messung zurück zum PC übertragen.

Der **Execute**-Befehl wartet, bis das komplette Messprogramm beendet ist. Da während des **Execute**-Befehls die Anwendung praktisch eingefroren wird, wird dieser Befehl nur für sehr kurze Messprogramme empfohlen (siehe Beispiel 1).

Wenn der **Start**-Befehl verwendet wird, kann der Status des Messprogramms mit dem **Busy**-Befehl laufend abgefragt werden. Dies hat den Vorteil, dass während der laufenden Messung andere Aufgaben erledigt werden können, z.B. können bereits fertiggestellte Messergebnisse angezeigt werden.

Da die Messbefehle nicht unmittelbar ausgeführt sondern zuerst gebuffert werden, kann z.B. ein **MeasureCurrent**-Befehl zum Zeitpunkt der Definition auch noch keinen Messwert liefern, sondern gibt zunächst eine Referenznummer zurück, mit der später, wenn die Messung durchgeführt wurde, der eigentliche Messwert abgeholt werden kann. Es müssen daher vom Anwendungsprogramm für alle Messungen zunächst die entsprechenden Referenznummern zwischengespeichert werden. Den Messwert erhält man - sobald die Messung durchgeführt wurde - mit Hilfe der Funktion **Result**.

Bei einem Messprogramm, das mit **Start** gestartet wurde, können bereits während der laufenden Messung fertiggestellte Messwerte abgeholt werden. Dazu steht die Funktion **Available** zur Verfügung, mit der geprüft werden kann, welche Messergebnis bereits verfügbar sind (siehe Beispiel 2).

Sweepfunktionen

Diese Funktionen ermöglichen eine einfach Definition zur Messung von Kennlinien- oder Kennlinienfeldern. Die Befehle **SweepVoltage** und **SweepCurrent** definieren eine Anzahl von Spannungs- bzw. Stromschritten und gleichzeitig eine Schleife, innerhalb der alle Messbefehle mit der Anzahl der Schritte wiederholt werden. Schleifen können beliebig verschachtelt werden. Das Ende einer Schleife wird von einem **SweepEnd**-Befehl markiert. Mit einem **Start** oder **Execute**-Befehl werden sämtliche Schleifen automatisch geschlossen. Der **SweepEnd**-Befehl kann dann entfallen.

MeasureVoltage, **MeasureCurrent** und **MeasureTime** Befehle innerhalb einer Schleife liefern nicht einen Refernzwert auf einen einzelnen Messwert sondern eine Referenz auf ein Array mit allen Messwerten, die in dieser Schleife gemessen werden. Die Anzahl der Messwerte in diesem Array kann mit der Funktion **Size** berechnet werden.

Die Befehle **Available**, **Result** und **Status** können nach wie vor verwendet werden, allerdings muss als zweiter Parameter der Messwertindex angegeben werden. Alternativ kann mit den Befehlen **ResultArray** und **StatusArray** das gesamte Array auf einmal zurückgegeben werden (siehe Beispiel #5).

Rampenbefehle

Die Rampenbefehle ermöglichen zeitgenaue Spannungs- oder Stromrampen und automatisch ablaufende Ausgangsspannungsmuster. Messungen können mit einer beliebigen Messrate durchgeführt werden.

Im Unterschied zu den Sweepfunktionen werden Rampenfunktionen dann verwendet, wenn es auf den exakten zeitlichen Ablauf ankommt, wie z.B. bei Hot-Carrier-Stress.

Das Ausgangsspannungs- bzw. Strommuster kann aus bis zu 100 einzelnen Abschnitten bestehen, die durch die Einstellwerte und die dazwischenliegenden Zeiten definiert werden. Für jeden Abschnitt sind unterschiedliche Messraten möglich. Das gesamte Rampenmuster oder Teile davon können bis zu einer Million mal wiederholt werden.

Es ist möglich, in einem Testprogramm unterschiedliche Rampenmuster auf mehreren SMUs parallel zu starten. Eine Anwendung wäre z.B. ein Hot-Carrier-Stress bei einem CMOS-Transistor, wo Gate und Drain linear auf die Stressspannung hochgefahren werden und am Ende der Stresszeit gemeinsam wieder herunter auf Null. Während der gesamten Stresszeit kann in regelmäßigen Zeitabständen der Strom gemessen werden, um eventuelle Ausfälle zu erkennen. Optional kann die Messrate während des Messablaufs auch variiert werden. Der Vorteil bei diesem Konzept gegenüber einem konventionellen Messprogramm ist, dass der gesamte Stressablauf inkl. Messung lokal im Testsytem abläuft und die Stress- und Messzeiten genau definiert sind (siehe Beispiel #4).

Falls während der Rampenfunktion Messungen definiert werden, wird über die Funktion **RampResult** eine Referenznummer übergeben, mit der dann, sobald die Messungen durchgeführt wurden, die Messwerte mit Hilfe der **Result** oder **ResultArray** Funktionen abgefragt werden können.

Rampenbefehle können derzeit nicht innerhalb von Sweep-Schleifen verwendet werden.

Die Verwendung der Rampenbefehle ist in den Beispielen 3 und 4 erläutert.

Fehlerbehandlung

Alle Befehle liefern einen Rückgabewert, der Auskunft darüber gibt, ob der Befehl korrekt durchgeführt wurde. Ein Fehler bedeutet entweder, dass (a) das Testsystem oder die angesprochenene Einheit nicht vorhanden oder die Kommunikation nicht möglich ist, (b) die

Hardware bereits von einem anderen Testprogramm verwendet wird, (c) ungültige Parameter für den Befehl angegeben wurden oder (d) eine falsche Befehlsreihenfolge angegeben wurde.

Der Rückgabewert sollte von der Anwendungssoftware überprüft werden. In den Beispielprogrammen wird aus Gründen der Übersichtlichkeit diese Überprüfung nicht durchgeführt.

In den meisten Fällen werden Fehler im Errorlog abgespeichert. Das Errorlog wird im Installationsordner (*c:\Programme\mbTester\Log*) als Textdatei abgespeichert.

Wenn der Ordner *Debug* im Installationsverzeichnis (*c:\Programme\mbTester\Debug*) angelegt wird, werden dort die zum Testsysterm geschickten Befehle im Klartext geloggt. Das Verzeichnis sollte allerdings nach der Fehlersuche wieder gelöscht werden, da diese Dateien unter Umständen sehr groß werden können.

Ein Testprogramm, das mit **Start** gestartet wurde, kann jederzeit mit dem **Pause**-Befehl gestoppt werden. Das Testprogramm wird mit einem neuen **Start**-Befehl wieder fortgesetzt, oder kann mit dem **Clear** oder **Reset**-Befehl zurückgesetzt werden. Mit **Execute** ist dieser Mechanismus nicht möglich, weil der Befehl ja erst verlassen wird, wenn das Messprogramm bereits beendet wurde.

Beispiele

Die nachfolgenden Beispiele werden als Microsoft© Visual Basic® Program mitgeliefert und sind nach der Installation unter *c:\Programme\mbTester\Example* zu finden.

Beispiel #1

Dieses Programm stellt bei einer Source-Measure-Unit eine Spannung ein und misst den Strom. Nach der Messung wird die Spannung abgeschaltet.

```
Dim o As New mbInterface          ' dll access
Dim r As Long                     ' reference number
Dim a As Double                   ' measured current

Const v = 5

o.Source 1                        ' use smu #1
o.VoltageRange 0                  ' use autoranging
o.CurrentRange 0
o.ForceVoltage v                  ' force voltage
r = o.MeasureCurrent()            ' mind reference number
o.Execute                         ' execute test program
a = o.Result(r)                   ' get current measurements
o.Reset                           ' reset test system

Print "Voltage = " & v & "V"
Print "Current = " & a & "A"
```

Beispiel #2

Dieses Testprogramm misst die Eingangskennlinie eines MOS-Transistors. Am Drain (SMU 1) werden 0.1 V angelegt. Am Gate (SMU 2) werden Spannungen von 0 bis 2 V eingestellt und die dazugehörigen Ströme am Drain gemessen.

Dieses Beispiel zeigt auch, wie auf das Ende des Testprogramms gewartet wird, ohne die Benutzeroberfläche zu blockieren.

```
Dim o As New mbInterface          ' dll access

Const vd = 0.1                    ' drain voltage
Const vg1 = 0                     ' min gate voltage
Const vg2 = 2                     ' max gate voltage
Const n = 101                     ' number of points

Dim vg(n) As Double               ' array of voltage values
Dim id(n) As Double               ' array of current values
Dim r(n) As Long                  ' current reference numbers
Dim i As Integer                  ' counter variable

' drain
o.Source 1                        ' smu #1
o.VoltageRange 0                  ' use autoranging
o.CurrentRange 0

' gate
o.Source 2                        ' smu #2
o.Output 1                        ' output on
o.VoltageRange 0                  ' use autoranging
o.CurrentRange 0

o.Source 1
o.ForceVoltage vd                 ' force drain voltage

For i = 1 To n
  vg(i) = (vg2 - vg1) / CDbl(n - 1) * CDbl(i - 1) + vg1

  o.Source 2                      ' smu #1
  o.ForceVoltage vg(i)            ' force gate voltage

  o.Source 1                      ' smu #2
  r(i) = o.MeasureCurrent()       ' measure drain current
Next

o.Start                           ' start test program
Do While o.Busy()                 ' wait until finished
  DoEvents                        ' do other stuff
Loop

For i = 1 To n
  id(i) = o.Result(r(i))          ' get current measurements
Next

o.Reset                           ' reset test system

For i = 1 To n
  Print "vg=" & vg(i) & "  id=" & id(i)
Next
```

Beispiel #3

Dieses Testprogramm verwendet die Rampenfunktionen um die Ausgangsspannung in 500 ms auf 10 V zu bringen, danach bleibt die Spannung für 250 ms auf dieser Spannung und geht dann wieder in 300 ms auf 0 V zurück. Während dieser Zeit werden alle 100 ms Spannung, Strom und Zeit gemessen. Das Beispiel zeigt auch, wie Messergebnisse während eine laufenden Messung abgefragt werden können.

```
  Dim o As New mbInterface          ' dll access
  Dim r1 As Long                    ' voltage reference
  Dim r2 As Long                    ' current reference
  Dim i As Long                     ' counter
  Dim n As Long                     ' number of measurements
  Dim v As Double                   ' voltage
  Dim a As Double                   ' current

  o.Source 1                        ' use smu #1
  o.VoltageRange 0                  ' use autoranging for voltage
  o.CurrentRange 0                  ' and current
  o.CurrentLimit 0.1                ' 100 mA current limit
  o.ForceVoltage 0                  ' start voltage

  o.ResetTime                       ' reset timer
' o.RampRepeat 5                    ' uncomment do make 5 ramps !!
  o.RampMeasure 0.1                 ' do measurements everey 100 ms
  o.RampVoltage 10, 0.5             ' ramp to 10 V in 500 ms
  o.RampVoltage 10, 0.25            ' stay at 10 V for 250 ms
  o.RampVoltage 0, 0.3              ' ramp do 0 V in 300 ms
  o.RampStart                       ' start ramp

  r1 = o.RampResult(1)              ' voltage
  r2 = o.RampResult(2)              ' current

  o.Start                           ' now start the measurement

  n = o.Size(r1)                    ' number of measurements
  i = 0

  Do While o.Busy()                 ' wait until program is finished
    If i < n Then
      If o.Available(r1, i + 1) Then
        i = i + 1
        v = o.Result(r1, i)         ' we just print the results
        a = o.Result(r2, i)
        Print v, a
        DoEvents
      End If
    End If
  Loop

  Do While i < n                    ' the testprogramm ist finished here
    i = i + 1                       ' but some result may still be missing
    v = o.Result(r1, i)
    a = o.Result(r2, i)
    Print v, a
    DoEvents
  Loop

  o.Reset                           ' finished
```

Beispiel #4

Dieses Testprogramm verwendet die Rampenfunktionen, um einen Hot-Carrier-Stress bei einem CMOS-Transistor durchzuführen. Drain wird für den Stress auf 7 V gelegt, Gate auf 2 V. Die Stressdauer ist 1 Minute. Die Spannungen werden innerhalb einer Sekunde auf die Stressbedingung hochgefahren und am Ende des Stresses in einer Sekunde wieder zurück auf 0 V. Während der Stressdauer wird am Drain alle 100 ms gemessen und das Experiment wird beendet, falls bei der Drain-SMU eine Strombegrenzung festgestellt wird.

```
Dim o As New mbInterface          ' dll access
Dim r As Long                     ' reference number
Dim i As Long                     ' counter
Dim n As Long                     ' number of measurements
Dim s As Integer                  ' source status

Const vd = 7                      ' drain stress voltage
Const vg = 2                      ' gate stress voltage
Const ts = 60                     ' stress time
Const tr = 1                      ' ramp time
Const mr = 0.1                    ' measure rate

' drain
o.Source 1                        ' use smu #1
o.VoltageRange 3                  ' 20V range
o.CurrentRange 8                  ' 10mA range
o.CurrentLimit 0.01               ' 10 mA current limit
o.ForceVoltage 0                  ' start voltage

' gate
o.Source 2                        ' use smu #2
o.VoltageRange 3                  ' 20V range
o.CurrentRange 4                  ' 1uA range
o.CurrentLimit 0.000001           ' 1uA current limit
o.ForceVoltage 0                  ' start voltage

' drain stress
o.Source 1                        ' use smu #1
o.RampMeasure mr                  ' measurements rate
o.RampVoltage vd, tr              ' ramp drain voltage
o.RampVoltage vd, ts              ' drain stress voltage
o.RampVoltage 0, tr               ' ramp down
o.RampStart                       ' start ramp

r = o.RampResult(2)               ' current reference

' gate stress
o.Source 2                        ' use smu #1
o.RampVoltage vg, tr              ' ramp gate voltage
o.RampVoltage vg, ts              ' gate stress voltage
o.RampVoltage 0, tr               ' ramp do 0 V
o.RampStart                       ' start ramp

o.Start                           ' now start the measurement

n = o.Size(r)                     ' number of measurements
i = 0

Do While o.Busy()                 ' wait until program is finished
  If i < n Then
    If o.Available(r, i + 1) Then
      i = i + 1
      s = o.Status(r, i)
      If s <> 1 Then Exit Do      ' 1=voltage, 3/4=compliance
    End If
  End If
Loop

o.Reset                           ' finished
```

Beispiel #5

Dieses Testprogramm misst das Ausgangskennlinienfeld eines MOS-Transistors mit Hilfe zweier verschachtelter Sweep-Funktionen. In der äußeren Schleife wird die Gatespannung in 1 V Schritten erhöht und in der inneren Schleife dann die Drainspannung in 0.1 V Schritten. Ab der ersten Sweep-Funktion werden sämtliche nachfolgende Messbefehle bis zur *Execute* Funktion mit der Anzahl der Spannungsschritte wiederholt. Die Funktionen *MeasureVoltage* und *MeasureCurrent* definieren die Messungen, liefern in diesem Fall jedoch nicht eine Referenz auf den einzelnen Messwert sondern auf ein Array aller Messwerte. Mit *ResultArray()* werden alle Messwerte abgeholt. Die Anzahl der Messwerte wird von der Funktion *Size* zurückgeliefert.

```
Dim o As New mbInterface          ' dll access

Const vg1 = 0                     ' start gate voltage
Const vg2 = 5                     ' stop gate voltage
Const vgs = 1                     ' gate voltage steps
Const vd1 = 0                     ' start drain voltage
Const vd2 = 5                     ' stop drain voltage
Const vds = 0.1                   ' drain voltage steps

Dim r1 As Long                    ' reference number für drain voltage array
Dim r2 As Long                    ' reference number für drain current array

Dim vd() As Double                ' drain voltage measurements
Dim id() As Double                ' drain current measurements

Dim n As Integer                  ' number of measurements
Dim i As Integer                  ' counter variable

o.Source 2                        ' smu #2 is gate
o.VoltageRange 0                  ' use autoranging
o.CurrentRange 0

o.Source 1                        ' smu #1 is drain
o.VoltageRange 0                  ' use autoranging
o.CurrentRange 0

o.Source 2
o.SweepVoltage vg1, vg2, vgs      ' define gate voltage steps (outer loop)

o.Source 1
o.SweepVoltage vd1, vd2, vds      ' define drain voltage steps (inner loop)

r1 = o.MeasureVoltage             ' reference number für drain voltage measurement array
r2 = o.MeasureCurrent             ' reference number für drain current measurement array

o.Execute                         ' start test program

o.ResultArray r1, vd              ' get all results in the array
o.ResultArray r2, id

n = o.Size(r1)

o.Reset                           ' reset test system

For i = 1 To n
  Print "vd=" & vd(i) & "  id=" & id(i)
Next
```

Beispiel #6

Dieses Programm ist ein Beispiel für einen Messablauf beim Package-Level-Testsystem. Es werden bei einem Bauteil 3 Stress- und Mess-Zyklen durchgeführt. Zunächst werden die Verbindungen und Spannungen am Stress-Bus definiert und der erste Stress-Zyklus gestartet. Vor dem Umschalten auf den Mess-Bus müssen dort alle Verbindungen und Spannungen gleich eingestellt werden. Danach wird das Bauteil umgeschaltet und einige Messungen durchgeführt. Meist werden dazu auch einige Relais- Verbindungen am Mess-Bus geändert und weitere SMUs benötigt. Das Zurückschalten auf den Stress-Bus erfolgt analog. Vor dem nächsten Stress-Zyklus wird das Programm ausgeführt und die Messergebnisse verarbeitet. Der gesamte Vorgang wird für jeden Zyklus wiederholt.

```
  Dim o As New mbInterface          ' dll access
  Dim n As Integer
  Dim r1 As Long
  Dim r2 As Long
  Dim v1 As Double
  Dim v2 As Double

  o.Configuration 3                 ' package level test system

' stress definitions
  o.Relay 1, 1                      ' pin 1 = smu 1
  o.Relay 6, 2                      ' pin 2 = gnd
  o.Channel 1, 1, 2                 ' connect oven 1 package 1 to stress bus
  o.Source 1
  o.Output 1
  o.ForceVoltage 5                  ' stress voltage on pin 1

' stress cycle
  For n = 1 To 3                    ' do 3 stress/measurement cycles
    o.Delay 5                       ' stress time (s)

' switch oven 1 package 1 to measure bus
    o.Relay 13, 1                   ' same connections and
    o.Relay 18, 2                   ' voltages on measure bus
    o.Source 13
    o.Output 1
    o.ForceVoltage 5
    o.Channel 1, 1, 1               ' switch bus

' now do some measurements
    o.Source 13
    r1 = o.MeasureCurrent           ' measure current
    o.RelayOpen 13, 1               ' open connections
    o.RelayOpen 18, 2
    o.Relay 14, 1                   ' use SMU 14, 15 to
    o.Relay 15, 2                   ' make an additional
    o.Source 14                     ' measurement
    o.Output 1
    o.ForceVoltage 1
    o.Source 15
    o.Output 1
    r2 = o.MeasureCurrent
    o.RelayOpen 14, 1               ' open connections
    o.RelayOpen 15, 2

' switch back to stress bus
    o.Relay 13, 1                   ' same connections and
    o.Relay 18, 2                   ' voltages on measure bus
    o.Source 13
    o.ForceVoltage 5
    o.Channel 1, 1, 2               ' switch bus

' execute one cycle
    o.Execute
    v1 = o.Result(r1)               ' get measurements
    v2 = o.Result(r2)
  Next

  o.Reset
```

Befehle

Available

Mit Hilfe dieser Funktion kann bei laufendem Testprogramm abgefragt werden, ob ein Messergebnis bereits zur Verfügung steht.

Syntax :
obj.**Available** (Ref, Index)

Parameter:

long Ref	Referenznummer
int Index	Index auf einen einzelnen Messwert bei Arrays (optional)

Rückgabe:

int	0 = Ergebnis nicht verfügbar 1 = Ergebnis verfügbar <0 = Fehlercode

Anmerkung:
Wenn das Testprogramm beendet wurde, stehen auch gleichzeitig sämtliche Messergebnisse zur Verfügung. Diese Funktion liefert dann für alle Messergebnisse der Wert 1.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer **SweepVoltage** oder **SweepCurrent-**Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife gemessen werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben.

Um einen einzelnen Messwert auf Verfügbarkeit zu prüfen, ist in der Funktion **Available** als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Wenn der Index-Parameter weggelassen oder auf 0 gesetzt wird, wird das gesamte Messwerte-Array überprüft und nur dann 1 zurückgegeben wenn alle Messwerte verfügbar sind.

Alle Ergebnisse und die dazugehörigen **Status**-Werte bleiben gespeichert und werden erst mit einem **Reset** oder **Clear** gelöscht.

Average

Dieser Befehl stellt für eine SMU den Averaging-Modus für nachfolgende Messungen ein.

Syntax:
obj.**Average** (Mode)

Parameter:

int Mode	1 = *Fast*
	2 = *Normal*
	3 = *Long*

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Mit der Averaging-Einstellung kann der Schwerpunkt entweder auf Messgeschwindigkeit oder Messgenauigkeit gelegt werden oder einen Kompromiss zwischen beiden.

Fast ergibt die höchste Messgeschwindigkeit mit kleinen Messungenauigkeiten sowie limitierter Stromauflösung in den kleinen Messbereichen. Dieser Modus ist für schnelle Charakterisierungen und Kennlinienfelder geeignet. *Fast* ist ungefähr doppelt so schnell wie der *Normal*-Modus.

Normal stellt einen guten Kompromiss dar und sollte für die meisten Messungen verwendet werden.

Long sollte verwendet werden, wenn eine hohe Messgenauigkeit und/oder eine gute Messauflösung in den kleinsten Strombereichen benötigt werden. In diesem Modus wird die Spezifikationsgenauigkeit garantiert, allerdings benötigen die Messungen etwa 2-3 mal länger als im *Normal*-Modus.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Board

Dieser Befehl definiert das DUT-Board für ein Reliability-Testsystem.

Syntax:
obj.**Board** (Number)

Parameter:
int Number DUT-Board (1..64)

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Das Board kann nicht geändert werden, wenn ein Testprogramm definiert wurde oder gerade ausgeführt wird.

Bei den DUT-Boards eines Reliability-Testsystems gibt es die Besonderheit, dass sich jeweils 4 SMUs zwei DUT-Boards teilen. Für den Fall, dass pro Board nur 2 SMUs verwendet werden, können beide Boards gleichzeitig parallel betrieben werden. Es sind dann 2 *mbInterface*-Objekte erforderlich. Diese beiden Testprogramme laufen vollkommen unabhängig und dieMessungen beeinflussen sich gegenseitig nicht. Alternativ kann jedes der beiden Boards auch bis zu 4 SMUs verwenden, ein gleichzeitiger Betrieb ist dann allerdings nicht mehr möglich.

Für Parameter-Analyzer wird dieser Befehl nicht benötigt.

Busy

Mit Hilfe dieser Funktion kann überpüft werden, ob das Messprogramm noch läuft.

Syntax:
obj.**Busy** ()

Rückgabe:

int	0 = Programm beendet
	1 = Programm läuft
	<0 = Fehlercode

Anmerkung:
Diese Funktion sollte verwendet werden um das Ende eines Testprogrammes zu erkennen, das mit dem **Start**-Befehl gestartet wurde. Während das Testprogramm läuft, können bereits fertiggestellte Messergebnisse ausgewertet werden. Um festzustellen, welche Messergebnisse bereits vorhanden sind, kann die Funktion **Available** verwendet werden. Wenn das Testprogramm fertig ist, sind auch gleichzeitig sämtliche Messergebnisse verfügbar.

Ein laufendes Testprogramm kann jederzeit mit dem **Pause**-Befehl unterbrochen oder mit den **Clear**- und **Reset**-Befehlen gestoppt werden.

Der **Execute**-Befehl startet das Testprogramm und wartet solange bis es beendet wurde.

Channel

Dieser Befehl schaltet bei Package-Level-Testern ein einzelnes Package entweder auf den Mess- oder Stress-Bus.

Syntax:
obj.**Channel** (Oven, Package, Bus)

Parameter:

int Oven	1 .. 6
int Package	1 .. 4
int Bus	0 = nicht verwendet, 1 = Mess-Bus, 2=Stress-Bus

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl kann nur bei Package-Level-Testsystemen verwendet werden (siehe **Configuration**)

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl ausgeführt.

Clear

Dieser Befehl stoppt ein laufendes Testprogramm und löscht sowohl den Programm- als auch den Ergebnisspeicher.

Syntax:
obj.**Clear** ()

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl ist ähnlich zum **Reset**-Befehl, schaltet jedoch die Source-Measure-Units und die Multiplexer-Relais nicht ab.

Configuration

Dieser Befehl definiert die Systemkonfiguration des Testsystems.

Syntax:
obj.**Configuration** (Mode)

Parameter :

int Mode	1 = Parameter-Analyzer (Default)
	2 = Reliability-Testsystem
	3 = Package-Level-Testsystem

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Falls ein Reliability- oder Package-Level-Testsystem verwendet wird, muss dieser Befehl am Anfang des Testprogramms stehen. Für Parameter-Analyzer ist dieser Befehl nicht notwendig.

Die Systemkonfiguration kann nicht geändert werden, wenn ein Testprogramm definiert wurde oder gerade ausgeführt wird.

CurrentLimit

Dieser Befehl stellt die Strombegrenzung für eine Source-Measure-Unit ein, die als Spanungsquelle verwendet wird.

Syntax:
obj.**CurrentLimit** (Limit)

Parameter:

double Limit	Strombegrenzung

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Die eingestellte Strombegrenzung wird erst aktiv, wenn die Source-Measure-Unit als Spannungsquelle verwendet wird.

Die Strombegrenzung gilt für beide Polaritäten, d.h. wenn z.B. 1 mA eingestellt wird, wird der Ausgangsstrom auf –1 mA bis +1 mA begrenzt. Die Strombegrenzung hat in jedem Fall Vorrang gegenüber der eingestellten Spannung, je nach Anwendung kann unter Umständen auch eine höhere Ausgangsspannung als die mit **ForceVoltage** eingestellte anliegen.

Der maximale Ausgangsstrom hängt vom ausgewählten Strombereich (falls nicht Auto-Range) und vom Source-Measure-Unit-Modell ab.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

CurrentRange

Diese Befehl wählt einen von 10 Strombereichen oder Auto-Ranging.

Syntax:
obj.**CurrentRange** (Range)

Parameter:

int Range	0 = Auto
	1 = 1 nA
	2 = 10 n
	3 = 100 nA
	4 = 1 uA
	5 = 10 uA
	6 = 100 uA
	7 = 1 mA
	8 = 10 mA
	9 = 100 mA
	10 = 1 A

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Mit Auto-Ranging wird automatisch der beste Strombereich für jeden Messpunkt ausgewählt, allerdings kann das Vorgeben eines Bereiches eine deutliche Verbesserung der Messgeschwindigkeit bewirken. Im Auto-Range Modus kann mit Hilfe des **LowestCurrentRange**-Befehls eine Einschränkung der zulässigen Strombereiche erfolgen.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

CurrentRangeMaximum

Diese Funktion gibt den maximal möglichen Ausgangsstrom der SMU zurück.

Syntax:
obj.**CurrentRangeMaximum** (Range)

Parameter:

int Range	Strombereich (0 oder 1 .. obj.CurrentRanges**)**
	wenn 0 angegeben wird, wird der maximale Strom des letzten Strombereiches zurückgegeben

Rückgabe:

double	Maximal möglicher Ausgangsstrom

Anmerkung:
Mit der **CurrentRanges**-Funktion kann die Anzahl der Strombereiche abgefragt werden.

Abhängig vom SMU-Modell kann der maximal möglich Ausgangsstrom bei höheren Ausgangsspannungen kleiner sein als diese Funktion zurückgibt. Für Details siehe die Hardwaredokumentation.

Dieser Befehl wird sofort ausgeführt.

CurrentRanges

Diese Funktion gibt die Anzahl der Strombereiche der SMU zurück.

Syntax:
obj.**CurrentRanges**

Rückgabe:
int Anzahl der Strombereiche

Anmerkung:
Mit der **CurrentRangeMaximum**-Funktion kann für jeden Bereich der maximale Ausgangsstrom abgefragt werden.

Dieser Befehl wird sofort ausgeführt.

CurrentToRange

Diese Befehl erwartet einen Stromwert und gibt den kleinsten passenden Strombereich zurück.

Syntax:
obj.**CurrentToRange** (Value)

Parameter:

double Value	Strom

Rückgabe:

int	1 = 1 nA 2 = 10 n 3 = 100 nA 4 = 1 uA 5 = 10 uA 6 = 100 uA 7 = 1 mA 8 = 10 mA 9 = 100 mA 10 = 1 A ERROR_CURRENT (unzulässiger Strom)

Anmerkung:
Dieser Befehl wird sofort ausgeführt.

Delay

Dieser Befehl verzögert den Programmablauf.

Syntax:
obj.**Delay** (Time)

Parameter:

double Time	Wartezeit (s)

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
In den meisten Fällen sind Verzögerungen im Messprogramm nicht notwendig, weil die Source-Measure-Unit nach dem Einstellen einer Spannung oder eines Stromes ohnehin wartet, bis der eingestellte Wert stabil ist. Dies gilt auch, wenn mehrere SMUs an einer Messung beteiligt sind.

Die Wartezeit kann mit 1 ms Auflösung eingestellt werden. Die maximale Wartezeit ist 2^{31} ms.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Execute

Dieser Befehl startet ein Testprogramm und wartet solange, bis das Programm beendet wurde.

Syntax:
obj.**Execute** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Execute sollte nur für kurze Testprogramme verwendet werden, weil das Anwendungsprogamm während der Ausführung unter Umständen blockiert ist.

Wenn das Testprogramm nach eine Minute noch nicht beendet wurde, wird das Programm gestoppt und der Befehl mit einem Fehler verlassen.

Ein **Execute**-Befehl schließt alle offenen Schleifen, die durch **SweepVoltage** oder **SweepCurrent** begonnen wurden. Ein **SweepEnd**-Befehl ist in diesem Fall nicht erforderlich.

Alternativ können die Befehle **Start** and **Busy** verwendet werden.

ForceCurrent

Dieser Befehl stellt einen Strom ein.

Syntax:
obj.**ForceCurrent** (Current)

Parameter:
double Current Ausgangsstrom

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Mit diesem Befehl wird die Source-Measure-Unit auf Stromquelle geschaltet, falls sie zuvor als Spannungsquelle verwendet wurde. Die mit dem Befehl **VoltageLimit** eingestellte Spannungsbegrenzung gilt für beide Polaritäten.

Nach dem Einstellen wartet die SMU bis der eingestellte Wert stabil ist. Es ist daher unter normalen Umständen keine zusätzliche Verzögerung notwendig.

Der maximale Ausgangsstrom hängt vom ausgewählten Strombereich (falls nicht Auto-Range) und vom Source-Measure-Unit-Modell ab.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

ForceVoltage

Dieser Befehl stellt eine Spannung ein.

Syntax:
obj.**ForceVoltage** (Voltage)

Parameter:
double Voltage Ausgangsspannung

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Mit diesem Befehl wird die Source-Measure-Unit auf Spannungsquelle geschaltet, falls sie zuvor als Stromquelle verwendet wurde. Die mit dem Befehl **CurrentLimit** eingestellte Strombegrenzung gilt für beide Polaritäten.

Nach dem Einstellen wartet die SMU bis der eingestellte Wert stabil ist. Es ist daher unter normalen Umständen keine zusätzliche Verzögerung notwendig.

Die maximale Ausgangsspannung hängt vom ausgewählten Spannungsbereich (falls nicht Auto-Range) und vom Source-Measure-Unit-Modell ab.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

LowestCurrentRange

Dieser Befehl stellt den kleinstmöglichen Strombereich ein, der bei Auto-Ranging verwendet wird.

Syntax:
obj.**LowestCurrentRange** (Range)

Parameter:

int Range	1 = 1 nA
	2 = 10 n
	3 = 100 nA
	4 = 1 uA
	5 = 10 uA
	6 = 100 uA
	7 = 1 mA
	8 = 10 mA
	9 = 100 mA
	10 = 1 A

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Die Angabe eines kleinsten Strombereiches kann die Messgeschwindigkeit für Auto-Ranging deutlich erhöhen, weil Messungen in kleinen Bereichen meist wesentlich länger dauern als in höheren.

Dieser Befehlt wird erst aktiv, wenn Auto-Range (0) im **CurrentRange**-Befehl ausgewählt wird.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

LowestVoltageRange

Dieser Befehl stellt den kleinstmöglichen Spannungsbereich ein, der bei Auto-Ranging verwendet wird.

Syntax:
obj.**LowestVoltageRange** (Range)

Parameter:

int Range	1 = 200 mV
	2 = 2 V
	3 = 20 V
	4 = 200 V

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Die Angabe eines kleinsten Spannungsbereiches kann die Messgeschwindigkeit für Auto-Ranging deutlich erhöhen, weil Messungen in kleinen Bereichen meist wesentlich länger dauern als in höheren.

Dieser Befehlt wird erst aktiv, wenn Auto-Range (0) im **VoltageRange**-Befehl ausgewählt wird.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

MeasureCapacitance

Dieser Befehl misst die Kapazität einer angeschlossenen Teststruktur.

Syntax:
obj.MeasureCapacitance (Range, Amplitude)

Parameter:

int Range	0=Auto, 1=1pF, 2=10pF, 3=100pF, 4=1nA, 5=10nF, 6=100nF, 7=1uF, 8=10uF, 9=100uF, 10=1mF, 11=10mF, 12=100mF, 13=1F
double Amplitude	Amplitude des Messsignals (10 mV .. 100V)

Rückgabe:

double	Kapazität (F) 0 = Messung kann nicht durchgeführt werden

Beispiel:

```
double cap;

ForceVoltage(5.);                          ' bias voltage 5V */
cap = MeasureCapacitance(0, 0.1);          ` 100 mV amplitude */
```

Anmerkung:
Die Messspannung ist die vorher z.B. mit **ForceVoltage** anliegende Spannung ± die angegebene Amplitude, in dem obige Beispiel also ungefähr 4,9 V .. 5,1 V.

Falls nicht Auto-Range gewählt wird: Aufgrund des Messprinzips gibt der Bereich nicht das Maximum sonder das Minimum des Messbereiches an, d.h. im 1 nF Bereich kann man von 1 nF bis etwa 100 nF messen.

Für kleine Kapaziätswerte wird empfohlen, die Offsetkapazität zu bestimmen und vom Messwert abzuziehen.

Dieser Befehl wird unmittelbar ausgeführt. Wenn zu diesem Zeitpunkt ein Testprogramm bereits definiert aber noch nicht gestartet wurde, wird dieses zuerst ausgeführt.

MeasureCurrent

Dieser Befehlt führt eine Spannungsmessung aus.

Syntax:
obj.**MeasureCurrent** ()

Rückgabe:
long Referenznummer (> 0) oder Fehlercode

Anmerkung:
Wenn dieser Befehl im Testprogramm definiert wird, wird eine eindeutige Referenznummer zurückgegeben, mit der später mit der **Result**-Funktion das eigentliche Messergebnis abgefragt werden werden kann. Mit derselben Referenznummer kann mit Hilfe der **Available**-Funktion bei laufendem Testprogramm abgefragt werden, ob das Messergebnis bereits zur Verfügung steht.

Wenn der Befehl **MeasureCurrent** im Inneren einer **SweepVoltage** oder **SweepCurrent** Schleife verwendet wird, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

MeasureTime

Dieser Befehlt speichert den SMU-internen Timerwert.

Syntax:
obj.**MeasureTime** ()

Rückgabe:
long Referenznummer (> 0) oder Fehlercode

Anmerkung:
Wenn dieer Befehl im Testprogramm definiert wird, wird eine eindeutige Referenznummer zurückgegeben, mit der später mit der **Result**-Funktion das eigentliche Messergebnis abgefragt werden werden kann. Mit derselben Referenznummer kann mit Hilfe der **Available**-Funktion bei laufendem Testprogramm abgefragt werden, ob das Messergebnis bereits zur Verfügung steht.

Wenn der Befehl **MeasureTime** im Inneren einer Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Jede Source-Measure-Unit beinhaltet einen genauen Timer mit Millisekundenauflösung. Der Zähler wird beim Einschalten des Testsystems zurückgesetzt oder im Testprogramm mit dem Befehl **ResetTime**.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

MeasureVoltage

Dieser Befehlt führt eine Spannungsmessung aus.

Syntax:
obj.**MeasureVoltage** ()

Rückgabe:
long Referenznummer (> 0) oder Fehlercode

Anmerkung:
Wenn dieser Befehl im Testprogramm definiert wird, wird eine eindeutige Referenznummer zurückgegeben, mit der später mit der **Result**-Funktion das eigentliche Messergebnis abgefragt werden werden kann. Mit derselben Referenznummer kann mit Hilfe der **Available**-Funktion bei laufendem Testprogramm abgefragt werden, ob das Messergebnis bereits zur Verfügung steht.

Wenn der Befehle **MeasureVoltage** im Inneren einer Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Output

Dieser Befehl schaltet das Ausgangsrelais der Source-Measure-Unit.

Syntax:
obj.**Output** (Mode)

Parameter:

int Mode	0 = Aus
	1 = Ein (Default)
	2 = Kalibrierausgang

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Beim ersten Auswählen einer Source-Measure-Unit wird das Ausgangsrelais automatisch eingeschaltet. Wenn nicht ein anderer Zustand gewünscht ist, ist ein eigener **Output**-Befehl daher nicht notwendig. Am Ende des Testprogrammes kann das Ausgangsrelais mit diesem Befehl wieder abgeschaltet werden, einfacher ist es allerdings, den **Reset**-Befehl zu verwenden, der gleich sämtliche Source-Measure-Units abschaltet.

Nach dem Einschalten des Ausgangs wird die Source-Measure-Unit auf folgende Standardeinstellungen gesetzt:

Average1 (Fast)
VoltageRange4 (200 V)
CurrentRange *)9 (100 mA)
LowestVoltageRage1 (200 mV)
LowestCurrentRange1 (1 nA)
ForceVoltage0 V
CurrentLimit *)100 mA

*) Für Package-Level-Testsysteme wird als CurrentRange 7 (1 mA) und als CurrentLimit 1 mA eingestellt.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Pause

Dieser Befehl unterbricht ein laufendes Testprogramm.

Syntax:
obj.**Break** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Mit einem erneuten **Start**-Befehl kann das Programm wieder fortgesetzt oder mit einem **Clear**-Befehl endgültig abgebrochen werden.

RampCurrent

Dieser Befehl definiert eine Stromrampe.

Syntax:
obj.**RampCurrent** (Current, Time)

Parameter:

double Current	Strom der erreicht werden soll
double Time	Zeit in der der Strom erreicht werden soll (s)

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl definiert eine lineare Stromrampe ausgehend vom zuletzt eingestellten Strom. Die SMU muss sich in Strommodus befinden; am besten definiert man den Startpunkt mit Hilfe eines **ForceCurrent**-Befehls.

Wenn vorher ein **RampMeasure-**Befehl angegeben wurde, werden während dieser Rampe mit der vordefinierten Messrate Spannungs-, Strom- und Zeitmessungen durchgeführt. In den meisten Fällen werden mehrere Stromrampen definiert, für die dann dieselbe Messrate gilt, sofern sie nicht mit einem neuen **RampMeasure-**Befehl geändert wird. Das Rampenmuster wird mit dem Befehl **RampStart** gestartet. Wenn zuvor ein **RampRepeat**-Befehl angegeben wurde, kann das gesamte Rampenmuster bis zu einer Million mal wiederholt werden.

Wenn nach dem **RampStart**-Befehl weitere Rampenbefehle für dieselbe SMU angegeben werden, werden diese gepuffert und ausgeführt, sobald das erste Rampenmuster abgeschlossen wurde. Wenn nach dem **RampStart**-Befehl weitere Rampenbefehle für eine andere SMU angegeben werden, werden diese sofort ausgeführt, d.h. die Rampen laufen auf den beiden SMUs dann parallel. Jeder andere Befehl wartet erst das Ende des kompletten Rampenmusters ab, bis das Testprogramm fortgesetzt wird.

Ein einzelnes Rampenmuster darf aus bis zu 100 einzelnen Rampenschritten bestehen.

Die Rampenzeit kann mit einer Auflösung von 100 µs eingestellt werden. Die minimale Zeit ist 200 µs wenn Spannungs- und Strombereiche fix sind, ansonsten 10 ms. Die maximale Zeit ist 200.000 Sekunden bzw. ca. 2 Tage.

Siehe auch Beispiele #3 und #4.

RampMeasure

Dieser Befehl definiert eine Messrate für nachfolgende Strom- und Spannungsrampen.

Syntax:
obj.**RampMeasure** (Rate)

Parameter:

double Rate	Messrate (s) 0 = keine Messung

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Wenn eine Messrate ungleich 0 angegeben wird, werden während der nachfolgenden Rampen Spannungs-, Strom- und Zeitmessungen durchgeführt. Die Messrate gilt für alle nachfolgenden Rampen, kann aber auch von Rampe zu Rampe geändert werden.

Das Rampenmuster wird mit dem Befehl **RampStart** gestartet, der für alle vorher angegebenen Rampenschritte die Anzahl der Messungen berechnet und diese Zahl zurückgibt. Mit der Funktion **RampResult** erhält man die Referenznummern für diese Messungen, mit denen man dann - sobald das Testprogramm gestartet wurde und die Messungen abgeschlossen sind - die Messergebnisse erhält.

Die Messrate kann länger oder kürzer sein als einzelne Rampenabschnitte. Die Zeiten werden in jedem Fall aufsummiert und wenn dieser Wert die aktuelle Messrate überschreitet, ein Messpunkt gespeichert.

Die Messrate kann mindestens 10 ms und maximal 200.000 s bzw. ca. 55 Stunden. Für alle definierten Rampenmuster pro SMU können maximal etwa 4000 Messpunkte (jeweils Spannung, Strom und Zeit) gespeichert werden.

Siehe auch Beispiele #3 und #4.

RampRepeat

Wenn dieser Befehl vor einem RampStart Befehl angegeben wird, wird das gesamte Rampenmuster mit der angebenen Zahl wiederholt.

Syntax:
obj.**RampRepeat** (Count)

Parameter:

long Count	Anzahl Wiederholungen (0 .. 1.000.000) 0 oder 1 = keine Wiederholung

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Es ist zu beachten, dass auch sämtliche Messungen, falls der Befehl **RampMeasure** verwendet wurde, wiederholt werden. Diesebezüglich gelten Einschränkungen bei der maximalen Anzahl von Messpunkten (siehe dort).

Siehe auch Beispiele #3 und #4.

RampResult

Diese Funktion gibt eine Referenznummer für die Messsdaten in einem Rampenmuster zurück.

Syntax:
obj.**RampResult** (Mode)

Parameter:

char Mode	1 = Spannung 2 = Strom 3 = Zeit

Rückgabe:

long	Referenznummer (> 0) oder Fehlercode

Anmerkung:
Während einer Rampenfunktion werden entsprechend der angegebenen Messrate Spannung, Strom und Zeit gespeichert.

Mit dieser Funktion erhält man eine Referenznummer für die Messergebnisse einer Rampenfunktion. Für Spannung, Strom und Zeit sind jeweils eigene Referenznummern abzufragen, wenn deren Messdaten benötigt werden.

Da die Anzahl der Messpunkte und deren Referenzen erst mit einem **RampStart** Befehl berechnet werden, kann diese Funktion auch erst nach einem **RampStart** Befehl aufgerufen werden.

Die Anzahl der Messpunkte kann mit der Funktion **Size** ermittelt werden.

Die Messwerte und die zugehörigen Statusinformationen können, sobald die Messung durchgeführt wurde, mit den Funktionen **Result**, **Status**, **ResultArray** oder **StatusArray** erhalten werden. Die Verfügbarkeit einzelner Messwerte kann mit **Available** überprüft werden.

Siehe auch Beispiele #3 und #4.

RampStart

Dieser Befehl startet alle vorher angegebenen Spannungs- oder Stromrampen.

Syntax:
obj.**RampStart** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Siehe auch Beispiele #3 und #4.

RampVoltage

Dieser Befehl definiert eine Spannungsrampe.

Syntax:
obj.**RampVoltage** (Voltage, Time)

Parameter:

double Voltage	Spannung die erreicht werden soll
double Time	Zeit in der die Spannung erreicht werden soll (s)

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl definiert eine lineare Spanungsrampe ausgehend von der zuletzt eingestellten Spannung. Die SMU muss sich in Spannungsmodus befinden; am besten definiert man den Startpunkt mit Hilfe eines **ForceVoltage**-Befehls.

Wenn vorher ein **RampMeasure-**Befehl angegeben wurde, werden während dieser Rampe mit der vordefinierten Messrate Spannungs-, Strom- und Zeitmessungen durchgeführt. In den meisten Fällen werden mehrere Spannungsrampen definiert, für die dann dieselbe Messrate gilt, sofern sie nicht mit einem neuen **RampMeasure-**Befehl geändert wird. Das Rampenmuster wird mit dem Befehl **RampStart** gestartet. Wenn zuvor ein **RampRepeat**-Befehl angegeben wurde, kann das gesamte Rampenmuster bis zu einer Million mal wiederholt werden.

Wenn nach dem **RampStart**-Befehl weitere Rampenbefehle für dieselbe SMU angegeben werden, werden diese gepuffert und ausgeführt, sobald das erste Rampenmuster abgeschlossen wurde. Wenn nach dem **RampStart**-Befehl weitere Rampenbefehle für eine andere SMU angegeben werden, werden diese sofort ausgeführt, d.h. die Rampen laufen auf den beiden SMUs dann parallel. Jeder andere Befehl wartet erst das Ende des kompletten Rampenmusters ab, bis das Testprogramm fortgesetzt wird.

Ein einzelnes Rampenmuster darf aus bis zu 100 einzelnen Rampenschritten bestehen.

Die Rampenzeit kann mit einer Auflösung von 100 µs eingestellt werden. Die minimale Zeit ist 200 µs wenn Spannungs- und Strombereiche fix sind, ansonsten 10 ms. Die maximale Zeit ist 200.000 Sekunden bzw. ca. 2 Tage.

Siehe auch Beispiele #3 und #4.

Relay

Dieser Befehl schließt ein Multiplexer-Relais.

Syntax:
obj.**Relay** (Input, Output, Display, Color)

Parameter:

int Input	Source-Nummer oder 0=Common
int Output	Multiplexer-Ausgang (für Reliability-Tester siehe Multiplexer-Dokumentation für Details)
int Display	Anzeige am Multiplexer (0=Default)
int Color	0 = Aus 1 = Grün 2 = Rot 3 = Gelb 5 = Grün blinkend 6 = Rot blinkend 7 = Gelb blinkend

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl kann nur verwendet werden, wenn ein Multiplexer im Testsystem vorhanden ist. Bei Reliability- und Package-Level-Testsystemen ist dies immer der Fall.

Input-Parameter: Für Package-Level-Testsysteme sind die Eingänge 1, 2, 3, 4, 5, 7, 8, 9, 10 und 11 den gleichnamigen Source-Measure-Units am Mess-Bus zugeordnet, die Nummern 13, 14, 15, 16, 17, 19, 20 und 21 den Source-Measure-Units am Stress-Bus. Die Eingänge 6, 12, 18 und 24 sind mit der gemeinsamen Masse aller Source-Measure-Units verbunden.

Bei Parameter-Analyzern und Reliability-Testern können Relais nur geschaltet werden wenn alle Source-Measure-Units ausgeschaltet sind. Am einfachsten wird das mit dem **Reset**-Befehl erreicht.

Bei Package-Level-Testsystemen können Relais auch unter Spannung geschaltet werden. Eine Voraussetzung dafür ist, dass bei der Source-Measure-Unit, die an den zu schaltenden Eingang angeschlossen ist, die Strombegrenzung auf maximal 1 mA eingestellt ist. Um Kurzschlüsse zwischen Source-Measure-Units zu verhindern, kann jeder Ausgang nur mit maximal einem Eingang verbunden werden.

Zum Öffnen der Relais sind die Befehle **RelayOpen** und **Reset** vorgesehen.

Der Parameter *Display* kann verwendet werden, um die Default-Anzeige am Multiplexer zu ändern. Das kann sinnvoll sein, um z.B. bei Probekarten die Device oder Padnummer anzuzeigen anstelle des Multiplexerausgangs.

RelayOpen

Dieser Befehl öffnet ein Multiplexer-Relais.

Syntax:
obj.**Relay** (Input, Output, Display, Color)

Parameter:

int Input	Source-Nummer oder 0=Common
int Output	Multiplexer-Ausgang (für Reliability-Tester siehe Multiplexer-Dokumentation für Details)
int Display	Anzeige am Multiplexer (0=Default)
int Color	0 = Aus 1 = Grün 2 = Rot 3 = Gelb 5 = Grün blinkend 6 = Rot blinkend 7 = Gelb blinkend

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl ist für Package-Level-Testsysteme vorgesehen Bei anderen Testsystemen kann zum Öffnen der Relais der **Reset**-Befehl verwendet werden.

Input-Parameter: Die Eingänge 1, 2, 3, 4, 5, 7, 8, 9, 10 und 11 sind den gleichnamigen Source-Measure-Units am Mess-Bus zugeordnet, die Nummern 13, 14, 15, 16, 17, 19, 20 und 21 den Source-Measure-Units am Stress-Bus. Die Eingänge 6, 12, 18 und 24 sind mit der gemeinsamen Masse aller Source-Measure-Units verbunden.

Relais können unter Spannung geschaltet werden. Eine Voraussetzung dafür ist, dass bei der Source-Measure-Unit, die an den zu schaltenden Eingang angeschlossen ist, die Strombegrenzung auf maximal 1 mA eingestellt ist.

Zum Schließen der Relais ist der Befehl **Relay** vorgesehen, Die Parameter *Display* und *Color* sind bei diesem Befehl für zukünftige Erweiterungen reserviert und werden derzeit nicht verwendet.

Reset

Dieser Befehl stoppt ein laufendes Testprogramm und setzt das Testsystem zurück.

Syntax:
obj.**Reset** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Wenn dieser Befehl ausgeführt wird, wird ein laufendes Testprogramm sofort gestoppt, die Source-Measure-Units abgeschaltet und die Multiplexer-Relais geöffnet. Die Programm- und Messdatenspeicher werden gelöscht, d.h. es können danach keine Messwerte mehr abgefragt werden.

ResetTime

Dieser Befehl setzt den internen Timer der Source-Measure-Unit zurück.

Syntax:
obj.**ResetTime** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Jede Source-Measure-Unit beinhaltet einen genauen Timer mit Millisekundenauflösung. Der Zähler beim Einschalten des Testsystems zurückgesetzt oder mit dem Befehl **ResetTime**.

Jede Source-Measure-Unit beinhaltet einen genauen Timer mit Millisekundenauflösung. Der Zähler wird beim Einschalten des Testsystems zurückgesetzt oder im Testprogramm mit dem Befehl **ResetTime**.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

Result

Diese Funktion gibt Spannungs-, Strom- oder Zeitmessungen zurück.

Syntax:
obj.**Result** (Ref, Index)

Parameter:

long Ref	Referenznummer
int Index	Index auf einen einzelnen Messwert bei Arrays (optional)

Rückgabe:

double	Messergebnis

Anmerkung:
Bei den Befehlen **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** wird zum Zeitpunkt der Definition im Testprogramm eine eindeutige Referenznummer zurückgegeben. Sobald das Testprogramm dann ausgeführt wird und die Messung erfolgt ist, kann das Messergebnis mit dieser Funktion und der Referenznummer abgefragt werden.

Die **Status**-Funktion gibt für Messung die Betriebsart der Source-Measure-Unit zum Zeitpunkt der Messung zurück, z.B. Strombegrenzung.

Mit Hilfe der **Available**-Funktion kann bei laufendem Testprogramm abgefragt werden, ob ein Messergebnis bereits verfügbar ist. Wenn das Testprogramm beendet wurde, stehen auch gleichzeitig sämtliche Messergebnisse zur Verfügung. Zum Abfragen der Messergebnisse ist es nicht erforderlich, eine bestimmte Reihenfolge einzuhalten. Alle Ergebnisse und die dazugehörigen **Status**-Werte bleiben gespeichert und werden erst mit einem **Reset** oder **Clear** gelöscht.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer **SweepVoltage** oder **SweepCurrent**-Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In der Funktion **Result** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben.

Beispiel:

```
Dim ref as Long
Dim amps as Double

obj.Source 1
obj.ForceVoltage 5                  ' smu 1 force 5V
ref = obj.MeasureCurrent            ' invoke current measurement

obj.Execute                         ' execute test program

amps = obj.Result(ref)              ' get measurement
```

ResultArray

Diese Funktion liefert alle Messwerte in einem Messwerte-Array.

Syntax:
obj.**Result** (Ref, Index)

Parameter:

long Ref	Referenznummer
double Array()	Array (wird von der Funktion neu dimensioniert falls erforderlich)

Rückgabe:

double Array()	Messwerte
int	0 = Ok oder Fehlercode

Anmerkung:
Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer **SweepVoltage** oder **SweepCurrent**-Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben.

Bei Verwendung der Rampenbefehle wird ebenfalls ein Messwerte-Array zurückgegeben.

Die Funktion **ResultArray** liefert die Messwerte für das gesamte Messwertarray, die Funktion **StatusArray** die dazugehörigen Statusinformationen.

Alternativ kann ein einzelner Messwerte in einem Messwertarray auch mit der Funktion **Result** abgefragt werden. In diesem Fall ist als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben.

Wenn das Messprogramm zum Zeitpunkt der Abfrage noch läuft, ist es möglich dass einzelne Messwerte im Array noch nicht verfügbar sind. Die Verfügbarkeit einzelner Messwerte oder des gesamten Arrays kann mit der Funktion **Available** geprüft werden.

SetTemperature

Mit dieser Funktion kann die Temperatur für einen Ofen eingestellt werden.

Syntax:
obj.**SetTemperature** (Oven, Value)

Parameter:

int Oven	1 .. 6
double Value	Neuer Temperaturwert (25 .. 400 °C)

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl benötigt ein Package-Level-Testsystem.

Size

Diese Funktion gibt die Anzahl der Messwerte in einem Messwerte-Array zurück.

Syntax:
obj.**Size** (Ref)

Parameter:

long Ref	Referenznummer auf eion Messwerte-Array

Rückgabe:

int	Anzahl der Messergebnisse (>= 0) oder Fehlercode

Anmerkung:
Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden.

Bei Verwendung der Rampenbefehle wird ebenfalls ein Messwerte-Array zurückgegeben.

In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Source

Dieser Befehl wählt eine Source-Measure-Unit für die nachfolgenden Befehle aus.

Syntax:
obj.**Source** (Number)

Parameter:
int Number Source-Measure-Unit (1..64)

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Source-Measure-Units, die mit diesem Befehl angesprochen werden, sind automatisch exklusiv für dieses Testprogramm reserviert und können von anderen Programmen nicht verwendet werden. Konsequenterweise gelten alle nachfolgenden Befehle wie **Reset** und **Start** auch nur für die eigenen, reservierten SMUs.

Die Source-Measure-Unit bleibt ausgewählt bis ein neuer **Source**-Befehl oder **Reset** ausgeführt wird. Der **Reset**-Befehl gibt alle reservierten Source-Measure-Units wieder für andere Testprogramme frei.

Mit dem Auswählen der SMU wird der normale Messausgang eingeschaltet (Output = 1).

Für Parameter-Analyzer sind die Nummer 1 to 64 gültig (sofern im Testsystem vorhanden).

Für Reliability-Tester werden für alle Boards die logischen Nummern 1..4 verwendet, d.h. die Source-Measure-Unit für Drain ist immer #1.

Für Package-Level-Testsysteme sind die Nummern 1, 2, 3, 4, 5, 7, 8, 9, 10 und 11 den Source-Measure-Units am Mess-Bus zugeordnet, die Nummern 13, 14, 15, 16, 17, 19, 20 und 21 den Source-Measure-Units am Stress-Bus. Diese Nummern werden auch beim **Relay**-Befehl als **Input**-Parameter verwendet (siehe dort).

Start

Dieser Befehl startet das Testprogramm.

Syntax:
obj.**Start** ()

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl schickt sämtliche gespeicherten Befehle an das Testsystem und startet das Programm. Es wird nicht auf das Ende des Testprogramms gewartet. Das Ende des Testprogramms kann mit der **Busy**-Funktion abgefragt werden.

Ein laufenden Testprogramm kann mit dem **Pause**-Befehl unterbrochen werden. Mit einem erneuten **Start**-Befehl kann das Programm wieder fortgesetzt oder mit einem **Clear**-Befehl endgültig abgebrochen werden.

Ein **Start**-Befehl schließt alle offenen Schleifen, die durch **SweepVoltage** oder **SweepCurrent** begonnen wurden. Ein **SweepEnd**-Befehl ist in diesem Fall nicht erforderlich.

Der **Execute**-Befehl startet ebenfalls das Testprogramm und wartet, bis das Programm beendet wurde.

Status

Diese Funktion gibt die Betriebsart der Source-Measure-Unit zum Zeitpunkt einer Messung zurück.

Syntax:
obj.**Status** (Ref, Index)

Parameter:

long Ref	Referenznummer
int Index	Index auf einen einzelnen Messwert bei Arrays (optional)

Rückgabe:

int	Betriebsart (siehe Tabelle) oder Fehlercode

Als Spannungsquelle:		**Als Stromquelle:**	
1	Spannungsregelung	2	Stromregelung
3	(+) Strombegrenzung	5	(+) Spannungsbegrenzung
4	(-) Strombegrenzung	6	(-) Spannungsbegrenzung

Anmerkung:
Die Betriebsart der Source-Measure-Unit wird mit jedem Messwert gespeichert. Das Messergebnis kann mit der **Result**-Function abgefragt werden.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer **SweepVoltage** oder **SweepCurrent**-Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In der Funktion **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben.

Beispiel:

```
obj.Source 1
obj.ForceVoltage 5                      ' smu 1 forces 5 volt
ref = obj.MeasureVoltage                ' invoke voltage measurement

obj.Execute                                     ' execute test program

volt = obj.Result(ref)                 ' get measurement
stat = obj.Status(ref)                 ' and compliance status

if (stat = 3 Or stat = 4) then
  ...                                   ' current compliance
endif
```

StatusArray

Diese Funktion gibt die Betriebsart der Source-Measure-Unit für alle Messungen in einem Messwerte-Array zurück.

Syntax:
obj.**StatusArray** (Ref, Array)

Parameter:

long Ref	Referenznummer
int Array()	Array (wird von der Funktion neu dimensioniert falls erforderlich)

Rückgabe:

int Array()	Statusinformationen für alle Messwerte (siehe Tabelle)
int	0 = Ok oder Fehlercode

Als Spannungsquelle:

1	Spannungsregelung
3	(+) Strombegrenzung
4	(-) Strombegrenzung

Als Stromquelle:

2	Stromregelung
5	(+) Spannungsbegrenzung
6	(-) Spannungsbegrenzung

Anmerkung:
Die Betriebsart der Source-Measure-Unit wird mit jedem Messwert gespeichert.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer **SweepVoltage** oder **SweepCurrent**-Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben.

Die Funktion **StatusArray** liefert die Statusinformationen für das gesamte Messwertarray, die Funktion **ResultArray** die dazugehörigen Messwerte.

Alternativ kann für einzelne Messwerte in einem Messwertarray auch die Funktion **Status** verwendet werden. In diesem Fall ist als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben.

Wenn das Messprogramm zum Zeitpunkt der Abfrage noch läuft, ist es möglich dass einzelne Messwerte noch nicht verfügbar sind. Die Verfügbarkeit einzelner Messwerte bzw. der dazugehörigen Statusinformationen oder des gesamten Arrays kann mit der Funktion **Available** geprüft werden.

SweepCurrent

Mit dieser Funktion definiert eine Anzahl von Stromschritten.

Syntax:
obj.**SweepCurrent** (Start, Step, Step, Count)

Parameter:

double Start	Startwert
double Stop	Endwert
double Step	Schrittweite (wird ignoriert, wenn *Count* angegeben wird)
int Count	Anzahl der Schritte (optional)

Rückgabe:

int	0=Ok oder Fehlercode

Anmerkung:
Dieser Befehl stellt die angegebenen Stromschritte ein. Gleichzeitig wird eine Schleife definiert, die sämtliche Befehle innerhalb dieser Schleife für jeden Stromwert wiederholt. Das Ende der Schleife wird durch einen **SweepEnd**, **Execute** oder **Start**-Befehl markiert. Mit mehreren **SweepVoltage** oder **SweepCurrent**-Befehlen sind auch geschachtelte Schleifen möglich.

Es kann entweder die Schrittweite (*Step*) oder die Anzahl der Schritte (*Count*) angegeben werden. Die Anzahl der eingestellten Stromwerte ist immer ums eins höher als *Count*.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von mit der Funktion **Size** zurückgegeben. In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Beispiele:

```
' die beiden nachfolgenden Befehle sind gleichwertig

obj.SweepCurrent 0, 0.02, 0.001            ' 0 to 20mA in 1mA steps
obj.SweepCurrent 0, 0.02,, 20
```

SweepEnd

Dieser Funktion beendet eine Schleife, die mit **SweepVoltage** oder **SweepCurrent** begonnen wurde.

Syntax:
obj.**SweepEnd**

Rückgabe:
int 0 = Ok oder Fehlercode

Anmerkung:
Mit dem Befehl **SweepVoltage** oder **SweepCurrent** wird eine Schleife definiert und sämtliche Messbefehle innerhalb dieser Schleife für alle Spannungs- bzw. Stromwerte wiederholt. Der **SweepEnd**-Befehl beendet diese Schleife, im Falle von verschachtelten Schleifen wird nur die innerste Schleife beendet.

Mit einem **Execute**- oder **Start**-Befehl werden sämtliche offenen Schleifen automatisch abgeschlossen. Der **SweepEnd**-Befehl kann dann auch weggelassen werden.

SweepVoltage

Mit dieser Funktion definiert eine Anzahl von Spannungsschritten.

Syntax:
obj.**SweepVoltage** (Start, Step, Step, Count)

Parameter:

double Start	Startwert
double Stop	Endwert
double Step	Schrittweite (wird ignoriert, wenn *Count* angegeben wird)
int Count	Anzahl der Schritte (optional)

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Dieser Befehl stellt die angegebenen Spannungsschritte ein. Gleichzeitig wird eine Schleife definiert, die sämtliche Befehle innerhalb dieser Schleife für jeden Spannungswert wiederholt. Das Ende der Schleife wird durch einen **SweepEnd**, **Execute** oder **Start**-Befehl markiert. Mit mehreren **SweepVoltage** oder **SweepCurrent**-Befehlen sind auch geschachtelte Schleifen möglich.

Es kann entweder die Schrittweite (*Step*) oder die Anzahl der Schritte (*Count*) angegeben werden. Die Anzahl der eingestellten Spannungswerte ist immer ums eins höher als *Count*.

Wenn die Befehle **MeasureVoltage**, **MeasureCurrent** or **MeasureTime** im Inneren einer Schleife verwendet werden, zeigt die zurückgegebene Referenznummer nicht auf einen einzelnen Messwert sondern auf ein Array mit allen Messwerten, die in dieser Schleife ermittelt werden. Die Anzahl der Messwerte wird von der Funktion **Size** zurückgegeben. In den Funktionen **Available**, **Result** und **Status** ist in diesem Fall als zweiter Parameter der Index des Messwertes (1 .. **Size**) anzugeben. Alternativ stehen die Funktionen **ResultArray** und **StatusArray** zur Verfügung, mit denen alle Daten des Arrays auf einmal zurückgegeben werden.

Beispiele:

```
' die beiden nachfolgenden Befehle sind gleichwertig

obj.SweepVoltage 0, 5, 0.1                    ' 0 to 5V in 100mV steps
obj.SweepVoltage 0, 5,, 50
```

Temperature

Mit dieser Funktion kann die aktuelle Temperatur eines Ofens ausgelesen werden.

Syntax:
obj.**Temperature** (Oven)

Parameter:

int Oven	1 .. 6

Rückgabe:

double	Aktuelle Temperatur in °C
	-999,9 ... falls kein Oven installiert ist

Anmerkung:
Dieser Befehl benötigt ein Package-Level-Testsystem.

VoltageLimit

Dieser Befehl stellt die Spannungsbegrenzung für eine Source-Measure-Unit ein, die als Stromquelle verwendet wird.

Syntax:
obj.**VoltageLimit** (Limit)

Parameter:

double Limit	Spannungsbegrenzung

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Die eingestellte Spannungsbegrenzung wird erst aktiv, wenn die Source-Measure-Unit als Stromquelle verwendet wird.

Die Spannungsbegrenzung gilt für beide Polaritäten, d.h. wenn z.B. 10 V eingestellt wird, wird die Ausgangsspannung auf –10 V bis +10 V begrenzt. Die Spannungsbegrenzung hat in jedem Fall Vorrang gegenüber dem eingestellten Strom, je nach Anwendung kann unter Umständen auch ein höherer Ausgangsstrom als der mit **ForceCurrent** eingestellte fließen.

Die maximale Ausgangsspannung hängt vom ausgewählten Spannungsbereich (falls nicht Auto-Range) und vom Source-Measure-Unit-Modell ab.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

VoltageRange

Diese Befehl wählt einen von 4 Spannungsbereichen oder Auto-Ranging.

Syntax:
obj.**VoltageRange** (Range)

Parameter:

int Range	0 = Auto
	1 = 200 mV
	2 = 2 V
	3 = 20 V
	4 = 200 V

Rückgabe:

int	0 = Ok oder Fehlercode

Anmerkung:
Mit Auto-Ranging wird automatisch der beste Spannungsbereich für jeden Messpunkt ausgewählt, allerdings kann das Vorgeben eines Bereiches eine deutliche Verbesserung der Messgeschwindigkeit bewirken. Im Auto-Range Modus kann mit Hilfe des **LowestVoltageRange**-Befehls eine Einschränkung der zulässigen Spannungsbereiche erfolgen.

Dieser Befehl wird gespeichert und wird erst mit einem **Start** oder **Execute** Befehl von der entsprechenden Source-Measure-Unit ausgeführt.

VoltageRangeMaximum

Diese Funktion gibt die maximal mögliche Ausgangsspannung der SMU zurück.

Syntax:
obj.**VoltageRangeMaximum** (Range)

Parameter:

int Range	Spannungsbereich (0 oder 1 .. obj.VoltageRanges**)**
	wenn 0 angegeben wird, wird die maximale Spannung des letzten Spannungsbereiches zurückgegeben

Rückgabe:

double	Maximal mögliche Ausgangsspannung

Anmerkung:
Mit der **VoltageRanges**-Funktion kann die Anzahl der Spannungsbereiche abgefragt werden.

Abhängig vom SMU-Modell kann die maximal mögliche Ausgangsspannung bei höheren Ausgangsströmen kleiner sein als diese Funktion zurückgibt. Für Details siehe die Hardwaredokumentation.

Dieser Befehl wird sofort ausgeführt.

VoltageRanges

Diese Funktion gibt die Anzahl der Spannungsbereiche der SMU zurück.

Syntax:
obj.**VoltageRanges**

Rückgabe:
int Anzahl der Spannungsbereiche

Anmerkung:
Mit der **VoltageRangeMaximum**-Funktion kann für jeden Bereich die maximale Ausgangsspannung abgefragt werden.

Dieser Befehl wird sofort ausgeführt.

VoltageToRange

Diese Befehl erwartet einen Spannungswert und gibt den kleinsten passenden Spannungsbereich zurück.

Syntax:
obj.**VoltageToRange** (Value)

Parameter:

double Value	Spannung

Rückgabe:

int	1 = 200 mV 2 = 2 V 3 = 20 V 4 = 200 V ERROR_VOLTAGE (unzulässige Spannung)

Anmerkung:
Dieser Befehl wird sofort ausgeführt.

Fehlercodes

Alle Methoden und Funktionen mit Ausnahme der **Result**-Funktion geben im Fehlerfall einen der folgenden Codes zurück. In einigen Fällen wird zusätzlich eine Eintrag im Fehlerlog (*c:\Programme\mbTester\Log*) vorgenommen, der den Fehler näher erläutert.

0	OK	Kein Fehler
-1	ERROR_PARAMETER	Ungültiger Parameter
-2	ERROR_CONFIGURATION	Ungülige Konfiguration
-3	ERROR_NOTFOUND	Hardware nicht gefunden
-4	ERROR_FAILED	Allgemeiner Fehler (siehe Errorlog)
-5	ERROR_TIMEOUT	Zeitüberschreitung
-6	ERROR_SOURCE	Source-Measure-Unit nicht angegeben
-7	ERROR_NOBOARD	Board nicht angegeben
-8	ERROR_REFERENCE	Ungültige Referenznummer
-9	ERROR_RAMPMODE	Ungültiger Rampen-Modus
-10	ERROR_RAMPSTEPS	Zu viele Rampenschritte
-11	ERROR_VOLTAGE	Unzulässige Spannung
-12	ERROR_CURRENT	Unzulässiger Strom
-13	ERROR_INDEX	Ungültiger Index in einem Messwerte-Array
-14	ERROR_OVEN	Ungültige Ofennummer
-15	ERROR_PACKAGE	Ungültige Package-Nummer
-16	ERROR_TEMPERATURE	Ungültige Temperatur
-17	ERROR_RELAY	Ungültige Relais-Nummer