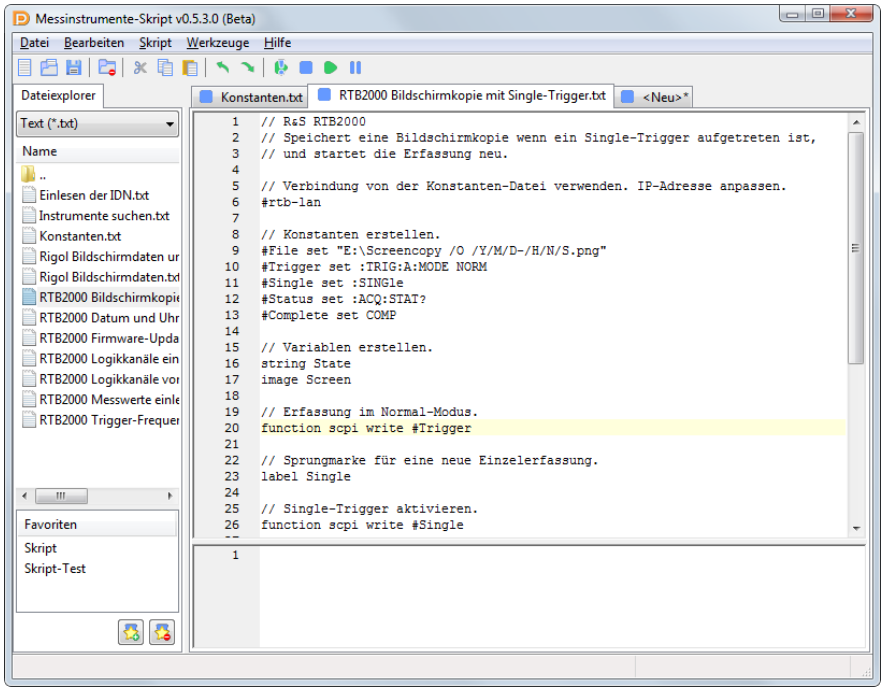


Messinstrumente-Skript v0.5.4.1

Inhalt

1 Einführung

Das Programm Messinstrumente-Skript kann mit Messinstrumenten kommunizieren. Durch einfache Anweisungen in Textdateien können Kommandos an Messinstrumente gesendet, und Daten empfangen werden. Diese Textdateien mit den Anweisungen werden Skripte genannt. Es können mehrere Skripte geöffnet, und unabhängig voneinander ausgeführt werden. Instrumente können beliebig in den Skripten verwendet werden, ein gleichzeitiges Verwenden in mehreren Skripten ist möglich.

Es können Skripte erstellt, geöffnet, bearbeitet und gespeichert werden. Die Skripte werden vor der Ausführung auf Fehler geprüft. Die Skriptanweisungen sind einfach gehalten, aber bestimmte Anweisungen führen komplexe Operationen aus, dadurch wird die Steuerung eines Messinstruments vereinfacht. Die eingelesenen Daten von den Messinstrumenten können gespeichert, und in eigenen Ansichtfenstern dargestellt werden.

Die Kommunikation mit den Instrumenten erfolgt über SCPI-Kommandos. Es können COM-, USB-TMC-, LXI- und RawTCP-Verbindungen verwendet werden.

Das Programm benötigt keine Installation und keine installierte Software oder Treiber von den Herstellern der Messinstrumente. Nur für die USB-TMC-Verbindung wird der WinUSB-Treiber von Microsoft benötigt. Ausgeführt kann das Programm ab Window XP werden, für Windows 2000 gibt es eine eigene Version.

Die Entwicklung bzw. Programmierung ist noch nicht abgeschlossen, und viele Funktionen fehlen noch. Anweisungen und Beschreibungen die in dieser Hilfe dokumentiert sind, sind auch im Programm vorhanden, ansonst gibt es einen Hinweis auf das Fehlen der Funktion.

Programmiert wird das Programm mit PureBasic von Fantaisie Software. Es wird immer die letzte Version, oder auch die aktuelle Beta-Version von PureBasic verwendet.

Getestet wurde das Programm unter Windows 2000, XP und Window 7, in den 32 Bit Versionen. Als Messinstrumente wurden verwendet: Rigol DG4000, DP831, DSA815 und R&S RTB2004.

Wer möchte kann gerne Kritik, Fehlerbeschreibungen oder Anregungen für verbesserte oder neue Funktionen per E-Mail senden.

2 Voraussetzungen

Das Programm kann unter Windows XP mit Service Pack 3, oder höher ausgeführt werden. Für Windows 2000 mit Service Pack 4, gibt es eine eigene Version. Das Programm funktioniert auch mit den 64 Bit Versionen von Windows.

Es wird keine installierte Software oder Treiber von den Herstellern der verwendeten Messinstrumente benötigt. Für die USB-TMC-Verbindung mit einem Messinstrument muss der Microsoft WinUSB-Treiber installiert werden.

Als CPU wird mindestens ein Pentium III benötigt, da SSE-Instruktionen verwendet werden.

3 Installation

Das Programm benötigt keine Installation. Damit Einstellungen gespeichert werden können, darf das Verzeichnis von dem das Programm gestartet wird, nicht schreibgeschützt sein. Es wird nur in das Verzeichnis 'Ressource' automatisch geschrieben. Eine Internetverbindung wird nicht benötigt, es wird keine Verbindung hergestellt.

Für die Kommunikation über die COM-, RawTCP- und LXI-Verbindung wird kein Treiber oder Software von den Herstellern der Messinstrumente benötigt.

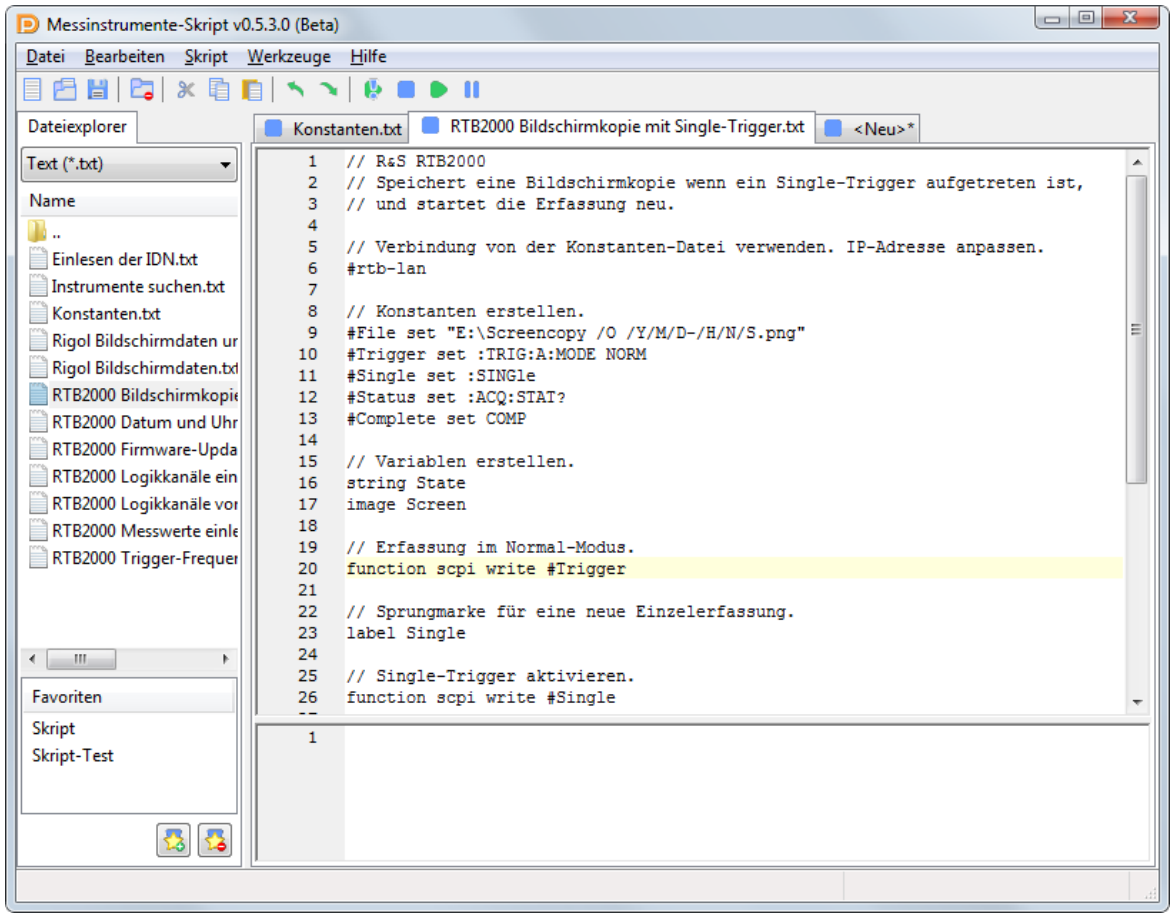
Messinstrumente mit USB-TMC-Verbindung benötigen den Microsoft WinUSB-Treiber. Eine '.inf'-Installationsdatei für das jeweilige Instrument ist ausreichend. Es wird nur der WinUSB-Treiber installiert, der schon in Windows vorhanden ist. Eine einfache Möglichkeit den Treiber zu installieren, bietet das Programm Zadig. Das Programm ist Open Source und wird für die Installation empfohlen.

Die Webseite ist https://zadig.akeo.ie/

Für Windows 2000 und XP kann die Version 2.2 verwendet werden. Unter Windows 2000 muss der Treiber libusb-win32 ausgewählt werden. Der jeweilige Treiber muss für jedes Instrument installiert werden. Die Installation kann manchmal etwas länger dauern, und das Zadig-Programm gibt keine Rückmeldung. Einfach etwas warten bis die Installation abgeschlossen ist.

4 Skript-Editor

Der Skript-Editor wird beim Start des Programms angezeigt. Mit dem Editor können Skripte geladen, bearbeitet, gespeichert und ausgeführt werden. Der Editor besteht aus drei Bereichen:



Skript- Bearbeitungsbereich
Hier werden die Skripte angezeigt und können bearbeitet werden. Mit den Reitern direkt oberhalb kann zwischen den Skripten gewechselt werden. Der Dateiname des gespeicherten Skripts wird im Reiter angezeigt. Wurde das Skript verändert, wird an den Namen ein Sternzeichen '*' angefügt, das wieder verschwindet wenn das Skript gespeichert wird.
Die Reiter zeigen einen Tooltip mit dem kompletten Pfad des Skripts an, wenn der Mauszeiger kurz auf einem Reiter verweilt. Mit einem Rechtsklick auf den Reiter wird der Editor ausgewählt und ein Kontextmenü angezeigt.

Ausgabebereich
Unterhalb des Bearbeitungsbereich für ein Skript befindet sich die Ausgabe des jeweiligen Skripts. Jedes Skript hat seine eigene Ausgabe in dem Meldungen und Fehler ausgegeben werden. Wird ein Skript z.B. angehalten und wartet auf eine Bestätigung, wird diese Meldung hier angezeigt. Die Ausgaben werden automatisch in den Sichtbereich gescrollt.

Werkzeugbereich
Links ist der Bereich für Werkzeuge die helfen das Verwalten und Bearbeiten von Skripten zu vereinfachen.

Es gibt dann noch ganz oben die Menüzeile und darunter eine Werkzeugleiste. Die Befehle können so auf verschiedene Weise ausgeführt werden. Vorhandene Tastaturkürzel werden in den geöffneten Menüs angezeigt.

Am unteren Rand des Skript-Editors befindet sich die Statusleiste. Hier wird für das jeweilige ausgewählte Skript Information während der Ausführung angezeigt.

Das Skript-Fenster speichert beim Beenden die Fensterposition und die Aufteilung der Bereiche. Die Bereiche können mit den Stegen zwischen den Bereichen mit der Maus verändert werden. Mit dem Menü oder Tastaturkürzel können sie aus- und wieder eingeblendet werden.

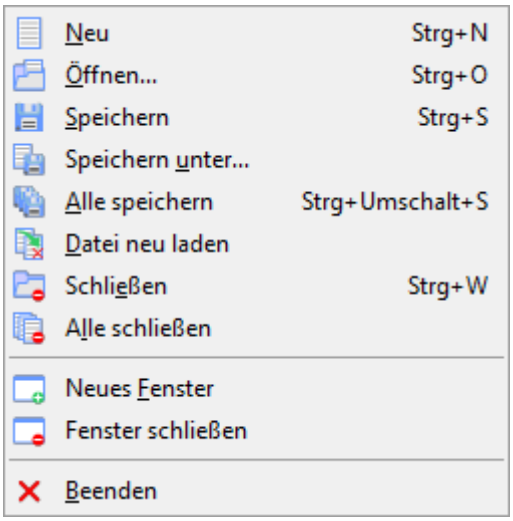
4.1 Arbeiten mit Dateien

Das Dateimenü ermöglicht das Öffnen und Speichern von Dateien bzw. Skripten.

Es können mehrere Skripte geöffnet und bearbeitet werden. Zwischen den Skripten kann mit Klick auf den gewünschten Reiter gewechselt werden. Mit den Tastaturkürzel *Strg+Tab* und *Strg+Umschalt+Tab* kann zum nächsten oder verherigen Skript umgeschalten werden.
Die Dateinamen der geöffneten Skripte werden auch im Menü Skript und im Kontextmenü der Reiter angezeigt.

Skripte sind Textdateien mit der Erweiterung '.txt', und werden im UTF8-Format gespeichert. Die Skripte können auch mit anderen Texteditoren bearbeitet werden. Unter Windows 2000 kann es sein, dass ein Skript in einem Texteditor nicht angezeigt wird, dann unterstützt dieser wahrscheinlich das UTF8-Format nicht.

Inhalt des Dateimenü:



Neu
Erstellt ein neues leeres Skript.

Öffnen
Zeigt einen Dateidialog für die Auswahl eines oder mehrerer Skripte an. Wenn ein neues leeres Skript vorhanden ist, wird dieses automatisch geschlossen. Skripte können auch mit Drag&Drop auf den Editorbereich geöffnet werden.

Speichern
Speichert das aktuell angezeigte Skript. Wurde das Skript noch nicht gespeichert, wird ein Dateidialog für den Dateinamen angezeigt.

Speichern unter
Speichert das aktuell angezeigte Skript unter einem neuen Namen. Es wird ein Dateidialog für den Dateinamen angezeigt.

Alle Speichern
Es werden alle geänderten oder neu erstellten Skripte gespeichert. Wurde ein Skript noch nicht gespeichert, wird ein Dateidialog für den Dateinamen angezeigt.

Datei neu laden
Das gespeicherte Skript wird neu geladen. Wurde das Skript verändert, wird ein Dialog angezeigt und der Vorgang kann abgebrochen werden.

Schließen
Schließt das aktuell angezeigte Skript. Wurde das Skript verändert, wird ein Dialog angezeigt und das Skript kann gespeichert oder der Vorgang abgebrochen werden.

Alle schließen
Schließt alle geöffneten Skripte. Wurde ein Skript verändert, wird ein Dialog angezeigt und das Skript kann gespeichert oder der Vorgang abgebrochen werden. Es wird automatisch ein neues leeres Skript erstellt.

Neues Fenster
Öffnet einen neuen Skript-Editor.
Wichtig: Die geöffneten Skripte und Einstellungen, wie z.B. Favoriten, Größe der Bereiche und Fensterposition, werden nicht zwischen den Skript-Editor-Fenstern synchronisiert. Es werden die Einstellungen des zuletzt geschlossenen Fensters gespeichert.

Fenster schließen
Schließt das aktuelle Skript-Editor-Fenster. Wurden Skripte noch nicht gespeichert, wird ein Dialog angezeigt und die Skripte können gespeichert oder der Vorgang abgebrochen werden.

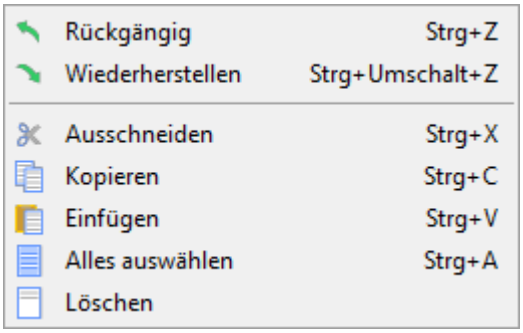
Beenden
Schließt alle Skript-Editor-Fenster und beendet das Programm. Wurden Skripte noch nicht gespeichert, wird ein Dialog angezeigt und die Skripte können gespeichert oder der Vorgang abgebrochen werden.

4.2 Bearbeitungsfunktionen

Der Editor für die Skripte verhält sich wie ein einfacher Texteditor. Die Funktionen für Auswählen, Ausschneiden, Kopieren, Einfügen und Löschen, sowie Rückgängig und Wiederherstellen sind vorhanden.

Der Editor passiert auf einem Scintilla-Steuerelement, welches viele Funktionen bietet, diese aber zurzeit nicht verwendet werden. Die Größe der Schrift kann mit dem Mausrad und Strg-Taste verändert werden.

Inhalt des Bearbeiten- und Kontextmenü:



Rückgängig
Stellt die letzte Veränderung wieder her. Es gibt einen Buffer für die Änderungen, und es können mehrere Änderungen wieder rückgängig gemacht werden.

Wiederherstellen
Stellt den Zustand vor der letzten Verwendung von 'Rückgängig' wieder her.

Ausschneiden
Schneidet den ausgewählten Text aus, und kopiert ihn in die Zwischenablage.

Kopieren
Kopiert den ausgewählten Text in die Zwischenablage. Ist kein Text ausgewählt, wird die gesamte Zeile in der sich der Textcursor befindet, in die Zwischenablage kopiert.

Einfügen
Fügt den Text von der Zwischenablage an die aktuelle Position des Textcursors ein.

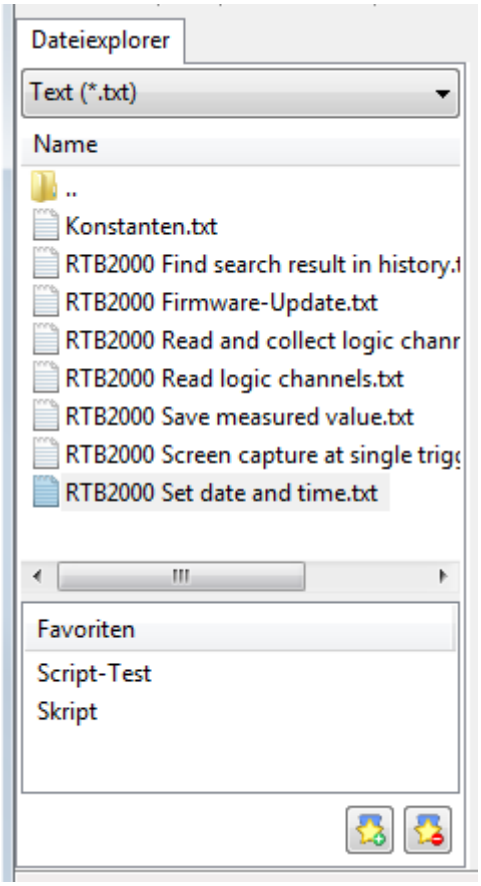
Alles auswählen
Wählt den gesamten Text im Editor aus.

Löschen
Löscht den ausgewählten Text.

4.3 Werkzeuge

Es gibt mehrere Werkzeuge die das Arbeiten mit Dateien und Skripten erleichtern.

Dateiexplorer



Der Dateiexplorer zeigt die Skriptdateien von einem ausgewählten Verzeichnis mit der Erweiterung '.txt' an. Skripte können per Doppelklick oder Drag&Drop auf den Editor, geöffnet werden. Mit Drag&Drop können auch mehrere Skripte ausgewählt und geöffnet werden.

Mit der Filterliste am oberen Rand, können nur Skripte oder alle Dateien in einem Verzeichnis angezeigt werden.

Unter der Dateiliste befindet sich die Favortienliste. Mit den Schaltflächen darunter können Verzeichnis in die Liste hinzugefügt oder entfernt werden.
Es wird entweder ein ausgewähltes Verzeichnis, oder das aktuell angezeigte Verzeichnis mit der linken Schaltfläche der Liste hinzugefügt. Mit einem Doppelklick auf einen Favoriten, wird in der Dateiliste zu diesem Verzeichnis gewechselt. Mit der rechten Schaltfläche kann man einen ausgewählten Favoriten aus der Liste entfernen.

Die Favoriten werden beim Beenden des Programms in der Einstellungsdatei 'Einstellungen.ini' im Verzeichnis 'Ressourcen' gespeichert.

4.4 Skripte verwenden

Ein Skript kann auf Syntaxfehler geprüft und danach ausgeführt werden. Ein laufendes Skript kann angehalten oder abgebrochen werden. Ein laufendes Skript gibt Meldungen oder Fehler im zugehörigen Ausgabebereich aus. Der aktuelle Zustand der einzelnen Anweisungen wird in der Statusleiste angezeigt.

Das geprüfte Skript und Fehlermeldungen werden im Log-Fenster gespeichert. Unter dem Menü 'Hilfe' und dem Eintrag 'Log...', kann das Log-Fenster geöffnet werden.

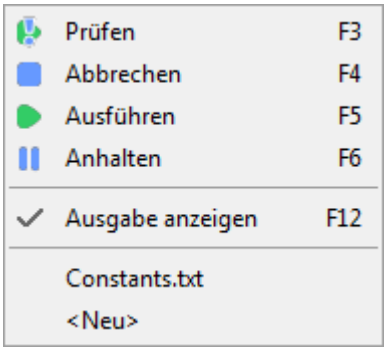
Bei einem Fehler wird das Skript angehalten, und im Reiter des Editors erscheint ein rotes Symbol für 'Anhalten'. Das Skript kann weiter ausgeführt oder abgebrochen werden.

Der Zustand der Skripte wird mit den entsprechenden Symbolen im jeweiligen Editor-Reiter angezeigt.

Skripte laufen unabhängig voneinander und es können mehrere Skripte gleichzeitig ausgeführt werden. Messintrumente können ebenfalls gleichzeitig in Skripten verwendet werden. Die gesendeten Kommandos der Skripte werden nacheinander ausgeführt. Die Skripte selbst laufen jeweils in einem eigenen Thread, dass verwendete Instrument wird aber von einem eigenen Thread gesteuert, der allen Skripten zur Verfügung steht.

Gesteuert werden können die Skripte über das Menü, die Werkzeugleiste oder die Tastaturkürzel.

Inhalt des Menü Skript:



Prüfen
Das aktuell angezeigte Skript wird auf Syntaxfehler geprüft. Fehler werden im Ausgabebereich angezeigt.

Abbrechen
Ein laufendes Skript wird abgebrochen. Es kann sein das nicht sofort abgebrochen wird wenn z.B. von einem Instrument Daten eingelesen, oder Daten gespeichert werden.

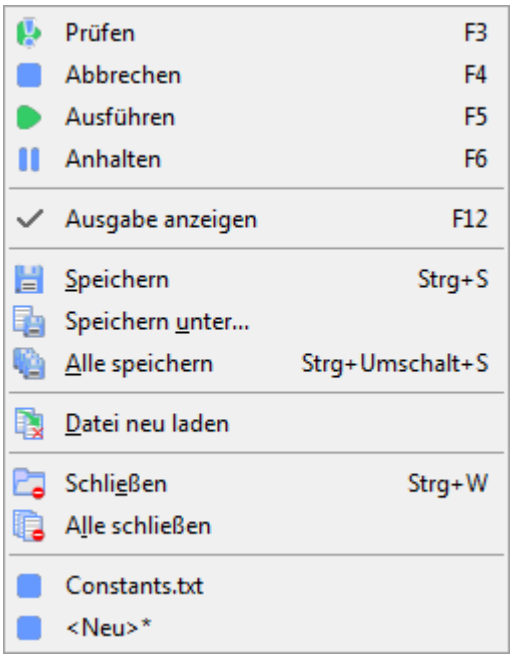
Ausführen
Ein Skript wird vor der Ausführung immer auf Syntaxfehler geprüft, und erst danach ausgeführt. Ein angehaltenes Skript kann weiter ausgeführt werden. Wenn ein Skript eine Bestätigung erwartet, muss es ebenfalls mit 'Ausführen' wieder gestartet werden.

Anhalten
Ein laufendes Skript wird angehalten, und kann abgebrochen oder weiter ausgeführt werden.

Ausgabe anzeigen
Blendet den Ausgabebereich für das jeweilige Skript aus oder wieder ein.

Liste der geöffneten Skripte
Die geöffneten Skripte werden am Ende des Menüs angezeigt, und könne so auch ausgewählt werden. Der Zustand der Skripte wird in diesem Menü nicht angezeigt, auch ob das Skript verändert wurde wird nicht angezeigt.

Inhalt des Kontextmenü der Editor-Reiter:



Die Liste der geöffneten Skripte und deren Zustand wird am Ende des Kontextmenü angezeigt.

4.5 Konstanten

Der Skript-Editor lädt beim Start automatisch die Datei 'Konstanten.txt' aus dem Verzeichnis 'Skript'. Wenn diese Datei geöffnet ist, können die Konstanten die in dieser Datei angegeben wurden, von den Skripten verwendet werden. Es wird nach dieser geöffneten Datei 'Konstanten.txt' beim Überprüfen oder Ausführen eines Skripts gesucht, und die Konstanten in das Skript eingebunden.
Wird diese Datei geschlossen, werden keine Konstanten daraus verwendet.

Mit dieser Datei können z.B. Verbindungseinstellungen für Messinstrumente angelegt, und in allen Skripten verwendet werden. Bei einer Änderung müssen dann nicht alle Skripte angepasst werden, sondern nur diese Konstanten-Datei.

Wenn man die Konstanten-Datei nicht automatisch laden möchte, kann man sie umbenennen oder löschen.

4.6 Kommandozeileoptionen

Das Programm kann mit Optionen in der Kommandozeile oder einer Verknüpfung gestartet werden. Die Optionen müssen mit einem Schrägstrich beginnen, die Groß- und Kleinschreibung wird ignoriert. Skripte müssen mit vollständigen Pfad in Anführungszeichen angegeben werden.

Optionen:

/hidden
Das Programm wird versteckt ausgeführt. Wenn ein Fehler bei der Ausführung des Skripts

auftritt, wird das Programm angezeigt.
Die Option wird nur mit der Option /run und /exit berücksichtigt.

/exit
Nach der Ausführung eines Skripts wird das Programm beendet. Wenn ein Fehler bei der Ausführung des Skripts auftritt, wird das Programm nicht beendet.
Die Option wird nur mit der Option /run berücksichtigt.

/load "T:\Skript.txt"
Lädt die angegebene Skriptdatei. Es muss der vollständige Pfad in Anführungszeichen angegeben werden.
Die Optionen /hidden und /exit werden ignoriert.

/run "T:\Skript.txt"
Lädt die angegebene Skriptdatei und führt sie sofort aus. Es muss der vollständige Pfad in Anführungszeichen angegeben werden.
Die Optionen /hidden und /exit werden berücksichtigt.

5 Ansichtsfenster

Mit der Skript-Anweisung **function view** können die Inhalte von Variablen in Ansichtsfenstern angezeigt werden. Je nach Datentyp der verwendeten Variable, wird ein eigenes Ansichtsfenster verwendet.

Ansichtsfenster:

Text	Wird für den Datentyp ***boolen***, i***nteger***, ***float*** und ***string*** verwendet.
Daten	Wird für den Datentyp ***data*** verwendet.
Bild	Wird für den Datentyp ***image*** verwendet.
Messkurve	Wird für den Datentyp ***waveform*** verwendet.

Hinweis: Die Ansicht des Datentyp ***waveform*** funktioniert noch nicht.

5.1 Bild

Das Ansichtsfenster für den Datentyp ***image*** zeigt die Bilder von den Variablen an. Wenn ein Name für das Fenster vergeben wurde, wird dieser in der Titelleiste angezeigt, ansonst der Dateiname des gespeicherten Skripts. Die Bilder können auf die Fenstergröße skaliert, oder in der originalen Größe angezeigt werden. Das Fenster kann sich auch an die Größe des Bildes anpassen.

Es werden die aktuellsten zehn Bilder im Ansichtsfenster gespeichert. Werden mehr als zehn Bilder angezeigt, werden die ältesten Bilder entfernt und die neuen Bilder eingefügt.

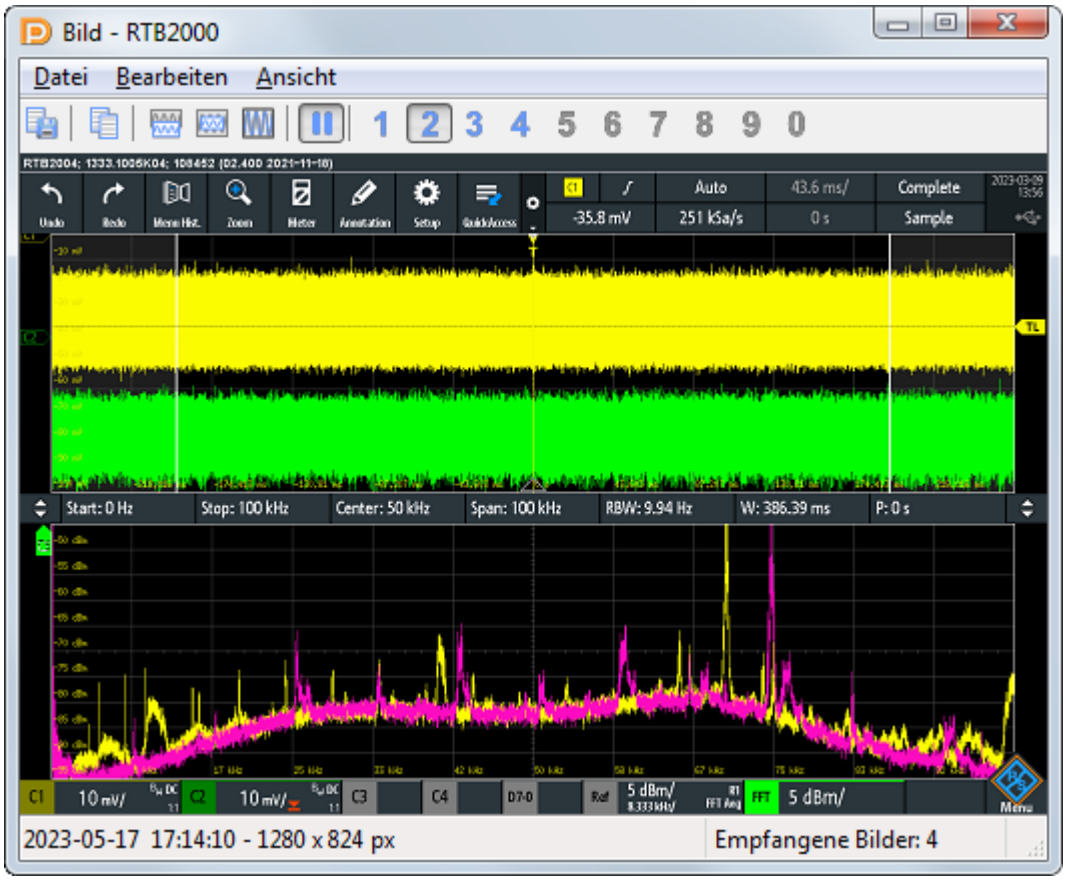
Es kann das angezeigte Bild jederzeit in die Zwischenablage kopiert werden. Wenn ein Bild gespeichert werden soll, muss die Anzeige der Bilder angehalten werden. Bilder die an das Ansichtfenster gesendet werden, gehen im angehaltenen Modus verloren.

Mit den Tasten 1 bis 9 und 0, kann das jeweilige Bild angezeigt werden. Das Ansichtsfenster wird

bei der Auswahl eines Bildes angehalten. Das aktuellste Bild ist unter der Nummer 1, und das älteste unter der Nummer 0 gespeichert.

Wird ein Bild für die Ansicht empfangen, wird immer der Schaltfläche 1 aktiviert, wo das neueste Bild gespeichert ist.

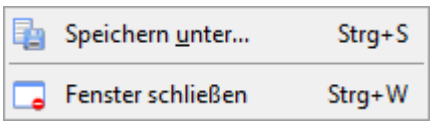
Bei der Skript-Anweisung **function view** ist beschrieben wie das Ansichtsfenster angezeigt und verwendet werden kann.



Menü und Werkzeugleiste

Das Menü und die Werkzeugleiste enthalten die gleichen Befehle. Aber nur in der Werkzeugleiste sind Schaltflächen für die Auswahl der gespeicherten Bilder vorhanden.

Inhalt des Dateimenü:



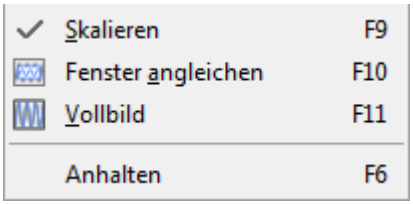
Speichern unter
Speichert das aktuell angezeigte Bild im PNG-Format. Es wird ein Dateidialog für den Dateinamen angezeigt.
Der Menüpunkt und die Schaltfläche in der Werkzeugleiste sind nur aktiviert, wenn das Ansichtsfenster gestoppt ist.

Fenster schließen
Schließt das Ansichtsfenster ohne weitere Nachfrage.

Inhalt des Bearbeitungsmenü:

Kopieren
Kopiert das aktuell angezeigte Bild in die Zwischenablage.

Inhalt des Ansichtmenü:



Skalieren
Mit aktivierten Menüpunkt *Skalieren*, wird das Bild an die Fenstergröße angepasst. Das Seitenverhältnis des Bilds bleibt erhalten. Wird der Menüpunkt deaktiviert, wird das Bild in der originalen Größe angezeigt.

Fenster angleichen
Gleicht das Ansichtsfenster an die aktuelle Größe des angezeigten Bildes an.

Vollbild
Das Ansichtsfenster kann mit diesen Menüpunkt im Vollbildmodus angezeigt werden. Beendet werden kann der Vollbildmodus mit der Funktionstaste *F11*, oder der Taste *Escape*.

Anhalten
Damit Bilder gespeichert werden können, muss das Ansichtsfenster angehalten werden. Der Menüpunkt wechselt zwischen dem Empfang und Anzeige der Bilder, und dem gestoppten Zustand.

Werkzeugleiste
Die Schaltflächen 1 bis 0 werden je nach Anzahl der vorhandenen Bilder aktiviert. Wird ein Bild über eine Schaltfläche oder der Tastatur angezeigt, wird das Ansichtsfenster gestoppt.

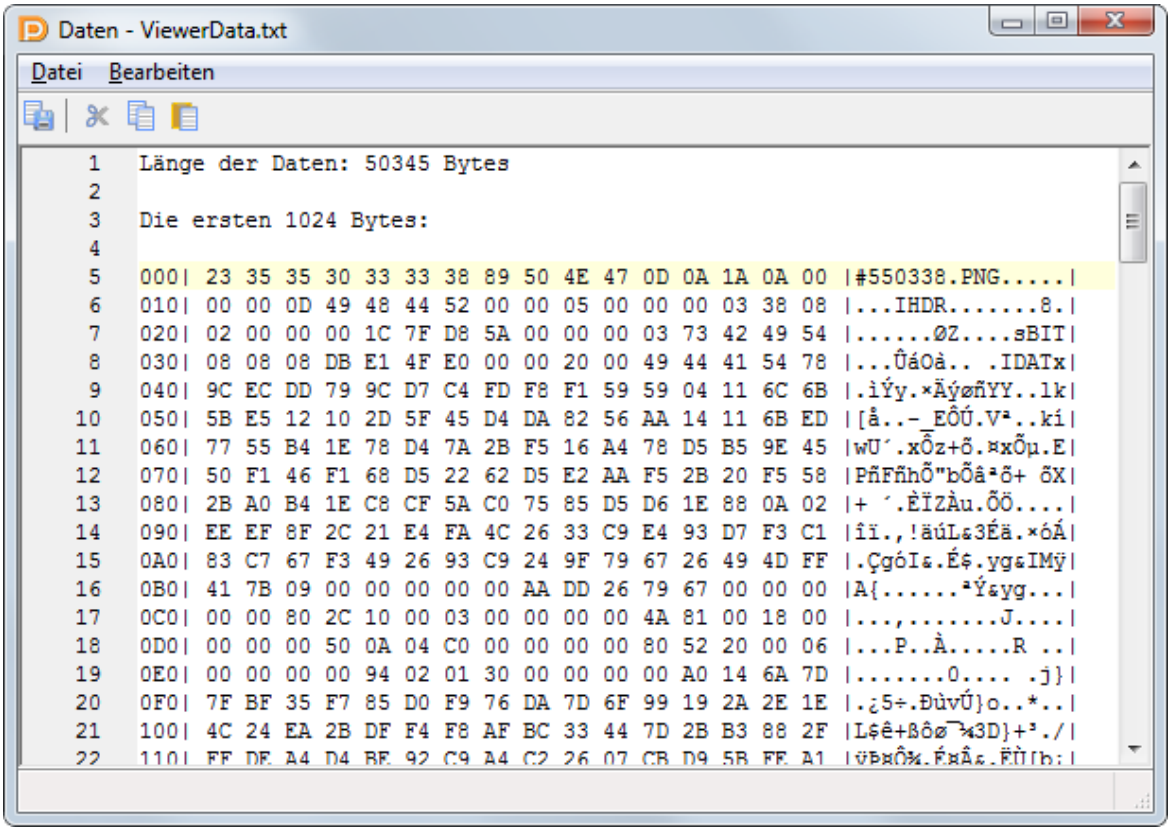
5.2 Daten

Das Ansichtsfenster für den Datentyp ***data*** zeigt den Inhalt der Variable an. Wenn ein Name für das Fenster vergeben wurde, wird dieser in der Titelleiste angezeigt, ansonst der Dateiname des gespeicherten Skripts.

Der Inhalt wird in Hexadezimalzahlen und ASCII-Zeichen dargestellt. Es werden die ersten und letzten 1.024 Bytes von den Daten angezeigt.

Der angezeigte Text kann in die Zwischenablage kopiert, oder in eine Datei gespeichert werden.

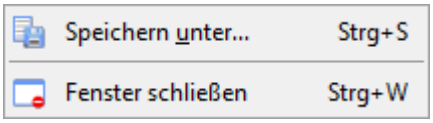
Bei der Skript-Anweisung **function view** ist beschrieben wie das Ansichtsfenster angezeigt und verwendet werden kann.



Menü und Werkzeugleiste

Das Menü und die Werkzeugleiste enthalten die gleichen Befehle.

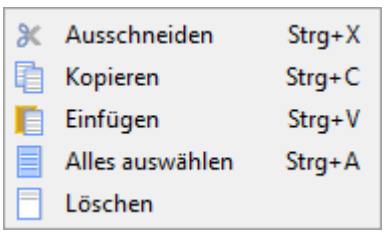
Inhalt des Dateimenü:



Speichern unter
Speichert den Text im UTF8-Format in eine Text-Datei. Es wird ein Dateidialog für den Dateinamen angezeigt.

Fenster schließen
Schließt das Ansichtsfenster ohne weitere Nachfrage.

Inhalt des Bearbeitungsmenü:



Ausschneiden
Schneidet den ausgewählten Text aus, und kopiert ihn in die Zwischenablage.

Kopieren
Kopiert den ausgewählten Text in die Zwischenablage.
Hinweis: Ist kein Text ausgewählt, wird der gesamte Text in die Zwischenablage kopiert.

Einfügen
Fügt den Text von der Zwischenablage an die aktuelle Position des Textcursors ein.

Alles auswählen
Wählt den gesamten Text im Editor aus.

Löschen
Löscht den ausgewählten Text.

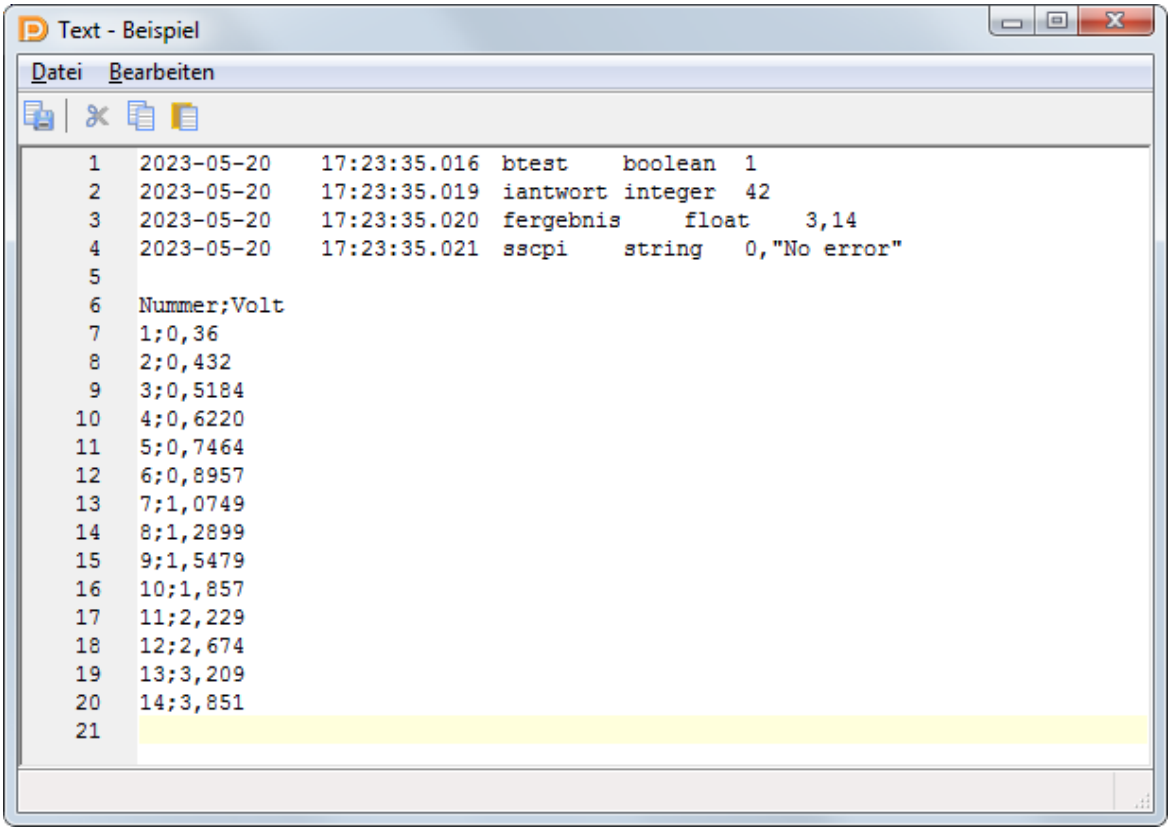
5.3 Text

Das Ansichtsfenster für den Datentyp ***boolean***, ***integer***, ***float*** und ***string*** zeigt die Inhalte von den Variablen an. Wenn ein Name für das Fenster vergeben wurde, wird dieser in der Titelleiste angezeigt, ansonst der Dateiname des gespeicherten Skripts.

Standardmäßig wird der Inhalt einer Variable zu vorhandenen Text im Ansichtsfenster mit einen Zeilenumbruch hinzugefügt. Mit der Angabe von Parametern kann die Ausgabe des Inhalts beeinflusst werden.

Der angezeigte Text kann in die Zwischenablage kopiert, oder in eine Datei gespeichert werden.

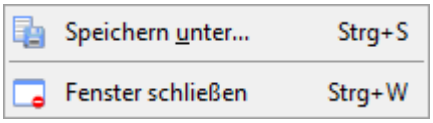
Bei der Skript-Anweisung **function view** ist beschrieben wie das Ansichtsfenster angezeigt und verwendet werden kann.



Menü und Werkzeugleiste

Das Menü und die Werkzeugleiste enthalten die gleichen Befehle.

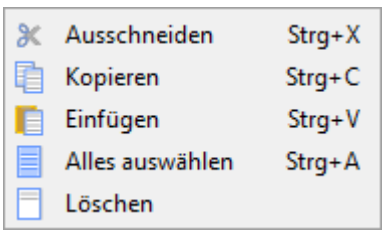
Inhalt des Dateimenü:



Speichern unter
Speichert den Text im UTF8-Format in eine Text-Datei. Es wird ein Dateidialog für den Dateinamen angezeigt.

Fenster schließen
Schließt das Ansichtsfenster ohne weitere Nachfrage.

Inhalt des Bearbeitungsmenü:



Ausschneiden
Schneidet den ausgewählten Text aus, und kopiert ihn in die Zwischenablage.

Kopieren
Kopiert den ausgewählten Text in die Zwischenablage.
Hinweis: Ist kein Text ausgewählt, wird der gesamte Text in die Zwischenablage kopiert.

Einfügen
Fügt den Text von der Zwischenablage an die aktuelle Position des Textcursors ein.

Alles auswählen
Wählt den gesamten Text im Editor aus.

Löschen
Löscht den ausgewählten Text.

6 Skript

Ein Skript führt Anweisungen aus, um ein oder mehrere Messinstrumente zu steuern, Daten einzulesen oder zu senden, und zu speichern. Die empfangenen Daten können in Ansichtsfenstern angezeigt werden, wie z.B. eine Bildschirmkopie oder Messkurve von einem Oszilloskop.

Die Anweisungen sind relativ einfach, und können nicht mit einer Programmiersprache verglichen werden. Einige Anweisungen führen aber komplexe Aufgaben aus, wie z.B. das Einlesen aller sichtbaren Messkurven von einem Oszilloskop.

Anweisungen werden als einzelne Wörter angegeben und mit eventuellen Parametern ergänzt. Es werden englische Wörter für die Anweisungen verwendet, deren Bedeutung aber nicht gleich von anderen Skript- oder Programmiersprachen sein muss.

Skripte werden vor der Ausführung analysiert und auf einfache Syntaxfehler geprüft. Danach werden die einzelnen Anweisung in einem Thread nacheinander ausgeführt. Ein Skript ist unabhängig von anderen Skripten, und Skripte können auch gleichzeit ausgeführt werden. Gemeinsame Ressourcen, wie die Verbindung zu einem Messinstrument, werden geteilt. Damit der gleichzeitige Zugriff auf ein Messinstrument funktioniert, muss darauf geachtet werden, dass sich die Kommandos nicht gegenseitig stören.

Es gibt Beispielskripte im Verzeichnis 'Skript' für einige Instrumente.

6.1 Generell

Die Groß- und Kleinschreibung wird generell ignoriert. Die Namen der Konstanten, Variablen und Anweisungen werden intern in Kleinbuchstaben gewandelt. Texte die sich in Anführungszeichen befinden, werden unverändert übernommen.

Kommentare werden bei der Analyse eines Skript entfernt, ebenso führende und nachfolgende Leerzeichen oder Tabulatoren.

Eine Anweisung muss vollständig in einer Zeile ohne Umbruch eingegeben werden. Leerzeichen in Namen für Konstanten, Variablen und Anweisungen sind nicht erlaubt.

Der Dezimalpunkt bei Fließkommazahlen muss im englischen Format, mit einem Punkt als Kommazeichen angegeben werden.

Wenn der Wert einer Variable bei einer Ausgabe verwendet werden soll, muss dem Variablennamen ein Dollarzeichen ***$*** vorangestellt werden.

6.2 Referenz

Kommentar	Skripte und Anweisungen kommentieren.
Konstante	Konstanten verwenden.
Locale	Die Ausgabe des Dezimalpunkt bei Fließkommazahlen ändern.
Print	Nachrichten in dem Ausgabebereich des Editors anzeigen.
Set	Konstanten und Variablen Werte zuweisen.

Variable

Boolean	Variable für bool'sche Werte.
Integer	Variable für Ganzzahlen.
Float	Variable für Fließkommazahlen.
String	Variable für Zeichenketten.
Data	Variable für binäre Daten.
Image	Variable für Bilder
Waveform	Variable für Messkurven..

Mathematik

Add	Variablen und Literale addieren.
Sub	Variablen und Literale subtrahieren.
Mul	Variablen und Literale multiplizieren.
Div	Variablen und Literale dividieren.
Crc...	Prüfsumme von Variablen berechnen.

Sprungmarke

Label	Sprungmarke für eine Verzweigung.
Goto	Verzweigen zu einer Sprungmarke.
Return	Zu dem lezten Goto zurückspringen.
True	Verzweigen nach einem Vergleich.
False	Verzweigen nach einem Vergleich.

Vergleich

Equal	Auf Gleichheit prüfen.
Greater	Auf 'Größer als' prüfen.
Less	Auf 'Kleiner als' prüfen.

Instrument

USBTMC	USB-Schnittstelle mit TMC-Protokoll.
LXI	Netzwerkschnittstelle mit LXI bzw. RPC/VXI11-Protokoll.
RawTCP	Netzwerkschnittstelle ohne Protokoll.
Serial	Serielle RS232-Schnittstelle ohne Protokoll.

Funktion

Beep	Hinweis- oder Warnton ausgeben.
Clipboard	Variablen in die Zwischenablage kopieren.
Remove	Inhalte von Variablen entfernen.
Replace	Inhalte von Variablen suchen und ersetzen.
Save	Variablen in eine Datei speichern.
Scpi	SCPI-Kommandos senden und empfangen.
Screencopy	Eine Bildschirmkopie vom Messinstrument erstellen.
Search	Messinstrumente im Netzwerk und lokal am PC suchen.
View	Inhalte von Variablen anzeigen.
Wait	Warten auf Bestätigung oder Zeit.
Waveform	Messkurven vom Messinstrument einlesen.

6.2.1 Kommentar

Ein Kommentar beginnt mit doppelten Schrägstrich **//**. Jeder Text danach wird bis zum Zeilenende ignoriert.

Beispiel

```
// Das ist ein Kommentar.
instrument serial com1 // Kommentar.
#text set "Kein // Kommentar innerhalb von Anführungszeichen"
```

6.2.2 Konstante

Eine Konstante beginnt mit dem Nummernzeichen #. Es gibt keine Einschränkungen für den Namen der Konstante. Ein Wert wird der Konstante mit der Anweisung ***set*** zugewiesen. Alles nach ***set*** bis zu einem Kommentar oder dem Zeilende wird der Konstante zugewiesen.

Anführungszeichen werden nicht von dem zugewiesenen Wert entfernt, sondern bleiben erhalten. Es werden führende und nachfolgende Leerzeichen und Tabulatoren entfernt.

Eine Konstante kann nur einmal erstellt werden. Wird eine weitere Konstante mit demselben Namen erstellt, ergibt die Syntaxprüfung einen Fehler.

Wird die Konstante ohne der Anweisung ***set*** verwendet, wird ihr Name im Skript mit dem der Konstante zugewiesenen Wert ersetzt. Wird die Konstante innerhalb von Anführungszeichen verwendet, wird ihr Name nicht ersetzt.

Das Nummernzeichen # kann verwendet werden, wenn es keine erstellte Konstante mit dem Namen gibt. Es wird kein Fehler bei der Syntaxprüfung ausgegeben. Damit kann das Zeichen z.B. bei SCPI-Kommandos verwendet werden.

Beispiel

```
#ip set 192.168.1.1
#echo set :SYSTem:ECHO? "Hallo"
#file set "T:\Daten\Test.csv"
#max set 50 // Dieses Kommentar wird ignoriert.
```

6.2.3 Locale

Die Anweisung ***locale*** beeinflusst die Ausgabe von Fließkommazahlen als Text. Standardmäßig werden Zahlen im englischen Format mit Punkt als Dezimalzeichen ausgegeben. Soll die lokale Einstellung des Betriebssystems verwendet werden, muss ein bool'scher Parameter für Wahr angegeben werden. Erlaubt sind die Parameter ***true***, ***on***, ***1*** und ***false***, ***off***, ***0***.

Beispiel

```
locale true
locale off
locale 1
```

6.2.4 Print

Die Anweisung ***print*** gibt den angegebenen Text in dem Ausgabebereich des jeweiligen Editor aus. Es können Konstanten und Variablen angegeben werden. Damit der Inhalt der Variablen ausgegeben wird, muss dem Variablenname ein Dollarzeichen ***$*** vorangestellt werden.
print kann auch mit den Anweisungen ***true*** oder ***false*** nach einem Vergleich verwendet werden.
Die Länge der Ausgabe ist auf 100 Zeichen beschränkt.

Beispiel

```
print Dieser Text erscheint in der Ausgabe.

#Text set Dieser Text erscheint in der Ausgabe.
print #Text
```

```
float pi
pi set 3.14
locale on
print Pi ist $pi // Ausgabe: Pi ist 3,14
```

6.2.5 Set

Einer Konstante oder Variable kann ein Wert mit der Anweisung ***set*** zugewiesen werden. Für die Zuweisung können Literale, Konstanten oder Variablen verwendet werden. Bei der Zuweisung wird alles bis zu einen Kommentar oder dem Zeilenende berücksichtigt. Zuweisungen von unterschiedlichen Datentypen werden automatisch konvertiert, siehe dazu die Beschreibung zu den jeweiligen Datentypen.

Bei Konstanten werden Anführungszeichen bei der Zuweisung nicht entfernt. Bei Variablen werden die Anführungszeichen entfernt wenn sie nur am Anfang und Ende vorhanden sind.

Die Anweisung ***set*** wird auch für die Verbindungseinstellung der Messinstrumente verwendet, siehe dazu Instrument und die entsprechende Schnittstelle..

Beispiel

```
#Datei set "T:\Datei.csv"
print #Datei // Ausgabe: "T:\Datei.csv"

integer Zahl
Zahl set "42"
print $Zahl // Ausgabe: 42

string Text
Text set "Hello World"
print $Text // Ausgabe; Hello World
Text set "Hello" World
print $Text // Ausgabe: "Hello" World
```

Spezielle $-Werte

Für die Zuweisung an Variablen können spezielle Werte verwendet werden. Die Werte beginnen mit einem Dollarzeichen ***$*** und können mit den Anweisungen ***set*** und ***add*** verwendet werden. Die Werte werden bei der Zuweisung wie Literale behandelt.

$eol	Zeilenumbruch mit CR,LF
$date	Das aktuelle Datum: 2023-05-09
$time	Die aktuelle Uhrzeit: 16:32:13
$datetime	Das aktuelle Datum mit Uhrzeit: 2023-05-09 16:32:13
$time2	Die aktuelle Uhrzeit mit Millisekunden: 16:32:13.275
$datetime2	Das aktuelle Datum mit Uhrzeit und Millisekunden: 2023-05-09 16:32:13.277
$file "T:\Datei..."	Lädt den Inhalt der angegebenen Datei.
$block	Erstellt ein SCPI-Blockformat mit der Länge der Daten am Anfang, z.B: #510200

Beispiel

```
string Text
Text set Zeile 1
Text add $eol
Text add Zeile 2
print $Text

Text set $date
print $Text // Ausgabe: 2023-05-09

Text set ABCDEFGHIJKLMNOPQRSTUVWXYZ
Text add $block
print $Text // Ausgabe: #226ABCDEFGHIJKLMNOPQRSTUVWXYZ

data Firmware
Firmware set $file "T:\Update.bin"
Firmware add $block
```

6.2.6 Variable

Eine Variable wird mit einer Anweisung für einen Datentyp erstellt und anschließenden Variablennamen. Der Name darf nicht gleich lauten wie die meisten Skriptanweisungen, ansonst wird ein Fehler bei der Syntaxprüfung ausgegeben.

Die Variable muss zuerst erstellt werden, und danach kann ihr mit der Anweisung ***set*** ein Wert zugewiesen werden. Es können Literale, Konstanten oder andere Variablen für die Zuweisung verwendet werden. Unterschiedliche Datentypen bei Variablen werden automatisch konvertiert.

Sind bei einem Literal Anführungszeichen vorhanden, werden diese entfernt.

Eine Variable kann mehrmals mit dem gleichen Datentyp erstellt werden. Werden verschiedene Datentypen angegeben, wird bei der Syntaxprüfung ein Fehler ausgegeben.

Datentypen:

boolean	Für bool'sche Werte Wahr oder Falsch.
integer	Für ganze Zahlen.
float	Für Fließkommazahlen.
string	Für Zeichenfolgen.
data	Für binäre Daten die von Messinstrumenten eingelesen werden.
image	Für Bilder die von Messinstrumenten eingelesen werden.
waveform	Für Messkurven die von Messinstrumenten eingelesen werden.

Beispiel

```
boolean Test
Test set true

integer Counter
Counter set 12
```

```
float Pi
Pi set 3.14

string Text
Text set Bitte Taste drücken.

data Trace

image Screencopy

waveform Logic
```

6.2.6.1 Boolean

Einer Variable vom Typ ***boolean*** können nur die Werte ***true*** oder ***false*** zugewiesen werden.

Beispiel

```
boolean Result
Result set true
```

Wird einer bool'schen Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird der Wert automatisch konvertiert. Wird ein Literal zugewiesen, wird nur die Angabe von ***true*** als Wahr interpretiert, alles andere als ***false***.

Datentyp	**Wert**	**Ergebnis**
integer	Null	false
	ungleich Null	true
float	Null	false
	ungleich Null	true
string	"true"	true
	ungleich "true"	false
data	Datenlänge ist Null	false
	Interpretation als String der leer ist, oder nur Leerzeichen enthält.	false
	Interpretation als String der Zeichen enthält.	true
	Interpretation als Fließkommazahl deren Wert Null ist.	false
	Interpretation als Fließkommazahl deren Wert ungleich Null ist.	true
image	Es ist kein Bild vorhanden.	false
	Es ist ein Bild vorhanden.	true
waveform	Es sind keine Messkurven vorhanden.	false
	Es sind Messkurven vorhanden.	true

6.2.6.2 Integer

Einer Variable vom Typ ***integer*** können Ganzahlen zugewiesen werden. Der Wert wird intern als Quad-Wort mit 8 Bytes gespeichert. Der erlaubte Bereich der Werte ist von -9223372036854775808 bis +9223372036854775807.

Beispiel

```
integer Counter
Counter set 12
```

Wird einer ***integer*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird der Wert automatisch konvertiert. Wird ein Literal zugewiesen, wird die Angabe als Ganzzahl interpretiert.

Datentyp	**Wert**	**Ergebnis**
boolean	false	0
	true	1
float	Wird mathematisch auf eine Ganzahl gerundet.	Ganzzahl
string	Wird als Ganzzahl interpretiert.	Ganzzahl
data	Datenlänge ist Null.	0
	Datenlänge ist ungleich Null.	1
image	Es ist kein Bild vorhanden.	0
	Es ist ein Bild vorhanden.	1
waveform	Es sind keine Messkurven vorhanden.	0
	Es sind Messkurven vorhanden.	1

6.2.6.3 Float

Einer Variable vom Typ ***float*** können Fließkommazahlen zugewiesen werden. Der Wert wird intern als Double-Wort mit 8 Bytes gespeichert. Der erlaubte Bereich der Werte ist von +-2.2250738585072013e-308 bis +- 1.7976931348623157e+308.

Beispiel

```
float Pi
Pi set 3.14
```

Wird einer ***float*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird der Wert automatisch konvertiert. Wird ein Literal zugewiesen, wird die Angabe als Fließkommazahl interpretiert.

Datentyp	Wert	Ergebnis
boolean	false	0
	true	1
integer	Wird in eine Fließkommazahl konvertiert.	Fließkommazahl
string	Wird als Fließkommazahl interpretiert.	Fließkommazahl
data	Datenlänge ist Null.	0
	Datenlänge ist ungleich Null.	1
image	Es ist kein Bild vorhanden.	0
	Es ist ein Bild vorhanden.	1
waveform	Es sind keine Messkurven vorhanden.	0
	Es sind Messkurven vorhanden.	1

6.2.6.4 String

Einer Variable vom Typ ***string*** können Zeichenfolgen zugewiesen werden. Die Länge der Zeichenfolge ist nicht beschränkt.

Beispiel

```
string Text
Text set Foo Bar
Text set "Foo" Bar" // Die äußeren Anführungszeichen werden entfernt.
```

Wird einer ***string*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird der Wert automatisch konvertiert.

Datentyp	Wert	Ergebnis
boolean	false	false
	true	true
integer	Wird in einen String konvertiert.	String
float	Wird in einen String konvertiert, und berücksichtigt ***locale***.	String
data	Datenlänge ist Null.	Leerer String
	Interpretation als String der leer ist, oder nur Leerzeichen enthält.	Leerer String
	Interpretation als ASCII-Zeichenfolge.	String
image	Es ist kein Bild vorhanden.	false
	Es ist ein Bild vorhanden.	true
waveform	Es sind keine Messkurven vorhanden.	false
	Es sind Messkurven vorhanden.	true

6.2.6.5 Data

Einer Variable vom Typ ***data*** können binäre Daten zugewiesen werden. Die Daten können von einem Messinstrument oder von einer Datei eingelesen werden.

Beispiel

```
data Einstellung
Einstellung set $file "T:\Daten\Einstellung.bin"
```

Wird einer ***data*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird während der Ausführung des Skript ein Fehler gemeldet. Bei der Syntaxprüfung wird dieser Fehler nicht erkannt.

Datentyp	Wert	Ergebnis
boolean	-	Fehler
integer	-	Fehler
string	-	Fehler
data	Datenlänge ist Null.	Data
	Datenlänge ist ungleich Null.	Data
image	-	Fehler
waveform	-	Fehler

6.2.6.6 Image

Einer Variable vom Typ ***image*** kann ein Bild zugewiesen werden. Das Bild kann nur von einem Messinstrument eingelesen werden. Das Einlesen von einer Datei ist nicht möglich.

Eine Variable vom Datentyp ***data*** kann zugewiesen werden, wenn die enthaltenen Daten ein gültiges Bildformat enthalten. Wird bei der Zuweisung kein Bild erkannt, wird ein Skriptfehler bei der Ausführung angezeigt.

Beispiel

```
image Bildschirmkopie
Bildschirmkopie function screencopy
```

Wird einer ***image*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird während der Ausführung des Skript ein Fehler gemeldet. Bei der Syntaxprüfung wird dieser Fehler nicht erkannt.

Datentyp	Wert	Ergebnis
boolean	-	Fehler
integer	-	Fehler
string	-	Fehler
data	-	Fehler
	Ein gültiges Bildformat ist in den Daten vorhanden.	Image
image	Ein Bild ist nicht vorhanden.	Image
	Ein Bild ist vorhanden.	Image
waveform	-	Fehler

Beispiel für Daten mit gültigen Bildformat

```
// Variablen erstellen.
data Bildschirmdaten
image Bildschirmkopie

// Die Bildschirmdaten einlesen.
Bildschirmdaten function scpi read :DISP:DATA?

// Das SCPI-Blockformat am Anfang der Daten,
// und das Terminierungszeichen am Ende entfernen.
Bildschirmdaten function remove block last 1

// Die Bildschirmdaten der Variable mit Datentyp image zuweisen.
Bildschirmkopie set Bildschirmdaten
```

6.2.6.7 Waveform

Einer Variable vom Typ ***waveform*** können Messkurven zugewiesen werden. Die Messkurven können nur von einem Messinstrument eingelesen werden. Das Einlesen von einer Datei ist nicht möglich.

Beispiel

```
waveform Logikkanäle
Logigkanäle function waveform logic
```

Wird einer ***waveform*** Variable ein Wert von einer anderen Variable mit unterschiedlichen Datentyp zugewiesen, wird während der Ausführung des Skript ein Fehler gemeldet. Bei der Syntaxprüfung wird dieser Fehler nicht erkannt.

Datentyp	Wert	Ergebnis
boolean	-	Fehler
integer	-	Fehler
string	-	Fehler
data	-	Fehler
image	-	Fehler
waveform	Es sind keine Messkurven vorhanden.	Waveform
waveform	Es sind Messkurven vorhanden.	Waveform

6.2.7 Mathematik

Die mathematischen Anweisungen sind sehr einfach und können immer nur einen Wert auf einmal verarbeiten. Es kann nur mit Variablen gerechnet werden, aber Literale als Werte sind erlaubt. Werden Variablen mit unterschiedlichen Datentyp verwendet, gelten die gleichen Regeln wie bei der Zuweisung, siehe dazu die Beschreibung zu den jeweiligen Datentypen.

Anweisungen

add	Addieren von Zahlen, Zeichenfolgen und Daten.
sub	Subtrahieren von Zahlen.
mul	Multiplizieren von Zahlen.
div	Dividieren von Zahlen.
crc...	Berechnen einer Prüfsumme von Daten.

6.2.7.1 Add

Mit der Anweisung ***add*** können zu Variablen Literale oder Variablen addiert werden. Bei Variablen vom Datentyp ***string, data*** und ***waveform,*** werden die Werte nicht addiert, sondern am Ende angefügt.

Beispiel

```
integer Zahl
Zahl set 3
```

```
Zahl add 4
print $Zahl // Ausgabe: 7

float Pi
Pi set 3.14
Zahl add Pi
print $Zahl // Ausgabe: 10

string Text
Text set Zahl
Text add " "
Text add Pi
print $Text // Ausgabe: 10 3.14

Text add " "
Text add $date // Für spezielle $-Werte siehe set
print $Text // Ausgabe: 10 3.14 2023-05-09
```

Variablen vom Typ ***data*** können nur zu Variablen mit demselben Datentyp am Ende hinzugefügt werden.

Beispiel

```
data Daten1
data Daten2
data DatenAlle
Daten1 set $file "T:\Daten1.bin"
Daten2 set $file "T:\Daten2.bin"
DatenAlle set Daten1
DatenAlle add Daten2
DatenAlle add $block
```

Variablen vom Typ ***waveform*** können nur zu Variablen mit demselben Datentyp hinzugefügt werden. Zurzeit werden nur Logik-Messkurven unterstützt, alle anderen Typen wie Analog, Math usw. werden ignoriert.

Beispiel

```
waveform Logik1
waveform Logik2
Logik1 function waveform logic
Logik2 function waveform logic
Logik1 add Logik2
Logik1 function save "T:\Logik" vcd
```

6.2.7.2 Sub

Mit der Anweisung **sub** können von Variablen Literale oder Variablen subtrahiert werden. Es werden die Variablen vom Datentyp ***integer*** und ***float*** unterstützt. Bei allen anderen Datentypen ist die Berechnung nicht erlaubt.

Beispiel

```
float Volt
Volt sub 5.2
print $Volt // Ausgabe: -5.2

integer Volt2
Volt2 sub Volt
```

```
print $Volt2 // Ausgabe: 5
```

6.2.7.3 Mul

Mit der Anweisung **mul** können von Variablen mit Literale oder Variablen multipliziert werden. Es werden die Variablen vom Datentyp ***integer*** und ***float*** unterstützt. Bei allen anderen Datentypen ist die Berechnung nicht erlaubt.

Beispiel

```
float Volt
integer Ampere
float Watt
Volt set 5.2
Ampere set 2
Watt set Volt
Watt mul Ampere
print $Watt // Ausgabe: 10.4
```

6.2.7.4 Div

Mit der Anweisung ***div*** können Variablen mit Literale oder Variablen dividiert werden. Es werden die Variablen vom Datentyp ***integer*** und ***float*** unterstützt. Bei allen anderen Datentypen ist die Berechnung nicht erlaubt. Eine Division durch Null wird ignoriert und löst keinen Fehler aus.

Beispiel

```
integer Ampere
float Watt
float Volt
Ampere set 2
Watt set 10.4
Volt set Watt
Volt div Ampere
print $Volt // Ausgabe: 5.2

Volt div 0 // Division durch Null wird ignoriert.
print $Volt // Ausgabe; 5.2
```

6.2.7.5 Crc...

Mit der Anweisung ***crc...*** kann von Variablen eine Prüfsumme berechnet werden.

crc16ccitt

Von Variablen mit dem Datentyp ***data*** kann mit der Anweisung ***crc16ccitt*** eine Prüfsumme berechnet werden. Die Prüfsumme kann nur einer Variable vom Datentyp ***integer*** zugewiesen werden. Die Prüfsumme kann z.B. für ein Firmware-Update beim R&S RTB2000 Oszilloskop verwendet werden. Die Variante der Prüfsumme nennt sich auch CRC-16/IBM-3740, CRC-16/CCITT-FALSE oder CRC-16/CCITT (0xFFFF). Ein ASCII-Text von 123456789 ergibt 0x29B1 als Prüfsumme.

Beispiel

```
#Datei set "T:\Update.bin"
string Text
Text set 123456789
Text function save #Datei

data Firmware
integer Crc
Firmware set $file #Datei
Crc crc16ccitt Firmware
print $Crc // Ausgabe: 10673
```

6.2.8 Sprungmarke

Mit Sprungmarken, die mit der Anweisung ***label*** erstellt werden, kann man den Skriptablauf steuern. Es kann im Ablauf direkt zu einer Sprungmarke verzweigt werden, oder nach dem Vergleich einer Variable, je nach Ergebnis.

Für den Namen einer Sprungmarke gibt es keine Einschränkungen. Wird eine gleichnamige Sprungmarke mehrmals erstellt, wird bei der Syntaxprüfung ein Fehler ausgegeben.

Mit der Anweisung ***goto*** wird zu der angegebenen Sprungmarke verzweigt. Wurde vorher ein Vergleich mit einer Variable durchgeführt, kann mit den Anweisungen ***true*** oder ***false*** und nachfolgender ***goto***-Anweisung zu einer Sprungmarke verzweigt werden. Das Ergebnis eines Vergleichs mit den Anweisungen ***equal***, ***greater*** oder ***less***, bleibt solange verfügbar, bis ein neuer Vergleich durchgeführt wird.

Label	Sprungmarke erstellen.
Goto	Zu einer Sprungmarke verzweigen.
Return	Zu dem letzten Goto zurückspringen.
True	Nach einem Vergleich verzweigen.
False	Nach einem Vergleich verzweigen.

Vergleichsanweisungen

Equal	Vergleich einer Variable auf Gleichheit.
Greater	Vergleich einer Variable auf 'Größer als'.
Less	Vergleich einer Variable auf 'Kleiner als'.

Beispiel

```
#KeinFehler set 0,"No error" // Antwort vom Messinstrument.
string Antwort
Antwort function scpi read :SYSTem:ERRor?
Anwort equal #KeinFehler // Vergleich der Variable mit einer Konstante.
false goto Fehler // Sprung wenn die Anwort und die Konstante unterschiedlich
sind.
print Es ist kein Fehler aufgetreten.

goto Ende // Ein "true goto ende" funktioniert auch,
true goto Ende // da das Ergebnis des Vergleichs verfügbar bleibt.
```

```
label Fehler
print Es ist ein Fehler aufgetreten: $Antwort

label Ende
```

6.2.8.1 Label

Mit der Anweisung ***label*** und einem nachfolgenden Namen wird eine Sprungmarke erstellt. Der Name der Sprungmarke kann beliebig gewählt erden. Wird ein Name für eine Sprungmarke mehrmals verwendet, wird ein Fehler bei der Syntaxprüfung ausgegeben.

Beispiel

```
goto Weiter

label Weiter
print Weiter
goto Ende des Skript

label Ende des Skript
print Ende
```

6.2.8.2 Goto

Mit der Anweisung ***goto*** und der Angabe einer Sprungmarke, kann zu dieser im Skript verzweigt werden. Wenn auf das Ergebnis eines Vergleichs verzweigt werden soll, können die Anweisungen ***true*** oder ***false*** mit ***goto*** verwendet werden. Das Ergebnis eines Vergleichs bleibt solange erhalten, bis ein neuer Vergleich durchgeführt wird. Wurde kein Vergleich durchgeführt, ist das Ergebnis falsch.

Beispiel

```
true goto True
false goto False
goto Ende

label True
print True
goto Ende

label False
print False

label Ende
```

6.2.8.3 Return

Mit der Anweisung ***return*** kann zu der zuletzt ausgeführten ***goto***-Anweisung verzweigt werden. Wenn auf das Ergebnis eines Vergleichs verzweigt werden soll, können die Anweisungen ***true*** oder ***false*** mit ***return*** verwendet werden. Das Ergebnis eines Vergleichs bleibt solange erhalten, bis ein neuer Vergleich durchgeführt wird. Wurde kein Vergleich durchgeführt, ist das Ergebnis falsch.

Beispiel

```
integer Zähler
```

```
Zähler add 1
print $Zähler
goto Prüfen

Zähler add 1
print $Zähler
goto Prüfen

print Dieser Text wird nicht ausgegeben.

label Prüfen
Zähler equal 2
false return
```

6.2.8.4 True

Die Anweisung ***true*** kann mit den Anweisungen ***goto***, ***return***, ***function*** und ***print*** verwendet werden. Wurde ein Vergleich einer Variable mit einer Vergleichsanweisung durchgeführt, kann entsprechend dem Ergebnis im Skript zu einer Sprungmarke verzweigt, oder die angegebene Anweisung ausgeführt werden.

Beispiel

```
integer Counter

label Count
Counter add 1
Counter equal 3
false print Es wird gezählt: $Counter
true goto Ende
goto Count

label Ende
true function beep
print Der Zähler ist $Counter.
```

6.2.8.5 False

Die Anweisung ***false*** kann mit den Anweisungen ***goto***, ***return***, ***function*** und ***print*** verwendet werden. Wurde ein Vergleich einer Variable mit einer Vergleichsanweisung durchgeführt, kann entsprechend dem Ergebnis im Skript zu einer Sprungmarke verzweigt, oder die angegebene Anweisung ausgeführt werden.

Beispiel

```
integer Counter

label Count
Counter add 1
Counter equal 3
false print Es wird gezählt: $Counter
false goto Count

true function beep
print Der Zähler ist $Counter.
```

6.2.9 Vergleich

Die Anweisungen ***equal***, ***greater*** und ***less*** können mit den Anweisungen für Sprungmarken verwendet werden. Die Vergleichsanweisungen ermöglichen den Vergleich von Variablen mit Literalen, Konstanten und anderen Variablen. Das Ergebnis eines Vergleichs ist wahr oder falsch, und steht bis zu einem nächsten Vergleich zur Verfügung. Es kann z.B. mit ***true goto*** und später im Skript mit ***false goto*** je nach Ergebnis im Skript verzweigt werden.

Vergleichsanweisungen

Equal	Vergleich einer Variable auf Gleichheit.
Greater	Vergleich einer Variable auf Größer als.
Less	Vergleich einer Variable auf Kleiner als.

6.2.9.1 Equal

Die Anweisung ***equal*** vergleicht eine Variable auf 'Gleichheit' mit einer Konstante, Literal oder anderen Variable. Das Ergebnis des Vergleichs ist Wahr oder Falsch. Das Ergebnis des Vergleichs bleibt solange erhalten, bis ein nächster Vergleich mit einer der Vergleichsanweisungen durchgeführt wird.

Die Regeln für einen Vergleich mit unterschiedlichen Datentypen, sind gleich wie bei der Zuweisung des Datentyps der verglichen wird. Siehe dazu den passenden Datentyp unter Variable.

Beispiel

```
integer Counter
Counter set 5
Counter equal 10
true print 5 ist gleich 10. // Wird nicht augegeben.
false print 5 ist ungleich 10. // Wird ausgegeben.
```

6.2.9.2 Greater

Die Anweisung ***greater*** vergleicht eine Variable auf 'Größer als' mit einer Konstante, Literal oder anderen Variable. Das Ergebnis des Vergleichs ist Wahr oder Falsch. Das Ergebnis des Vergleichs bleibt solange erhalten, bis ein nächster Vergleich mit einer der Vergleichsanweisungen durchgeführt wird.

Die Regeln für einen Vergleich mit unterschiedlichen Datentypen, sind gleich wie bei der Zuweisung des Datentyps der verglichen wird. Siehe dazu den passenden Datentyp unter Variable.

Beispiel

```
integer Counter
Counter set 10
Counter greater 5
true print 10 ist größer als 5. // Wird augegeben.
false print 10 ist nicht größer als 5. // Wird nicht ausgegeben.
```

6.2.9.3 Less

Die Anweisung ***less*** vergleicht eine Variable auf 'Kleiner als' mit einer Konstante, Literal oder anderen Variable. Das Ergebnis des Vergleichs ist Wahr oder Falsch. Das Ergebnis des Vergleichs bleibt solange erhalten, bis ein nächster Vergleich mit einer der Vergleichsanweisungen durchgeführt wird.

Die Regeln für einen Vergleich mit unterschiedlichen Datentypen, sind gleich wie bei der Zuweisung des Datentyps der verglichen wird. Siehe dazu den passenden Datentyp unter Variable.

Beispiel

```
integer Counter
Counter set 10
Counter less 5
true print 10 ist kleiner als 5. // Wird nicht augegeben.
false print 10 ist nicht kleiner als 5. // Wird ausgegeben.
```

6.2.10 Instrument

Mit der Anweisung ***instrument*** wird eine Verbindung zu dem angegebenen Messinstrument hergestellt. Kann keine Verbindung hergestellt werden, wird das Skript mit einem Fehler angehalten.

Es können mehrere Instrumente in einem Skript verwendet werden. Damit ein Instrument ausgewählt werden kann, muss ein Name angegeben werden. Der Name für das Instrument bzw. Verbindung muss mit einem Dollarzeichen ***$*** beginnen. Es wird immer das zuletzt geöffnete Messintrument für die weitere Verwendung ausgewählt. Soll ein anderes Instrument ausgewählt werden, muss der vergebene Name alleine in einer Zeile angegeben werden.

Die Verbindung zu einem Messinstrument wird nach beenden eines Skripts geschlossen. Wird das Instrument in mehreren Skripten verwendet, bleibt die Verbindung bestehen bis diese Skripte beendet werden.

Wenn ein Name für eine Verbindung verwendet wird, können Einstellung zu Verbindung gesetzt bzw. verändert werden. Dazu muss der Name mit der Anweisung ***set*** und der gewünschten Einstellung angegeben werden.

Beispiel

```
instrument vxi 192.168.1.100

instrument $DMM serial com3 9600
$DMM set timeout 2000

instrument $Oszi usbtmc 0AAD 01D6
$Oszi
function scpi write :SYSTem:PRESet
```

Schnittstellen

USBTMC	USB-Verbindung für Test & Measurement Class.
LXI	Netzwerkverbindung mit LXI bzw. RPC/VXI11-Protokoll.
RawTCP	Netzwerkverbindung ohne Protokoll.
Serial	Serielle RS232-Verbindung ohne Protokoll.

6.2.10.1 USBTMC

Die Anweisung ***usbtmc*** erstellt eine Verbindung zu einem USB-Messinstrument der Klasse Test & Measurement Class. Damit diese Verbindung verwendet werden kann. muss der WinUSB-Treiber installiert sein, siehe dazu Installation.

Für die Verbindung muss die Vendor- und Product-ID in hexadezimaler Schreibweise angegeben werden. Optional kann eine Seriennummer angegeben werden, wenn mehrere Instrumente mit gleicher VID und PID verwendet werden.

```
instrument [$name] usbtmc VID PID [Seriennummer]
```

Beispiel

```
instrument $DMM usbtmc 164E 0DAD
```

Verbindungseinstellungen

Wenn ein Name für die Verbindung verwendet wird, können folgende Einstellungen mit der Anweisung ***set*** für die Verbindung vorgenommen werden.

timeout <Millisekunden>	Setzt die Wartezeit wie lange auf eine Antwort vom Instrument gewartet wird, Wird 0 (Null) angegeben oder keine Zeit, wird ein Standardwert für die Wartezeit verwendet.
wait scpi <Millisekunden>	Setzt die Wartezeit in Millisekunden, die zwischen einzelnen SCPI-Anweisungen ***scpi read/write*** gewartet wird.
wait stbmav	Beim Einlesen von Daten vom Instrument, wird gewartet bis das MAV-Bit im Statusbyte gesetzt wird.
wait srq	Beim Einlesen von Daten vom Instrument, wird auf den Service Request-Interrupt gewartet.
wait	Ohne Parameter wird nicht mehr auf das Bereitstellen von Daten gewartet. Die Optionen ***stbmav*** und ***srq*** werden gelöscht.

Beispiel

```
instrument $DMM usbtmc 164E 0DAD
$DMM set timeout 2000
$DMM set wait scpi 100
```

Beispiel - wait stbmav

```
float Volt
instrument $DMM usbtmc 164E 0DAD
$DMM set wait stbmav
Volt function scpi read :INIT;:FETCH?
$DMM set wait
```

Beispiel - wait srq

```
float Volt
instrument $DMM usbtmc 164E 0DAD
$DMM set wait srq
function scpi write *CLS
function scpi write *ESE 0;*SRE 16
Volt function scpi read :INIT;:FETCH?
function scpi write *SRE 0
$DMM set wait
```

6.2.10.2 LXI

Die Anweisung ***lxi*** erstellt eine Verbindung zu einem Messinstrument über eine Netzwerkverbindung mit dem Protokoll RPC/ VXI11.
Für die Verbindung muss die IP-Adresse des Instruments angegeben werden. Wird der VXI-Port nicht angegeben, oder 0 (Null), wird der Port vom Instrument abgefragt. Dafür wird der RPC-Port 111 mit dem UDP- und TCP-Protokoll abgefragt.
Sind mehrere Netzwerkkarten im PC vorhanden, kann die IP-Adresse der gewünschten Netzwerkkarte zuletzt angegeben werden. Ohne Angabe wird versucht die passende Netzwerkkarte automatisch zu ermitteln.

```
instrument [$name] lxi IP-Adresse [VXI-Port [IP-Adresse der Netzwerkkarte]]
```

Beispiel

```
instrument lxi 192.168.1.101
instrument $RTB lxi 192.168.1.102 1024
instrument lxi 192.168.1.103 0 192.168.200.60
```

Verbindungseinstellungen

Wenn ein Name für die Verbindung verwendet wird, können folgende Einstellungen mit der Anweisung ***set*** für die Verbindung vorgenommen werden.

timeout <Millisekunden>	Setzt die Wartezeit wie lange auf eine Antwort vom Instrument gewartet wird, Wird 0 (Null) angegeben oder keine Zeit, wird ein Standardwert für die Wartezeit verwendet.
wait scpi <Millisekunden>	Setzt die Wartezeit in Millisekunden, die zwischen einzelnen SCPI-Anweisungen ***scpi read/write*** gewartet wird.
local	Entsperrt das Instrument für die lokale Bedienung.
remote	Sperrt das Instrument für die entfernte Bedienung.

Beispiel

```
instrument $RTB lxi 192.168.1.102
```

```
$RTB set timeout 2000
$RTB set wait scpi 100
$RTB set local
$RTB set remote
```

6.2.10.3 RawTCP

Die Anweisung ***rawtcp*** erstellt eine Verbindung zu einem Messinstrument über eine Netzwerkverbindung. Es wird kein Protokoll verwendet, die SCPI-Kommandos werden mit einem Terminierungszeichen gesendet.
Für die Verbindung muss die IP-Adresse und der Port des Instruments angegeben werden. Sind mehrere Netzwerkkarten im PC vorhanden, kann die IP-Adresse der gewünschten Netzwerkkarte zuletzt angegeben werden. Ohne Angabe wird versucht die passende Netzwerkkarte automatisch zu ermitteln.

```
instrument [$name] rawtcp IP-Adresse Port [IP-Adresse der Netzwerkkarte]
```

Beispiel

```
instrument rawtcp 192.168.1.101
instrument $Hameg rawtcp 192.168.1.102 5025
instrument rawtcp 192.168.1.103 5025 192.168.200.60
```

Verbindungseinstellungen

Wenn ein Name für die Verbindung verwendet wird, können folgende Einstellungen mit der Anweisung ***set*** für die Verbindung vorgenommen werden.

timeout <Millisekunden>	Setzt die Wartezeit wie lange auf eine Antwort vom Instrument gewartet wird, Wird 0 (Null) angegeben oder keine Zeit, wird ein Standardwert für die Wartezeit verwendet.
wait scpi <Millisekunden>	Setzt die Wartezeit in Millisekunden, die zwischen einzelnen SCPI-Anweisungen ***scpi read/write*** gewartet wird.
termchar <Zeichen,...>	Setzt ein oder mehrere Terminierungszeichen, die am Ende der übertragenen Daten vorhanden sind. Wird kein Zeichen angegeben, wird beim Empfang der Daten nur die Wartezeit verwendet. Die Zeichen müssen in hexadezimaler Schreibweise angegeben werden.

Beispiel

```
instrument $Hameg rawtcp 192.168.1.102 5025
$Hameg set timeout 2000
$Hameg set wait scpi 100
$Hameg set termchar 0A,0D
```

6.2.10.4 Serial

Die Anweisung ***serial*** erstellt eine Verbindung zu einem Messinstrument über einen seriellen RS232-Anschluss oder virtuellen COM-Port. Es wird kein Protokoll verwendet, die SCPI-Kommandos werden mit einem Terminierungszeichen gesendet.
Für die Verbindung muss der Name des Anschlusses angegeben werden. Die weiteren Einstellungen für den Anschluss sind optional. Ohne Angabe werden Standardwerte verwendet.

```
instrument [$name] serial Portnummer [Baudrate [Datenbits [Stoppbits [Parität
[Flusskontrolle]]]]]
```

Einstellungen

Portnummer	Com1, Com2 usw., oder ein anderer gültiger Schnittstellenname.
Baudrate	9600, ... 115200 usw.
Datenbits	7 oder 8
Stoppbits	1, 1.5 oder 2
Parität	None, Odd, Even, Mark oder Space
Flusskontrolle	None, RTSON, RTSCSTS oder XON

Standardwerte

Baudrate	9600
Datenbits	8
Stoppbits	1
Parität	None
Flusskontrolle	None

Beispiel

```
instrument serial COM1
instrument $Tonghui serial COM2 115200 8 1
instrument serial COM2 115200 8 1 none none
```

Verbindungseinstellungen

Wenn ein Name für die Verbindung verwendet wird, können folgende Einstellungen mit der Anweisung ***set*** für die Verbindung vorgenommen werden.

timeout <Millisekunden>	Setzt die Wartezeit wie lange auf eine Antwort vom Instrument gewartet wird, Wird 0 (Null) angegeben oder keine Zeit, wird ein Standardwert für die Wartezeit verwendet.
wait scpi <Millisekunden>	Setzt die Wartezeit in Millisekunden, die zwischen einzelnen SCPI-Anweisungen ***scpi read/write*** gewartet wird.
termchar <Zeichen,...>	Setzt ein oder mehrere Terminierungszeichen, die am Ende der übertragenen Daten vorhanden sind. Wird kein Zeichen angegeben, wird beim Empfang der Daten nur die Wartezeit verwendet. Die Zeichen müssen in hexadezimaler Schreibweise angegeben werden.

Beispiel

```
instrument $Hameg serial COM2 115200
$Hameg set timeout 500
$Hameg set wait scpi 100
$Hameg set termchar 0D
```

6.2.11 Function

Mit der Anweisung ***function*** können verschiedene Funktionen ausgeführt werden. Einige Funktionen geben Werte zurück, die in Variablen gespeichert werden können, andere geben keinen Wert zurück. Funktionen können auch mit den Anweisungen ***true*** oder ***false*** nach einem Vergleich verwendet werden.

Funktionen

Beep	Gibt einen Hinweiston aus.
Clipboard	Kopiert den Inhalt einer Variable in die Zwischenablage.
Remove	Entfernt Teile von String oder Data-Variablen.
Replace	Ersetzt Teile von String oder Data-Variablen.
Save	Speichert den Inhalt einer Variable in eine Datei.
Scpi	Sendet und empfängt SCPI-Kommandos.
Screencopy	Erstellt eine Bildschirmkopie von einem Messinstrument.
View	Zeigt den Inhalt einer Variable an.
Wait	Wartet auf eine Bestätigung, ein Datum oder Zeitspanne.
Waveform	Liest Messkurven von einem Messinstrument ein.

6.2.11.1 Beep

Mit der Anweisung ***beep*** und optionalen Parameter wird ein Hinweiston ausgegeben. Es werden die Systemklänge verwendet, wie sie in der Systemsteuerung unter 'Sound' eingestellt sind. Ohne Parameter wird der Standardton ausgegeben.

```
function beep [sound]
```

Parameter

ok	Standardton Warnsignal *
information	Sternchen *
question	Frage *
warning	Hinweis *
error	Kritischer Abbruch *

* Windows 7 Sound-Einstellungen.

Beispiel

```
function beep
function beep question
function beep error
```

6.2.11.2 Clipboard

Mit der Anweisung ***clipboard*** wird der Inhalt einer Variable in die Zwischenablage kopiert. Mit Ausnahme vom Datentyp image werden die Inhalte als konvertierter Text kopiert.

```
variable function clipboard
```

Datentyp

boolean	Text true oder false
integer	Text Zahl
float	Text Zahl
string	Text
data	Hex- und ASCII-Darstellung der ersten 256 und letzten 128 Bytes.
image	Bild
waveform	Ist nicht erlaubt. Ein Fehler wird bei der Ausführung angezeigt.

Beispiel

```
integer Counter
Counter set 7
Counter function clipboard
```

6.2.11.3 Remove

Mit der Anweisung ***remove*** werden Inhalte von Variablen, mit dem Datentyp ***string*** oder ***data,*** entfernt. Es können mehrere Parameter angegeben werden, und die Reihenfolge wird bei der Bearbeitung der Inhalte berücksichtigt.

```
variable function remove [block] [first x] [last x]
```

Parameter

block	Entfernt das SCPI-Blockformat am Anfang des Inhalts wenn es vorhanden ist, z.B. #510200.
first <Anzahl>	Entfernt die Anzahl an Zeichen oder Bytes vom Anfang des Inhalts.
last <Anzahl>	Entfernt die Anzahl an Zeichen oder Bytes vom Ender des Inhalts.

Beispiel

```
data Screencopy
Screencopy function scpi read :HCOPy:DATA?
Screencopy function remove block last 1
```

6.2.11.4 Replace

Mit der Anweisung ***replace*** werden Inhalte von Variablen, mit dem Datentyp ***string*** oder ***data***, durch andere Werte ersetzt. Es werden zwei Parameter erwartet, der dritte optionale Parameter gibt die Anzahl der Ersetzungen an. Die Parameter werden mit einem Leerzeichen getrennt angegeben.

Der erste Parameter gibt den Inhalt an, der gesucht und ersetzt werden soll. Der zweite Parameter wird für die Ersetzung verwendet. Soll der gesuchte Inhalt gelöscht werden, muss eine leere Zeichenfolge angegeben werden. Der optionale dritte Parameter gibt die Anzahl der Ersetzungen an, ohne Angabe werden alle gefundenen Vorkommen des gesuchten Inhalts ersetzt.

Es können Zeichenfolgen oder Bytes in hexadezimaler Schreibweise, für den gesuchten und zu ersetzenden Inhalt angegeben werden. Die Zeichenfolgen müssen in Anführungszeichen angegeben werden. Bei Zeichenfolgen wird die Groß- und Kleinschreibung bei der Suche berücksichtigt.

Die hexadezimalen Werte müssen mit Komma getrennt, ohne Leerzeichen angegeben werden. Bei Variablen vom Datentyp ***string***, dürfen keine Werte mit Null ('00') angegeben werden.

```
variable function replace <Suche> <Ersetzung> [<Anzahl>]
```

Beispiel

```
string Datum
Datum set $date
print $Datum // Ausgabe: 2023-05-09
Datum function replace "-" ","
print $Datum // Ausgabe: 2023,05,09

... function replace "DAT2" "" 1 // Den ersten gefundenen Text löschen.
... function replace 0A,0D "" // Alle Zeichen für einen Zeilenumbruch (CR,LF)
löschen.
```

```
... function replace A5,D3,30,00 00,30,D3,A5 1 // Die ersten vier gefundenen
Bytes ersetzen.
... function replace "END" 0A,0D // Alle gefundenen Texte mit einen
Zeilenumbruch ersetzen.
```

6.2.11.5 Save

Mit der Anweisung ***save*** werden die Inhalte von Variablen in eine Datei gespeichert. Es kann ein Pfad und Dateiname für die Datei in Anführungszeichen angegen werden. Je nach Datentyp der Variable können weitere optionale Parameter verwendet werden. Der Pfad zur Datei und die Parameter können in beliebiger Reihenfolge angegeben werden.

Es können Platzhalter für den Dateinamen angegeben werden. Diese Platzhalter werden beim Speichern durch ihren aktuellen Wert ersetzt. Es können auch Variablen als Platzhalter verwendet werden.

Variablen vom Datentyp ***image*** werden im PNG-Format gespeichert. Die Erweiterung '.png' wird automatisch an den Dateinamen angefügt.

Vorhandene Dateien werden ohne Nachfrage überschrieben.

Wird nur der Dateiname ohne Pfad angegeben, wird die Datei in dem Verzeichnis gespeichert, in dem das ausführende Skript gespeichert wurde. Wurde das ausführende Skript noch nicht gespeichert, wird in das Verzeichnis gespeichert, von wo aus das Programm gestartet wurde.

Wird kein Dateiname angegeben, erscheint ein Speichern-Dialog für die Eingabe oder Auswahl eines Dateinamens. Wird der Dialog abgebrochen wird der Inhalt der Variable nicht gespeichert, und das Skript ohne Fehler weiter ausgeführt.

```
variable function save <Pfad> [add] [asc] [eol] [vcd]
```

Parameter

add	Die Daten werden an die vorhandene Datei am Ende hinzugefügt.
asc	Variablen vom Datentyp ***boolean***, ***integer*** und ***float*** werden als ASCII-Zeichenfolge gespeichert. Die Anweisung ***locale*** wird berücksichtigt. Ohne diesen Parameter werden sie als binäre Daten gespeichert.
eol	Am Ende der Daten wird ein Zeilenumbruch angefügt.

Parameter für waveform

vcd	Die Messkurvern werden im 'Value Change Dump' (VCD) Format gespeichert. Das VCD-Format wird z.B. von Sigrok PulseView verwendet.

Wichtig: Das VCD-Format kann zurzeit nur für die Logikkanäle verwendet werden.
Die anderen Kanäle wie Analog, Math usw. können überhaupt nicht gespeichert werden, diese Funktion ist noch nicht vorhanden.

Platzhalter für Dateipfad

/$...	Den Inhalt der angegebenen Variable verwenden.
/F	Herstellername des Messinstruments.
/O	Modellbezeichnung des Messinstruments.
/Y	Vierstellige Jahreszahl.
/M	Zweistellige Monatszahl.
/D	Zweistellige Tageszahl.
/H	Zweistellige Stundenzahl.
/N	Zweistellige Minutenzahl.
/S	Zweistellige Sekundenzahl.
/Z	Dreistellige Millisekundenzahl

Beispiel

```
string Zeit
Zeit set $time
Zeit function save "T:\Daten.txt" eol

float Pi
Pi set 3.14
locale on
Pi funcion save "T:\Daten.txt" add asc eol

integer Zähler
Zähler set 2
Zähler function save "T:\Daten Nr /$Zähler.dat"

image Bildschirmkopie
Bildschirmkopie function screencopy
Bildschirmkopie funcion save "T:\Bildschirmkopie /Y-/M-/D /H/N/S-/Z"

waveform Logikkanäle
Logikkanäle function waveform logic memory
Logikkanäle function save "T:\Logikkanäle /F /O" vcd
```

6.2.11.6 Scpi

Mit der Anweisung ***scpi*** werden SCPI-Kommandos an das Messinstrumente gesendet, und die Antwort des Instruments empfangen. Es gibt zwei Parameter ***read*** und ***write***. Mit ***write*** wird das angegebene SCPI-Kommando gesendet, eine Antwort vom Instrument wird nicht erwartet. Mit ***read*** wird das angegebene SCPI-Kommando gesendet, und es wird eine Antwort vom Instrument erwartet. Die empfangene Antwort kann einer Variable zugewiesen werden. Die Variable muss mit dem passenden Datentyp erstellt werden.

Die angegebenen SCPI-Kommandos werden ohne Änderung bis zu einem Kommentar oder bis zum Zeilenende verwendet.

Es können die Inhalte von Variablen gesendet werden, wenn der Variablennamen mit einem Dollarzeichen ***$*** angegeben wird. Die Werte der Variablen vom Datentyp ***boolean***, ***integer*** und ***float*** werden in eine Zeichenfolge konvertiert. Bei Fließkommazahlen wird das englische Format mit Punkt für den Dezimalpunkt verwendet.

Bei Variablen vom Datentyp ***data*** wird nur die zuletzt angegebene Variable verwendet. Die Daten werden an das Ende des SCPI-Kommandos angefügt. Alle anderen Datentypen werden durch eine leere Zeichenfolge ersetzt.

Wird im SCPI-Kommando das Zeichen für eine Konstante **#**, oder das Dollarzeichen ***$*** für eine Variable verwendet, wird geprüft ob eine Konstante oder Variable mit dem angegebenen Namen vorhanden ist, und der jeweilige Wert eingefügt. Wenn keine Konstante oder Variable vorhanden ist, werden die Zeichen unverändert übernommen.

```
variable function scpi read | write <SCPI-Kommando>
```

Parameter

read	Sendet Daten, und erwartet eine Antwort.
write	Sendet Daten, und erwartet keine Antwort.

Beispiel

```
function scpi write *RST // Das Instrument zurücksetzen.

string IDN
IDN function scpi read *IDN?
print IDN

integer Frequenz
Frequenz set 1000
function scpi write :HOR:FFT:CENT $Frequenz // Der Wert der Variable
'Frequenz' wird eingefügt.

data Einstellung
Einstellung set $file "T:\DS1000Z.bin"
Einstellung add $block
function scpi write :SYST:SET $setting // Die binären Daten werden am Ende
eingefügt.
```

6.2.11.7 Screencopy

Mit der Anweisung ***screencopy*** wird eine Bildschirmkopie von unterstützten Messinstrumenten eingelesen. Das Ergebnis der Anweisung muss einer Variable vom Datentyp ***image*** zugewiesen werden.

```
variable function screencopy
```

Unterstützte Messinstrumente

- R&S RTB2000 Oszilloskop
- Rigol DSA800 Spektrumanalyser
- Rigol DG4000 Funktionsgenerator
- Rigol DP800 Labornetzteil

Beispiel

```
image Bildschirmkopie
Bildschirmkopie function screencopy
```

6.2.11.8 Search

Mit der Anweisung ***search*** werden Messinstrumente im lokalen Netzwerk, und auch direkt mit dem PC verbundene Instrumente gesucht. Die gefundenen Instrumente müssen einer Variable vom Datentyp ***string*** zugewiesen werden. Die Variable kann mit der Anweisung ***view*** angezeigt werden. Die Anweisung ***print*** sollte nicht verwendet werden, da die Länge der Ausgabe beschränkt ist.

Mit den optionalen Parametern kann die Suche auf eine Schnittstelle begrenzt werden. Wird kein Parameter angegeben, werden alle unterstützten Schnittstellen für die Suche verwendet.

Die gefundenen Instrumente enthalten die IDN, und die Verbindungseinstellung für die Skript-Anweisung instrument

Hinweis: Die Suche im Netzwerk sendet einen UDP-Broadcast an den RPC-Port 111 für die LXI-Instrumente. Eine Firewall oder eventuell ein Virusschutz-Programm wird diesen Verbindungsversuch wahrscheinlich blockieren. Damit die Suche funktioniert muss eine Erlaubnis für das Programm bestätigt oder erstellt werden.

```
variable function search [usbtmc] [lxi] [rawtcp]
```

Parameter

usbtmc	Sucht nach lokal mit dem PC verbundene USB-TMC-Instrumente mit installierten WinUSB-Treiber.
lxi	Sucht im lokalen Netzwerk nach LXI-Instrumenten.
rawtcp	Sucht im lokalen Netzwerk nach RawTCP-Instrumenten mit Port 5025.

Beispiel

```
string Suche
Suche function search
Suche function view

Ausgabe:

IDN: Dreisiebner ; libusb1 ; 0001 ; 04-20171013
instrument $libusb1 usbtmc 1AB1 0849

IDN: Rohde&Schwarz ; RTB2004 ; 1333.1005k04/107362 ; 02.400
instrument $RTB2004 lxi 192.168.1.100 1024 192.168.1.102
```

6.2.11.9 View

Mit der Anweisung ***view*** wird der Inhalt einer Variable in einem Ansichtsfenster angezeigt. Für die verschiedenen Datentypen der Variable gibt es jeweils ein eigenes Ansichtsfenster. Wird als erster Parameter ein Name mit Dollarzeichen ***$*** angegeben, kann mit diesem Namen das Ansichtsfenster auch von anderen Skripten verwendet werden. Der gleiche Name kann für unterschiedliche Datentypen verwendet werden.
Wird kein Name angegeben, ist das jeweilige Ansichtsfenster nur für das aufrufende Skript verfügbar.

Mit dem Parameter ***close*** kann ein Ansichtsfenster geschlossen werden. Wird ein Name angegeben, wird nur dieses Fenster geschlossen. Wurde das benannte Ansichtsfenster von einem anderen Skript erstellt, wird es nicht geschlossen. Wird kein Name angegeben, wird das

Fenster mit dem gleichen Datentyp geschlossen.

Wird der Parameter ***closeall*** angegeben, werden alle Ansichtsfenster des Skript geschlossen. Ein Name oder Datentyp wird ignoriert.

Der Datentyp ergibt sich aus der angegebenen Variable. Die Datentypen ***boolean***, ***integer***, ***float*** und ***string*** werden gemeinsam in einem Ansichtsfenster für Text angezeigt. Die Datentypen ***data***, ***image*** und ***waveform*** werden jeweils in eigenen Ansichtsfenstern angezeigt für Daten, Bild und Messkurven.

Im Ansichtsfenster für Text wird standardmäßig der Text zu vorhandenen Text hinzugefügt, und am Ende ein Zeilenumbruch ausgeführt. Die Werte der angegebenen Parameter werden getrennt mit einem Tabulatorzeichen ausgegeben.

Im Ansichtsfenster für Daten wird der Inhalt immer ersetzt. Es gibt keine zusätzlichen Parameter.

[variable] function view [$Name] [close | closeall] [...]

Parameter für alle Ansichtsfenster

$Name	Optionaler Name für das Ansichtsfenster. Es muss der erste Parameter sein.
close	Schließt ein Ansichtsfenster mit dem angegebenen Namen oder Datentyp.
closeall	Schließt alle Ansichtsfenster des Skripts, unabhängig von Namen oder Datentyp.

Parameter für Text

set	Ersetzt den vorhandenen Text im Ansichtsfenster, anstatt ihn anzufügen.
eol	Fügt einen zusätzlichen Zeilenumbruch am Ende an.
neol	Fügt keinen Zeilenumbruch am Ende an. Ein ***eol*** wird ignoriert.
time	Gibt das Datum und die Uhrzeit an Anfang der Zeile aus.
debug	Gibt den Namen und Datentyp der Variable am Anfand der Zeile aus.

Beispiel

```
image Bildschirmkopie
Bildschirmkopie function screencopy
Bildschirmkopie function view // Ein Ansichtfenster für den Datentyp image
anzeigen.
Bildschirmkopie function view close // Das Ansichtsfenster mit dem Datentyp
image schließen.

Bildschirmkopie function view $Oszilloskop // Ein benanntes Ansichtsfenster
anzeigen.
function view $Bild close // Nur das benannte Ansichtsfenster schließen.

function view closeall // Alle Ansichtsfenster des Skripts schließen.

function view close // Es wird kein Ansichtsfenster geschlossen, es fehlt der
Name oder der Datentyp.
```

Hinweis: Die Ansicht des Datentyp ***waveform*** funktioniert noch nicht.

6.2.11.10 Wait

Mit der Anweisung ***wait*** wird die Ausführung des Skript angehalten. Es wird je nach Parameter auf eine Bestätigung für die weitere Ausführung gewartet. Eine vorgegebene Zeit lang gewartet, oder auf eine bestimmte Uhrzeit oder Datum.

Wird auf eine Bestätigung gewartet, muss für die weitere Ausführung das Skript mit 'Ausführen' wieder gestartet werden.

Mit dem Parameter ***msg*** kann eine Nachricht angegeben werden. Es können Konstanten und Variablen angegeben werden. Damit der Inhalt der Variablen ausgegeben wird, muss dem Variablenname ein Dollarzeichen ***$*** vorangestellt werden.

Soll eine Schleife mit Anweisungen möglichst immer in der gleichen Zeit ausgeführt werden, kann man den Parameter ***abs*** verwenden. Damit wird bei erneuten Ausführen der Anweisung, nur noch die restliche Zeit gewartet, die seit dem vorherigen Aufruf auf die angegebene Wartezeit noch fehlt.

```
function wait [abs] <Millisekunden> | msg <Nachricht> | time <Uhrzeit> | date
<Datum>
```

Parameter

<Millisekunden>	Wartezeit, mit Null wird auf eine Bestätigung gewartet.
abs <Millisekunden>	Absolute Wartezeit.
msg <Nachricht>	Nachricht mit warten auf Bestätigung.
time <Uhrzeit>	Wartet auf den Zeitpunkt. Format HH:NN:SS
date <Datum>	Wartet auf das Datum mit Uhrzeit. Format JJJJ-MM-TT HH:NN:SS

Beispiel

```
function wait 1000
function wait 0 // Wartet auf eine Bestätigung.

string Uhrzeit
Uhrzeit set $time
function wait msg Es wird seit $Uhrzeit gewartet.

function wait time 19:30:00 // Wartet solange bis die Uhrzeit übereinstimmt.
function wait date 2023-05-09 19:30:00 // Wartet auf das Datum und Uhrzeit.
```

Beispiel für abs

```
integer Zähler
label Schleife
function wait abs 1000 // Beim ersten Aufruf wird eine Sekunde gewartet,
function wait 700      // danach wird jedesmal nur die noch verbleibende Zeit
Zähler add 1           // auf die Sekunde gewartet.
Zähler equal 5         // In diesen Beispiel sind es dann ungefähr 300
Millisekunden.
print $Zähler
false goto Schleife
```

6.2.11.11 Waveform

Mit der Anweisung ***waveform*** werden Messkurven von unterstützten Messinstrumenten eingelesen. Das Ergebnis dieser Anweisung muss einer Variable vom Datentyp ***waveform*** zugewiesen werden.

Wenn kein Parameter angegeben wird, werden alle sichtbaren Kanäle eingelesen, ausgenommen z.B. die FFT-Ansicht bei einem Oszilloskop.

Die Parameter sind für verschiedene Instrumente benannt, können aber für jedes Instrument verwendet werden. Es ist gleich ob z.B. ***bus1*** oder ***pod1*** für einen Logikanschluss verwendet wird.

Unterstützte Messinstrumente

- R&S RTB2000 Oszilloskop

Parameter

screen	Daten aus dem Bildschirmspeicher einlesen.
memory	Daten aus dem Speicher einlesen.
allmemory	Daten aus dem gesamten Speicher einlesen.
8	Bittiefe der Daten für 8 Bit.
16	Bittiefe der Daten für mehr als 8 Bit.
all	Alle sichtbaren Kanäle einlesen.
analog	Alle sichtbaren analogen Kanäle einlesen.
logic	Alle sichtbaren digitalen Kanäle einlesen.
ch1, ch2, ch3, ch4	Auswahl eines analogen Kanals.
ma1, ma2, ... ma5	Auswahl eines mathematischen Kanals.
re1, re2, re3, re4	Auswahl eines Referenzkanals.
bus1, bus2	Auswahl eines Logikanschluss.
pod1, pod2	Auswahl eines Logikanschluss
d0, d1, ... d15	Auswahl eines digitalen Kanals.

Beispiel

```
instrument vxi 192.168.1.100
waveform Messkurven
Messkurven function waveform // Alle sichtbaren Kanäle einlesen.

Messkurven function waveform analog // Alle sichtbaren analogen Kanäle
einlesen.
Messkurven function waveform ch1 ch3 16 // Kanal 1 und 3 mit mehr als 8 Bit
einlesen.
Messkurven function waveform ch1 logic // Kanal 1 und alle sichtbaren
digitalen Kanäle einlesen.
Messkurven function waveform bus1 d12 // Alle sichtbaren digitalen Kanäle von
```

```
Bus1 oder Pod1 und D12 einlesen.
```

Hinweis zu R&S RTB2000 Oszilloskop

Es werden zu den analogen Kanaldaten, auch die Envelopedaten eingelesen wenn dieser Erfassungsmodus aktiviert ist. Die Envelopedaten sind immer 2 x 1.200 Punkte, so wie sie am Bildschirm dargestellt werden.

7 Geschichte

Die erste Version von dem Programm Messinstrumente wurde im September 2018 veröffentlicht. Es wurde nur die USB-TMC-Verbindung unterstützt.

Mit der zweiten Version konnten auch LXI-Verbindungen verwendet werden, und wurde im März 2020 veröffentlicht. Es wurde später noch mit den COM- und RawTCP-Verbindungen erweitert.

Die Verwendung von Skripten wurde mit der dritten Version möglich, die November 2021 veröffentlicht wurde.

In der vierten Version vom August 2022, wurde die integrierte Unterstützung von einigen Messinstrumente entfernt, da damit nicht mehr getestet werden konnte.

Die Entwicklung und Bedienung der Programmoberfläche war nicht gut, und es wurde schwieriger neue Funktionen zu integrieren. Mit dem aktuellen Programm Messinstrumente-Skript, wurde darum der Schwerpunkt auf die Skripte gelegt. Die Steuerung der Instrumente erfolgt nur über Skripte, die eingelesenen Daten können aber in Fenstern dargestellt werden. Diese fünfte Version sollte die letzte sein mit großen Änderungen, und nur mehr um neue Funktionen erweitert werden. Veröffentlich wurde die Version am 11. Mai 2023.

Änderungen

2023-06-01 - Version 0.5.4.1

- Geändert: Die Anweisung ***function save*** kann auch ohne Angabe eines Dateinamen für das Speichern verwendet werden, es wird dann ein Speichern-Dialog angezeigt.

2023-05-24 - Version 0.5.4.0

- Neu: Mit der Anweisung ***return*** kann zu der zuletzt ausgeführten ***goto***-Anweisung zurückgesprungen werden.

2023-05-22 - Version 0.5.3.0

- Neu: Die Anweisung ***function view***, für die Ansicht von Variablen mit dem Datentyp ***data*** ist jetzt vorhanden. Siehe Ansichtsfenster Daten.
- Neu: Mit der Anweisung ***function search*** können Instrumente gesucht, und die Verbindungseinstellungen angezeigt werden.

2023-05-20 - Version 0.5.1.3

- Neu: Die Anweisung ***function view***, für die Ansicht von Variablen mit dem Datentyp ***boolean***, ***integer***, ***float*** und ***string***, ist jetzt vorhanden. Siehe Ansichtsfenster Text.

2023-05-18 - Version 0.5.1.2

- Neu: Die Anweisung ***function view***, für die Ansicht von Variablen mit dem Datentyp ***image***, ist jetzt vorhanden.

- Neu: Variablen mit Datentyp ***image*** können Variablen mit Datentyp ***data*** zugewiesen werden, wenn die Daten ein gültiges Bildformat enthalten.

2023-05-11 - Version 0.5.0.0

- Neu: Messinstrumente-Skript ist die Nachfolgeversion von Messinstrumente v0.4.0.18.
- Neu: Der Skript-Editor wurde neu programmiert.
- Neu: Die Hilfe wurde integriert.

8 Kontakt

Autor:	Peter Dreisiebner
E-Mail:	web@dreisiebner.at
Webseite:	https://www.dreisiebner.at