Winguard Softwareanleitung

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Was ist Winguard

Winguard ist eine Testumgebung mit integrierter Entwicklungsumgebung zum Erstellen von Prüfabläufen.

Auf einfache Weise erstellen Sie per Drag-and-Drop und dem Ausfüllen von Formularen einen Prüfablauf mit den Befehlszeilen zur Steuerung der Hardware und zur Erfassung und Auswertung der Ergebnisse.

Programmierkenntnisse sind dafür nicht erforderlich. Mit Winguard können Sie sowohl das Guardian-Testsystem steuern, als auch Geräte die über RS-232, Canbus oder TCP/IP an den PC angeschlossen sind. Mit dem integrierten Debugger und der Laborfunktion mit virtuellen Bedienelementen verfügen Sie über komfortable Tools, die Sie bei der Inbetriebnahme und Fehlersuche Ihrer Prüfvorrichtung unterstützen.

1.2 Was ist ein Guardian-Testsystem

Das Guardian-Testsystem ist ein universell einsetzbarer Funktionstester. Es besitzt eine 288-polige Schnittstelle zum Andocken von Prüfadaptern und wird von einem Windows-PC über Winguard gesteuert. Zur Einbindung in eine eigene Software steht eine Treiber-DLL zur Verfügung.

Abbildung 1.1: Guardian Testsystem

Das System eignet sich für folgende Anwendungen:

- Funktionsprüfung von elektronischen Baugruppen (Analog, Digital, CPU-Programmierung, Leistungselektronik, Automotive, u.v.m)
- Incircuit-Test, selbstlernender Kurzschluss- und Unterbrechungstest, Messungen von Widerständen, Diodenstrecken, Kondensatoren
- Endprüfung von elektronischen Geräten
  - Stimulation von Eingängen, Prüfung von Ausgängen
  - Kommunikation über digitale Schnittstellen
  - Parametrieren und Kalibrieren
  - Drucken und Archivieren von Messprotokollen
  - Grafische Bedienerführung
  - und vieles mehr ...
- Lebensdauertest, Typprüfung, Klimatest, Burn-In-Test
- Einsetzbar als manueller Prüfplatz oder innerhalb einer Fertigungslinie

1.3 Konventionen in dieser Hilfe

Menüpunkte, Schaltflächen, Text auf dem Bildschirm
Projekt/Öffnen, Start

Tastenkürzel
Strg+O, F9

Beispielprogramme der Skriptsprache

```
TextOut(Text:”Hallo Welt”);
```

1.4 Installation

Winguard wird als Windows-Installer-Paket ausgeliefert und kann für alle Benutzer des Rechners oder nur für den angemeldeten Benutzer installiert werden. Außerdem sind einige Funktionen als Plugins implementiert und können bei der Installation übersprungen werden. Für fast alle Anwendungszwecke sind aber keine Änderungen der Installationsoptionen nötig.

Winguard läuft auf Windows ab Windows 10. Zur Ansteuerung eines Guardian-Systems wird eine RS-232-Schnittstelle benötigt.

Anmerkung
Bei Verwendung eines USB/RS-232 Wandlers reduziert sich die Testgeschwindigkeit des Systems deutlich. Verwenden Sie daher möglichst einen Comport des Mainboards oder einer PCI Steckkarte.

1.5 Installation einer Lizenz

Nach der Installation startet Winguard automatisch im *Demo-Modus*. Dieser Modus ist für Anwender gedacht, die Winguard zuerst ausprobieren möchten. Der Demo-Modus besitzt keine Einschränkungen zu einer Vollversion, jedoch müssen Sie für eine gewerbliche Nutzung eine Lizenz erwerben.

Mit dem Erwerb einer Lizenz erhalten Sie eine *Lizenzdatei*. Diese Datei müssen Sie Winguard bekannt machen.

Öffnen des Lizenzdialogs
Starten Sie zuerst Winguard. Wählen Sie im Hilfemenü den Punkt Lizenzen.

Import der Lizenz
In dem sich öffnenden Lizenzdialog finden Sie eine Übersicht über alle aktivierten Lizenzen. Um eine Lizenz hinzuzufügen klicken Sie auf Lizenz importieren und wählen Sie die entsprechende Lizenzdatei aus. Die neue Lizenz sollte jetzt im Lizenzdialog erscheinen. Sie können den Lizenzdialog nun schliessen. Sollte der Import einer Lizenz fehlschlagen, wenden Sie sich bitte an unseren Support.

1.6 Inbetriebnahme des Guardian-Systems

Verbinden Sie den Guardian mit dem PC
Verbinden Sie die RS-232-Schnittstelle *System*(A) auf der Rückseite des Guardians mit einer RS-232-Schnittstelle des PC und schalten Sie das Testsystem ein.

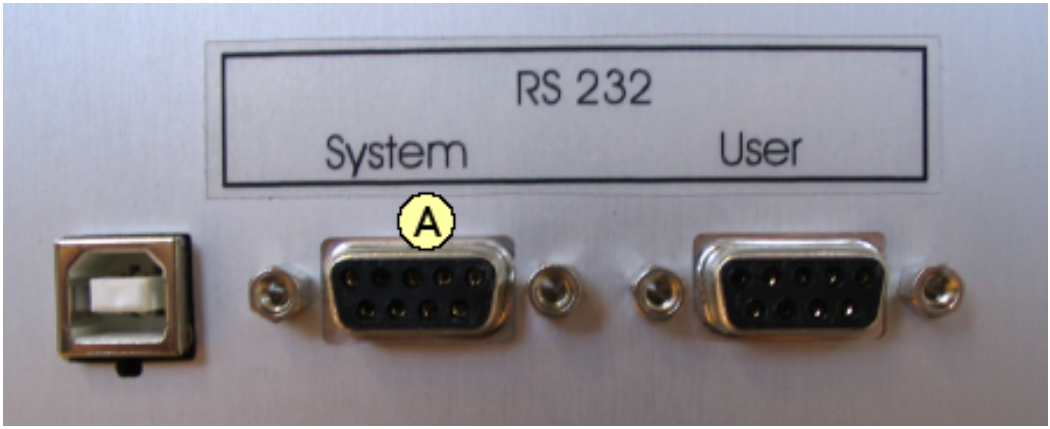


Abbildung 1.2: Rückseite des Guardian Testsystems

Starten Sie Winguard und öffnen Sie den Einstellungsdialog
Starten Sie Winguard und wechseln Sie über den Menüpunkt Ansicht/Editor in den Editor. Öffnen Sie den Einstellungsdialog im Menü Optionen/Einstellungen.

Starten Sie die Hardware-Autoerkennung
Geben Sie unter Schnittstelle die serielle Schnittstelle an, an der das

Guardian-Testsystem angeschlossen ist. Starten Sie dann die Hardwareerkennung durch Klick auf Autoerkennung starten.

Abschliessen der Autoerkennung

Nach der Autoerkennung werden die im Guardian installierten Karten in einer Tabelle aufgelistet. Klicken Sie Übernehmen um die gefundene Hardware zu speichern.

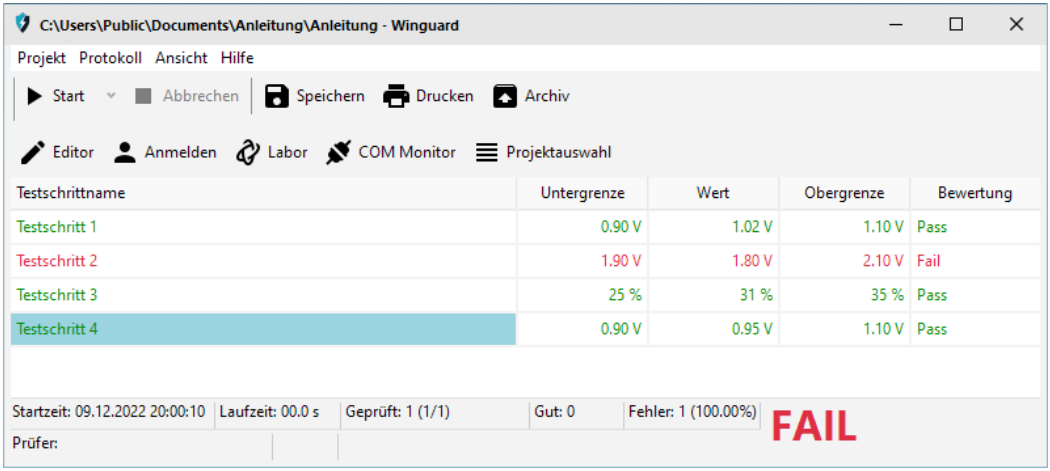
Einstellung der Kalibrierung

Wenn Sie zu Ihrem Guardian Kalibrierdateien erhalten haben, müssen Sie diese in Winguard laden. Wählen Sie dazu den Reiter Kalibrierung aus und tragen Sie die Pfad in den Feldern Einzel-Kalibrierdateien ein.

Kapitel 2

Die Testumgebung

Die Testumgebung ist die Steuerzentrale der Prüfsoftware. Hier können Sie Prüfprogramme laden und starten, hier werden die Messergebnisse der einzelnen Testschritte angezeigt. In diesem Kapitel finden Sie eine Übersicht über die verschiedenen Funktionen der Testumgebung.



2.1 Erste Schritte in der Testumgebung

2.1.1 Laden und Starten eines Prüfprogramms

Laden eines Prüfprogramms

Wählen Sie im Hauptmenü den Punkt Projekt/Öffnen oder benutzen Sie die Tastenkombination Strg+O um ein Programm auszuwählen. Zur schnelleren Auswahl häufig benutzter Programme speichert Winguard die letzten 10 geladenen Programme im Menü Projekt (A). Den Pfad des aktuell geladenen Programmes finden Sie in der Titelzeile des Fensters.

Starten und Beenden des Programms

Mit den Knöpfen links oben im Hauptfensters können Sie das aktuell geladene Prüfprogramm starten oder das laufende Prüfprogramm beenden. Alternativ können Sie zum Starten auch die Taste F9 und zum Beenden die Taste Esc verwenden.

Fehlerbehandlung

Tritt während des Programmablaufs ein Fehler auf erscheint ein Fehlerdialog. Dieser enthält unter anderem eine Beschreibung des Fehlers. Falls möglich, sollten Sie versuchen den Fehler zu beheben. Klicken Sie in diesem Fall auf Wiederholen um den Prüfablauf fortzusetzen. Können Sie den Fehler nicht beheben, haben Sie die Wahl den Fehler entweder zu ignorieren oder den Prüfablauf abzubrechen.

2.1.2 Messprotokoll und Prüfablauf

Während eines Prüfablaufs werden nacheinander verschiedene Testschritte durchgeführt. Die Ergebnisse der Tests werden im Messprotokoll aufgeführt. Ein fehlgeschlagener Test wird rot(A) markiert, ein erfolgreicher grün(B). Die Bewertung wird zusätzlich in der Spalte Bewertung des Messprotokolls angezeigt.

Sehr viele Tests vergleichen einen Messwert mit zwei Grenzwerten. Liegt der Messwert innerhalb der Grenzen gilt der Test als bestanden, ansonsten als fehlgeschlagen. Das Messprotokoll zeigt in diesem Fall die Untergrenze, den Messwert(Wert) und die Obergrenze in den entsprechenden Spalten an.

Anmerkung

Mit Ansicht/Fehlerquote und Ansicht/Testschrittzeit können zwei zusätzliche Spalten eingeblendet werden die zu jedem Testschritt den prozentualen Fehler bzw. die benötigte Zeit anzeigen.

2.1.3 Statusinformationen

Unten im Fenster finden Sie Statusinformationen zu dem aktuellen Prüfprogramm.

Sie sehen die Laufzeit der letzten Prüfung, die Anzahl der bereits durchgeführten Prüfungen, die Anzahl der mit den Bewertungen „Pass“ und „Fail“ durchgeführten Prüfungen sowie die daraus resultierende Fehlerquote.

Diese Zähler werden beim Laden eines Projektes und beim Neustart von Winguard auf Null zurückgesetzt.

2.1.4 Prüfer anmelden

Über Anmelden kann ein Prüfer angemeldet werden. Dabei wird der Name des Prüfers abgefragt. Dieser Name erscheint nach der Anmeldung in der Statusleiste und in allen erzeugten Prüfprotokollen. Soll der angemeldete Prüfer geändert werden, klicken Sie erneut auf Anmelden.

Optional kann der angegebene Prüfername gegen eine Liste von gültigen Namen verglichen werden. Eine Anmeldung ist dann nur mit einem gültigen Namen möglich.

Die Einstellungen zur Prüferanmeldung können Sie über den Einstellungsdialog im Reiter Programm ändern (Achtung, Sie benötigen Zugriff auf die Entwicklungsumgebung). Um beim Programmstart automatisch eine Prüferanmeldung zu erfragen muss die Option Prüfer beim Programmstart anmelden aktiviert werden. Wenn die Option Prüferliste in Datenbank verwendet wird, ist der Login nur mit in der Datenbank registrierten Prüfernamen möglich. Eine Liste dieser Namen können Sie über den Button Prüferliste editieren einsehen und verändern.

Siehe auch Einstellungen zur Prüfer-Anmeldung.

2.2 Das Messprotokoll

In das Messprotokoll werden während des Testlaufs die Ergebnisse der einzelnen Testschritte sowie weitere Informationen zum Testlauf eingetragen. Das Messprotokoll wird in der Tabelle der Testumgebung dargestellt und während des Ablaufs aktualisiert. Nach dem Testlauf können Sie das Protokoll speichern oder auf einem Drucker ausgeben. Haben Sie Zugriff auf die Entwicklungsumgebung so können Sie außerdem die einzelnen Testschritte editieren.

2.2.1 Anzeigeoptionen des Messprotokolls

Über das Menü Ansicht können Sie die Anzeige des Messprotokolls beeinflussen. Mit Nur fehlerhafte Bewertungen wählen Sie aus, ob alle Testschritte angezeigt werden sollen oder nur die fehlerhaften. Über die Menüpunkte Fehlerquote und Testschrittzeit können Sie zusätzliche Spalten im Protokoll einblenden.

Die Spalte Fehlerquote zeigt für einen Ergebnisbefehl an, wie oft dieser bisher fehlgeschlagen ist. Zur Berechnung werden alle Auswertungen seit dem Laden des Programmes herangezogen. Die Spalte Zeit zeigt für einen Testschritt die Zeit an, die seit dem letzten Testschritt vergangen ist.

2.2.2 Sichern des Messprotokolls

Das Messprotokoll kann zur späteren Verwendung in der Systemdatenbank gesichert werden. Winguard kann das Messprotokoll automatisch speichern wenn ein Test beendet ist oder beim Drucken des Protokolls. Sie können das Protokoll auch händisch nach Abschluss des Tests oder über einen Befehl während des Tests speichern.

Wann werden Protokolle gespeichert

In der Testumgebung können Sie im Menüpunkt Protokoll/Speicherzeitpunkt den Zeitpunkt einstellen, zu dem das Protokoll gesichert werden soll:

Manuell / Script
: Das Protokoll wird nicht automatisch gespeichert

Nach Erstellung
: Das Protokoll wird am Ende des Tests gespeichert

Beim Drucken
: Das Protokoll wird gespeichert wenn es gedruckt wird

Nie
: Das Protokoll wird selbst auf Scriptbefehl hin nicht gespeichert

Wie sichere ich das Protokoll manuell

Um das Protokoll des letzten Tests zu speichern, klicken Sie auf Speichern oder verwenden Sie die Tastenkombination Strg+S. Das Protokoll wird dann in der Systemdatenbank gesichert.

Eine weitere Möglichkeit besteht darin, das Protokoll aus dem laufenden Prüfprogramm heraus zu sichern. Wenn Sie den Befehl Report speichern aufrufen, wird die aktuelle Fassung des Protokolls in der Datenbank gespeichert.

Was wird im Protokoll gespeichert
Zu jedem Test werden die einzelnen Testschritte mit Messwerten und Messwertgrenzen sowie folgende Daten gesichert:

- Das Prüfprogramm
- Mit welchem Ergebnis die Prüfung abgeschlossen wurde (Pass, Fail oder Abgebrochen)
- Der Zeitpunkt zu dem die Prüfung beendet wurde und die Laufzeit
- Der Inhalt der Systemvariablen `Seriennummer`
- Der Inhalt der Systemvariablen `ProduktID`
- Der Inhalt der Variablen `Auftrag` und `ProduktID2`, wenn vorhanden
- Der angemeldete Prüfer
- Der eingestellte Prüfplatzname

2.2.3 Drucken des Messprotokolls

Um ein Protokoll bei Bedarf drucken zu lassen verwenden Sie die Tastenkombination Strg+P oder klicken Sie auf Drucken.

Sie können das aktuelle Protokoll auch aus dem Test heraus drucken indem Sie den Befehl Report drucken in Ihr Prüfprogramm einbauen.

Sie können die Druckereinstellungen mit dem Menüpunkt Protokoll/Druckeinrichtung... ändern.

2.2.3.1 Das Aussehen des Ausdruckes anpassen

Mit Protokoll/Gestaltung... wird ein Einstellungsdialog für den Protokolldruck geöffnet:

Druckkopf
Hier können Sie die Texte angeben, die vor dem eigentlichen Protokoll ausgegeben werden. Außerdem kann gewählt werden, welche Zeilen mit Informationen über die Prüfung ausgegeben werden sollen. Beispielsweise kann hier der Prüfplatz abgewählt werden, wenn es nur einen Prüfplatz gibt.

Druckkopf-Logo
Ein Bild, dass oben rechts auf dem Protokoll ausgegeben wird.

Farbdruck
Das Protokoll wird farbig gedruckt. Dabei werden die Testschritte abhängig von ihrem Ergebnis rot oder grün gedruckt.

Druckkopf-Schrift, Messwert-Schrift
Hier können Sie die für den Druck verwendeten Schriftarten angeben.

2.2.4 Editieren der Testschritte

Wenn der Zugriff auf die Entwicklungsumgebung freigeschaltet ist, können die Testschritte des Programms auch aus der Testumgebung heraus editiert werden. Dazu klicken Sie im Messprotokoll auf den entsprechenden Testschritt doppelt. Es öffnet sich das zugehörige Ergebnis-Modul in dem Sie die gewünschten Änderungen vornehmen können.

Siehe auch Einstellungen des Ergebnismoduls.

2.2.5 Archiv

2.2.5.1 Anzeige alter Protokolle

Zuvor gespeicherte Protokolle können Sie aus dem Archiv anzeigen lassen. Wenn aktiv, wird ein zuvor gespeichertes Protokoll statt dem der letzten Prüfung angezeigt. Die Kontrollen zur Auswahl der Protokolle finden Sie am unteren Rand der Testumgebung.

Während die Anzeige der gespeicherten Protokolle aktiv ist, können Sie über die Buttons Vor bzw. Zurück durch die Liste der Protokolle blättern. Dabei zeigt Ihnen das Feld Protokoll an, wieviele Protokolle in der Datenbank enthalten sind und welches Protokoll Sie gerade sehen.

Wenn Sie sehr viele Messprotokolle gespeichert haben, kann es mühsam sein ein bestimmtes Protokoll zu finden. Sie können die Anzeige der Protokolle über den Protokollfilter einschränken. Klicken Sie dazu auf Filter:. Es öffnet sich ein Dialog mit Filtereinstellungen:

Prüfprogramm
Möchten Sie nur Protokolle zu einem bestimmten Programm anzeigen lassen, wählen Sie hier dieses Programm aus.

Bewertungen
Mit dieser Option geben Sie an, mit welchem Ergebnis der Test abgeschlossen wurde.

Zeitraum
Hier geben Sie an, in welchem Zeitraum die Messung erfolgte.

Prüflingsinformationen
Wenn Sie nur Protokolle zu bestimmten Prüflingen sehen möchten, tragen Sie hier die Seriennummer/ Prüflings-ID ein. Sie können hier „%“ als Platzhalter für beliebigen Text und „_“ als Platzhalter für ein beliebiges Zeichen verwenden.

Prüfer
Über diese Option können Sie nur Protokolle für einen bestimmten Prüfer anzeigen lassen.

Prüfplatz
Über diese Option können Sie nur Protokolle für einen bestimmten Prüfplatz anzeigen lassen.

Dauer
Sind die Werte ungleich 0, werden sie mit den Testlaufzeiten verglichen.

Alle Einstellungen sind optional. Wird kein Filter angegeben werden alle in der Datenbank gespeicherten Protokolle gezeigt.

Eine kurze Zusammenfassung des eingestellten Filters sehen Sie im Hauptfenster.

2.2.5.2 Löschen alter Protokolle

Mit Protokoll/Löschen und Protokoll/Alle Löschen können das momentan ausgewählte beziehungsweise alle mit dem Protokollfilter ausgewählten Protokolle gelöscht werden. Dies lässt sich nur durch Wiederherstellen eines Datenbank-Backups rückgängig machen.

2.2.6 Datenbankschema

Die Protokolle werden in verschiedenen Tabellen einer SQL-Datenbank abgelegt. Winguard bietet Plugins um den Zugriff auf verschiedene Datenbanksysteme zu gewährleisten. In der Standardinstallation wird die Datenbank im *SQlite*-Format abgelegt. Die Datenbankdatei wird dabei im Anwendungsverzeichnis des angemeldeten Benutzers gespeichert (`%APPDATA%\Winguard\default.db`).

Unabhängig von der Art der verwendeten Datenbank werden drei Tabellen angelegt. Diese enthalten die folgenden Daten:

progs
Alle verwendeten Prüfprogramme.

progs.Id
Eindeutige Nummer des Prüfprogramms.

progs.name
Datei-Pfad des Prüfprogramms.

tests
Die Protokolle.

tests.Id
Eindeutige Nummer des Protokolls.

tests.prog
Nummer des Prüfprogramms welches das Protokoll erzeugt hat. (prods.Id)

tests.finish
Zeitpunkt, an dem das Protokoll gespeichert wurde.

tests.duration
Dauer des Prüfablaufes.

tests.valuation
Bewertung: 0 für Pass, 1 für Fail, 2 für Abgebrochen.

tests.tester
Name des Prüfers.

tests.serial
Inhalt der Variable `Seriennummer`.

tests.ident
Inhalt der Variable `ProduktID`.

tests.ident2
Inhalt der Variable `ProduktID2`.

tests.batch
Inhalt der Variable `Auftrag`.

tests.stationid
Name des Prüfplatzes.

teststeps
Die Zeilen der Protokolle.

teststeps.Id
Eindeutige Nummer der Protokollzeile.

teststeps.etype
Typ: 0 für Text, 1 für Ergebnis, 2 für Ergebnis in Hexadezimal

teststeps.name
Text der Zeile.

teststeps.min
Untergrenze.

teststeps.max
Obergrenze.

teststeps.value
Wert.

teststeps.rating
Bewertung: 0 für Pass, 1 für Fail

teststeps.info
Schriftart.

teststeps.err_rate
Fehlerrate dieser Zeile seit dem letzten Start von Winguard.

teststeps.sheet
Nummer der Script-Unit, die die Protokollzeile produziert hat.

teststeps.row
Zeile in der Script-Unit, die die Protokollzeile produziert hat.

teststeps.testid
Nummer des Protokolls. (tests.Id)

teststeps.unit
Einheit des Wertes.

teststeps.precision
Anzahl der angezeigten Nachkommastellen.

Siehe auch Wie ändere ich die Einstellungen zur Systemdatenbank?.

2.3 Projektauswahl beim Programmstart

Mit der Projektauswahl können wichtige Projekte in einer Liste zusammengefasst und vom Bediener schneller geladen werden. Dazu reicht die Eingabe eines Projektschlüssels (zum Beispiel mit einem Barcode-Scanner). Das ausgewählte Projekt wird automatisch geladen und gestartet. Der Auswahldialog kann direkt nach dem Start angezeigt werden und öffnet sich nach jedem Testdurchlauf erneut. Falls gewünscht können vom Bediener noch Parameter abgefragt werden.

2.3.1 Bedienung der Projektauswahl

Wurde die Projektauswahl geöffnet, wird dem Bediener eine Liste der verfügbaren Projekte gezeigt. Die Einträge der Liste zeigen den *Projektschlüssel* gefolgt vom Pfad der Projektdatei in eckigen Klammern. Der Projektschlüssel muss eindeutig sein, die Projektdatei hingegen kann beliebig oft verwendet werden.

Um ein Projekt zu laden wird in das obere Eingabefeld der Projektschlüssel eingegeben und mit der Eingabetaste bestätigt.

Verlangt das ausgewählte Projekt zusätzliche Parameter, werden entsprechende Eingabefelder eingeblendet. Der Bediener gibt auch hier die nötigen Informationen ein und bestätigt jede Eingabe mit der Eingabetaste. Nach Eingabe der Parameter wird das Projekt geladen und gestartet.

2.3.2 Aktivierung der Projektauswahl

Winguard erlaubt das Speichern verschiedener Konfigurationen in separaten *Projektauswahldateien* mit der Endung `.wgps`. Ist in den Einstellungen die Option `Aktuelles Projekt beim Programmstart öffnen` aktiviert und ist das letzte geöffnete Projekt eine Projektauswahldatei, so wird diese Projektauswahl beim Start angezeigt. Winguard kann auch mit einer bestimmten Projektauswahl gestartet werden, indem die Auswahldatei als Parameter übergeben wird.

2.3.3 Konfiguration der Projektauswahl

Die Konfigurationsdateien der Projektauswahl können mit einem beliebigen Texteditor editiert werden. Eine Konfigurationsdatei enthält zu jedem angezeigten Projekt einen *Block*. Ein Block beginnt mit dem Projektschlüssel in eckigen Klammern und endet wenn ein neuer Block beginnt oder das Ende der Datei erreicht wird. Jeder Block besteht aus mehreren Feldern die einem *Feldnamen* einen Wert zuweisen. Dabei sind Feldnamen und -werte durch = voneinander getrennt.

```
[ART01]
ProjectPath=Anleitung.wgp

[ART02]
ProjectPath=Anleitung.wgp
Foo=Bar

[ART05]
ProjectPath=Anleitung.wgp
```

```
Foo=?

[ART07]
ProjectPath=Anleitung.wgp
Foo=?ÜbergabeParameter
```

Die Konfiguration zu obigem Beispiel enthält vier Blöcke `[ART01]` bis `[ART07]`. Wir werden die einzelnen Blöcke im Folgenden erklären.

2.3.3.1 Ein einfaches Beispiel

Der erste Block erzeugt einen Eintrag im Auswahldialog ohne weitere Informationen. Der Projektschlüssel *ART01* wird am Anfang des Blocks in eckigen Klammern angegeben. Wird dieses Projekt vom Bediener ausgewählt, wird die unter *ProjectPath* angegebene Datei geladen und gestartet.

2.3.3.2 Übergabe von Parametern

Mit Ausnahme von *ProjectPath* werden alle Felder als Variablen vom Typ *String* in das geladene Programm übernommen. Der Wert im Programm entspricht dem Wert im geladenen Block. Dies ist zum Beispiel hilfreich um verschiedene Varianten eines Prüflings zu unterscheiden. Wird der zweite Beispieleintrag geladen, wird eine Variable *Foo* mit dem Wert *„Bar“* initialisiert.

2.3.3.3 Benutzereingabe von Parametern

Mitunter kann es sinnvoll sein, vom Benutzer bestimmte Daten zu erfragen. Wird ein Feld statt mit einem Wert mit einem Fragezeichen belegt, muss der Bediener den Wert im Projektauswahldialog eintragen. Das Eingabefeld wird mit dem Namen der Variablen beschriftet.

2.3.3.4 Alternative Beschriftung für Benutzereingaben

Um dem Bediener bei der Eingabe von Parametern mehr Informationen zu geben, kann hinter dem Fragezeichen optional eine Beschriftung angegeben werden. Diese wird dann im Auswahldialog anstatt des Parameternamens verwendet.

Anmerkung
Auf die vom Auswahldialog angelegten Variablen kann nicht zugegriffen werden wenn das Projekt über den Menüpunkt Laden geladen wird. Soll das Projekt also auch ohne den Auswahldialog benutzbar sein, müssen diese Variablen explizit angelegt werden. Achtung: die Variablendefinition im Programm überschreiben die Vorgaben aus dem Auswahldialog. Normalerweise sind Variablen aus dem Auswahldialog mit dem Flag nicht zurücksetzen markiert und haben keinen Standardwert. Wird die Variable im Programm anders definiert, kann dies zu unvorhergesehenen Verhalten bei der Verwendung des Projektauswahldialogs führen.

Kapitel 3

Werkzeuge

3.1 Das Guardianlabor

Das Guardianlabor erlaubt Ihnen die direkte Ansteuerung eines Guardian-Testsystems ohne dass ein Prüfprogramm geschrieben werden muss. Sie können das Labor mit Ansicht/Guardian Labor oder Strg+L starten.

Das Labor besitzt zu jeder Guardiankomponente ein Steuerelement.

3.1.1 PSU-Steuerung

Das Labor verfügt über ein Steuerelement zu jeder Guardianstromversorgung. In der Statusanzeige oben wird die gemessene Spannung und Stromstärke angezeigt.

Die Schaltflächen erlauben Ihnen die Auswahl zwischen Gleich- (⎓) und Wechselstrom (∼) und das Ein- bzw. Ausschalten der Versorgung (⏻).

Um die Sollwerte für die Stromausgabe einzustellen benutzen Sie die Eingabefelder. Die Stromstärke ist dabei ein Maximalwert.

3.1.2 ADX-Steuerung

Die ADX-Karte ist das interne Digitalmultimeter des Guardiansystems. In der Statusanzeige sehen Sie die eingestellte Messart, den Messbereich sowie den aktuellen Messwert, sofern die Messung gestartet wurde. Über die Schaltflächen können Sie die Messart der Karte einstellen.

Unterstützt werden Gleichspannungs- (V⎓), Wechselspannungs- (V~), Widerstands- (Ω) und Diodenspannungsmessung (⊣). Über die Schalter `T+` und `T-` können Sie die Messzeit erhöhen oder verringern. Eine höhere Messzeit führt zu einem ruhigerem Messwert. Darunter können Sie den Messbereich zu der jeweiligen Messart einstellen, die Schaltfläche unten rechts (⏻) startet beziehungsweise beendet die Messung.

Während die Messung läuft wird permanent die ADX-Karte des Guardiansystems abgefragt und der aktuelle Messwert angezeigt.

Die Messung erfolgt mit 10 MΩ Abschlusswiderstand und ohne Entladung oder AC-Kopplung.

3.1.3 Messstellenumschalter

Die MSU-Steuerung erlaubt das Verbinden zweier Messkanäle. Die Eingabe der Kanäle erfolgt über die Kanalnummer. Sie können die Kanalnummer über die Schaltflächen `+` und `-` einstellen oder im Eingabefeld eintippen. Die Schaltfläche ↕ vertauscht die ausgewählten Kanäle, was einer Umpolung gleichkommt.

Die Schaltfläche 🔍 öffnet den *Pinfinder*. Der Pinfinder kann zum Lokalisieren von Messkanälen verwendet werden. Verbinden Sie eine Messspitze mit dem positiven Messeingang der ADX-Karte und starten Sie den Pinfinder über ⏻. Berühren Sie nun mit der Messspitze einen Kontakt im Prüfadapter, werden Ihnen alle verbundenen Messkanäle angezeigt.

3.1.4 Relaiskarten

Zu jeder Relaiskarte kann im Labor eine Relaissteuerung aktiviert werden. Diese zeigt den Relaisstatus an und erlaubt es die einzelnen Relais zu schalten. Dazu gibt es zu jedem Relais einen Schalter. Ein Relais aktivieren Sie, indem Sie die rechte Hälfte des Schalters klicken. Um ein aktiviertes Relais abzuschalten klicken Sie die linke Hälfte.

3.2 Der COM Monitor

Der COM Monitor loggt alle Zugriffe auf die seriellen Schnittstellen über Winguard. Wählen Sie im Menü `Ansicht/COM Monitor`.

Wenn der COM Monitor angezeigt wird, werden im Textfeld des COM-Monitors die COM-Port-Zugriffe des Prüfprogramms mitgeschrieben. Durch Klicken der Schaltflächen oben können Sie das Log drucken, speichern oder löschen. Außerdem können Zeitstempel und Hexadezimaldarstellung gewählt werden.

Eine Zeile hat das Format `COMX: BEFEHL PARAMETER`

Dabei enthält `COMX` den Namen des COM-Ports auf den der Zugriff erfolgte, zum Beispiel `COM4`.

Ist der Befehl `:<` handelt es sich um eine Leseoperation. Der COM Monitor zeigt dann die gelesenen Daten einmal als Text und einmal die ASCII-Zeichen in Hexadezimaldarstellung.

Ist der Befehl `>:`, hat Winguard Daten über den COM-Port gesendet. In diesem Fall enthält `PARAMETER` die gesendeten Daten als Bytearray.

Der Befehl `OPEN` zeigt an, dass der COM-Port geöffnet wurde. `PARAMETER` enthält in diesem Fall eine Liste von Parametern durch Schrägstriche voneinander getrennt. Die einzelnen Felder beschreiben:

- `B` Die Baudrate
- `F` FlowControl (None, RtsCts, XonXoff)
- `P` Parität (None, Odd, Even, Mark, Space)
- `S` Die Anzahl der Stopbits
- `D` Die Anzahl der Datenbits
- `DTR` Der Wert auf den DTR gesetzt wurde
- `RTS` Der Wert auf den RTS gesetzt wurde
- `Break` Der Wert auf den BREAK gesetzt wurde

Der Befehl `CLOSE` zeigt an, dass der COM-Port geschlossen wurde. Für diesen Befehl werden keine Parameter angegeben.

Kapitel 4

Einstellungen

Dieses Kapitel enthält eine Auflistung und Erklärung sämtlicher Konfigurationsschalter die über den Einstellungsdialog geändert werden können. Um den Einstellungsdialog zu öffnen, wechseln Sie in den Editor und klicken im Menü Optionen den Punkt Einstellungen.

Die Optionen sind in sechs Seiten organisiert:

- Guardian enthält alle Einstellungen zum angeschlossenen Prüfgerät
- Programm lässt Sie verschiedene Einstellungen zu Winguard festlegen
- Editor enthält Einstellungen zum Programmeditor
- Aktuelles Projekt enthält Projekt-spezifische Einstellungen
- Kalibrierung erlaubt die Eingabe und das Ändern der Kalibrierdateien für das Guardiansystem
- Geräte listet zusätzlich installierte Hardware

4.1 Speicherort

Die Einstellungen werden für den momentan an Windows angemeldeten Benutzer und optional für alle Benutzer des PCs gespeichert. Für ersteres wird die Datei `%APPDATA%\Winguard\Winguard.ini`, für letzteres die Datei `%ProgramData%\Winguard\Winguard.ini` verwendet. Die Berechtigungen dieser Datei bestimmen, wer die gemeinsamen Einstellungen ändern kann.

Die benutzerspezifischen Einstellungen haben Vorrang vor den Einstellungen für alle Benutzer.

Auf der Seite Programm können die Einstellungen für alle Benutzer gespeichert werden.

4.2 Guardian

4.2.1 Angeschlossenes Gerät

In diesem Dialog werden Einstellungen für das Guardiansystem vorgenommen. Im Feld Schnittstelle wird die RS-232 Schnittstelle des PCs eingegeben, an der das Guardiansystem angeschlossen ist.

Die Komponentenauflistung zeigt die im angeschlossenen Guardian verbauten Zusatzkomponenten mit ihrer Basisadresse und ihrer Anzahl. Sie können über den Button Autoerkennung starten diese Komponenten vom System selbst bestimmen lassen. Dazu muss der Guardian am angegebenen COM-Port angeschlossen und eingeschaltet sein.

ADX-Polung
Legt fest, wie die Messkanäle der ADX-Karte gepolt sind.

PSU mit Synchronisationsoption
Für mit zusätzlicher Hardware für Drehstrom ausgerüstete PSU.

Relais-Karten konfigurieren
Für frühere Versionen der Lastrelaiskarten mit anderer Anzahl Relais.

4.2.2 Optionen

Gerät am Programmende zurücksetzen
Führt einen Reset der Guardianhardware durch, wenn Winguard beendet wird.

ADX Widerstands-Offsetmessung
Diese Funktion verbessert die Genauigkeit von Widerstandsmessungen im Messbereichsanfang. Es wird die Differenz zwischen aus- und eingeschaltetem Messstrom gemessen. Die Messzeit bei Widerstandsmessungen verdoppelt sich hierbei.

Simuliere Gerät in Software
Nützlich um Prüfprogramme an einem Rechner ohne Guardian probeweise ausführen zu können.

Ignoriere Übertragungsfehler
Busfehler werden automatisch gelöscht ohne eine Fehlermeldung auszugeben. Diese Funktion dient nur zu Servicezwecken.

4.3 Programm

4.3.1 Optionen

Letztes Projekt beim Programmstart laden
Beim Programmstart wird das zuletzt geladene Projekt geladen.

Sprache
Stellt die Sprache der Benutzeroberfläche ein

Schrift
Die Schriftart die für Testumgebung und Befehle wie InfoBox.

winguard_dbg.log-Inhalt
Es kann eine Datei mit Informationen angelegt werden, die bei der Behebung von Fehlern in Projekten oder Winguard nützlich sein können. Jede Stufe beinhaltet sämtliche Informationen der vorigen.

Symboleverzeichnis
Symbole werden in diesem Verzeichnis gesucht.

4.3.2 Protokolle

Protokolldatenbank
Wählt aus, in welcher Datenbank die Protokolle gespeichert werden sollen. Über den Button können Sie zusätzliche Verbindungsoptionen für Netzdatenbanken angeben.

Prüfplatz
Der Name des Prüfplatzes wird in jedem Protokoll gespeichert.

In CSV-Datei speichern
Das Messprotokoll wird zusätzlich als CSV-Datei gespeichert. Die CSV-Datei wird im Verzeichnis `%APPDATA%\Winguard` unter dem Namen `PROJEKTNAME-MonatJahr.csv` abgelegt.

Fehler beim Protokollschreiben melden
Diese Option kann abgeschaltet werden, um zu verhindern, dass eine unerreichbare Datenbank den Prüfablauf unterbricht.

Nur angezeigte Ergebnisse speichern
Ergebnisse, die durch Nur im Fehlerfall anzeigen versteckt wurden, werden nicht in der Datenbank oder Textprotokoll gespeichert.

4.3.3 Prüfer-Anmeldung

Prüfer beim Programmstart anmelden
Falls aktiviert, muss der Bediener beim Start von Winguard einen Namen oder ein Kürzel angeben. Dieser Name wird mit jedem Protokoll gespeichert.

Prüferliste in Datenbank
Wenn aktiviert kann über `Prüferliste editieren` eine Liste von gültigen Namen angegeben werden und der Bediener muss beim Start einen dieser Namen wählen. Damit diese Option aktiviert werden kann, muss zuerst die Option `Prüfer beim Programmstart anmelden` aktiviert werden.

4.3.4 Mehrere Benutzer

Einstellungen für alle Benutzer kopieren
Kopiert die aktuell gültigen Einstellungen in die Einstellungsdatei für alle Benutzer.

Eigene Einstellungen verwerfen
Setzt alle Einstellungen auf die Standardwerte zurück. In der Einstellungsdatei für alle Benutzer gespeicherten Einstellungen werden nicht zurückgesetzt.

4.4 Editor

Diese Seite beinhaltet Einstellungen zum Editor von Testprogrammen.

Doppelklick auf Eingabefelder öffnet den Variablenbrowser
Werden Eingabefelder in GUI-Modulen doppelt geklickt, öffnet sich der Variablenbrowser und man kann direkt auf die Programmvariablen zugreifen. Ist diese Option deaktiviert, kann der Variablenbrowser über die Taste F10 geöffnet werden.

Projekt vor dem Starten automatisch speichern
Das Projekt wird vor dem Starten automatische gespeichert.

Editor durch Passwort schützen
Dies ermöglicht das Öffnen des Editors nur nach Eingabe eines Passworts.

Anmerkung
Dies allein verhindert nicht, dass das Prüfprogramm geändert wird. Es müssen zusätzlich die Dateien des Projekts schreibgeschützt werden.

Überwachte Ausdrücke anzeigen
Die Überwachten Ausdrücke werden beim Debuggen automatisch eingeschaltet.

Editor und Testumgebung gleichzeitig anzeigen
Wenn deaktiviert, wird der Editor ausblendet wenn in die Testumgebung gewechselt wird und umgekehrt.

Aussehen
Hiermit kann das Erscheinungsbild des Script-Quelltextes individuell angepasst werden.

4.5 Aktuelles Projekt

Diese Seite enthält Einstellungen, die nur das aktuell geladene Projekt betreffen. Sie werden in der Konfigurationsdatei des Projektes gespeichert.

4.5.1 Allgemeine Einstellungen

Gebe den Testablauf nach dieser Anzahl Fehler auf
Wenn ein Wert größer als 0 angegeben ist, wird ein Testablauf mit der angegebene Zahl an Fehlern vorzeitig beendet. Der Test wird dann als *fehlerhaft* bewertet.

Programm nach dem Laden starten
Das Projekt wird gestartet, sobald der Ladevorgang beendet wurde.

Fehlermeldung bei Tippfehler in Ergebnisvariablen
Nur zur Abwärtskompatibilität mit alten Projekten.

4.5.2 Hardware

Relaiskarte mit Start/Stop-Funktion
Einige Prüfadapter haben eine Stop-Taste, die mit einem Digitaleingang einer Lastrelais-Karte verbunden ist. Mit dieser Option wird der Testablauf abgebrochen, wenn die Taste gedrückt wird.

Strikte ADX2-Überlauferkennung
Wenn eine Messgröße nur zeitweise den Messbereich überschreitet, lieferte die ADX1-Karte keinen Überlauffehler. Ohne diese Option ahmt die ADX2-Karte dies zwecks Abwärtskompatibilität nach.

MSU-Messkanäle sortieren
Die Messkanäle am Guardiansysteminterface werden bei Aktivierung durchgehend nummeriert.

4.5.3 Wiederholungsmodus

Winguard kann nach Ende eines Prüfablaufes automatisch den nächsten Starten. Hier kann konfiguriert werden, wie oft dies geschieht und ob zwischendurch eine Pause eingelegt wird. Dieselben Optionen sind auch von der Testumgebung zugänglich.

4.5.4 Projektparameter

Pfad der Parametertabelle
Pfad zur CSV- oder INI-Datei die die Projektparameter enthält.

Zeile des Projekts in der Tabelle
Hier kann eine ID angegeben werden, mit der der benötigte Eintrag in der Tabelle identifiziert wird. Wenn leer, muss mit `SelectParamDataSet` zur Laufzeit eine Zeile ausgewählt werden.

Schütze Einträge gegen manuelle Änderung
Zu jeder Zeile wird eine Checksumme berechnet, die beim Laden der Tabelle überprüft wird.

4.6 Kalibrierung

Die hardwarebedingten Toleranzen der Guardiankomponenten werden von Winguard rechnerisch kompensiert, um die in der Spezifikation angegebenen Messgenauigkeiten zu erreichen. Die entsprechenden Parameter befinden sich in Kalibrierdateien, die zum Lieferumfang des Testsystems gehören.

Klicken Sie zum Eintragen der Kalibrierdatei auf die Schaltfläche mit den drei Punkten hinter dem Feld Alte Kalibrierdatei, und wählen Sie die zu Ihrem System gehörende Datei aus. Der Name der Datei hat folgenden Aufbau: „Guardian-Seriennummer-Jahr.kal“. Diese Datei enthält die Korrekturwerte der PSU-Karten und des ADX-Multimeters in kombinierter Form. Die Seriennummer des Testsystems befindet sich auf dem Typenschild auf der Rückseite des Gerätes und muss identisch sein mit der Seriennummern der Kalibrierdatei.

Alternativ können zur Kalibrierung von Komponenten auch einzelne Dateien angegeben werden. Diese Möglichkeit ist von Vorteil, wenn im Testsystem eine Karte ersetzt oder nachgerüstet wird. Markieren Sie dazu die Komponente, die Sie kalibrieren wollen (PSU1, PSU2 oder ADX) und klicken Sie auf die Schaltfläche Kalibrierdatei Autodetekt. Winguard liest die Seriennummern der Karten aus und zeigt Sie an. Klicken Sie anschließend auf die entsprechende Schaltfläche [..] und wählen Sie die gewünschte Datei im Verzeichnis *Calibration* aus.

Anmerkung
Bei älteren Karten kann die Seriennummer nicht softwaremäßig ausgelesen werden und es wird die Nummer 00000 angezeigt. In diesem Fall muss die Seriennummer von der Karte abgelesen werden.

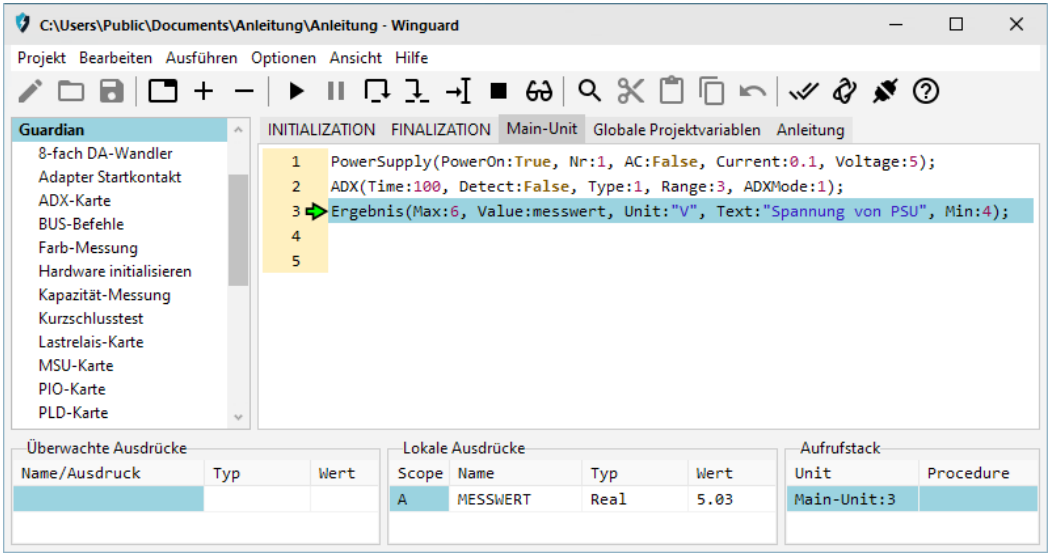
4.6.1 Kalibrierinformation in Hardware

Außerdem können hier die Kalibrierdaten von Guardiankomponenten, die intern kalibriert sind, angezeigt werden.

Kapitel 5

Editor

Mit Hilfe der Editors können Sie eigene Prüfprogramme erstellen. Werkzeuge wie der integrierte Debugger oder das Guardian-Labor unterstützen Sie dabei. In diesem Kapitel werden die einzelnen Konzepte des Editors vorgestellt. Kapitel 7 gibt einen detaillierten Einblick in die Skriptsprache und Kapitel 6 beschreibt die verfügbaren Befehle.



5.1 Erste Schritte im Editor

In diesem Abschnitt werden Sie mit den grundlegenden Funktionen des Editors vertraut gemacht. Anhand kurzer Beispiele werden die wichtigsten Befehle erläutert. Die Programme zu den Beispielen finden Sie im Winguard-Abschnitt des Startmenüs unter Beispiele.

Um in den Editor zu wechseln, verwenden Sie die Schaltfäche Editor oder Strg+F6. Sie können mit Ansicht/Testumgebung oder Strg+F6 in die Testumgebung zurückkehren.

5.1.1 Ein neues Projekt erstellen

Wählen Sie Projekt/Neu um ein neues Projekt zu erstellen. Es öffnet sich ein Dialog in dem Sie den Speicherort und den Namen des Projektes angeben. Nach dem Erzeugen wird das Projekt geladen.

Anmerkung
Ein Winguard-Projekt besteht aus mehreren Dateien. Es empfiehlt sich daher, jedes Projekt in einem eigenen Ordner zu sichern.

5.1.2 Editieren von Projekten

Nachdem ein Projekt erstellt oder geladen wurde, zeigt der Editor links eine Auflistung aller verfügbaren Befehle im *Befehlsbrowser* sowie rechts daneben das *Skript* an.

Der Befehlsbrowser verwaltet die verfügbaren Befehle in Kategorien. Wenn Sie eine Kategorie anklicken, wird diese aufgeklappt und die enthaltenen Befehle eingerückt angezeigt. Zum Beispiel sind die Befehle Infobox, Meldungsbox, Statusleiste und TextOut in der Kategorie Ausgabe.

Sie können die Befehle aus dem Befehlsbrowser ins Skript ziehen. Klicken Sie dazu auf den gewünschten Befehl und halten Sie die Maustaste gedrückt. Bewegen Sie dann den Mauszeiger mit gedrückter Taste ins Skript. Eine schwarze Linie zeigt die Position an, an der der Befehl eingefügt wird. Zum Einfügen lassen Sie die Maustaste los.

Viele Befehle haben Parameter, die Sie in einem Einstellungsdialog eingeben können. Dieser Dialog wird automatisch geöffnet wenn der Befehl ins Skript gezogen wird. Um den Befehl endgültig einzufügen schließen Sie den Dialog mit OK. Mit Abbrechen wird das Einfügen des Befehls verhindert. Sie können den Einstellungsdialog zu einem Befehl jederzeit wieder öffnen indem Sie doppelt auf die entsprechende Zeile klicken.

5.1.3 Hallo, Welt!

Traditionellerweise gibt das erste Programm den Text „Hallo Welt“ aus. Dieses Beispiel verwendet dafür das Messprotokoll.

Ein neues Projekt erstellen
Erstellen Sie wie oben beschrieben ein neues Projekt.

Einen Text ausgeben
Suchen Sie nun im Befehlsbrowser den Befehl TextOut. Dieser befindet sich in der Kategorie Ausgabe. Fügen Sie den Befehl an einer beliebigen Stelle in den Editor ein. Es öffnet sich ein Dialog. Geben Sie im Textfeld den Text ”Hallo, Welt!” ein und schließen Sie den Dialog durch Klick auf OK. Im Editor sollte jetzt diese Zeile erscheinen:

```
TextOut(Text:”Hallo, Welt!”);
```

Wird dieser Befehl beim Prüfablauf abgearbeitet erscheint im Messprotokoll der Testumgebung eine neue Zeile die den Text „Hallo, Welt!“ enthält.

Anmerkung
Soll im Prüfprogramm Text verwendet werden muss dieser immer in Anführungszeichen stehen.

Das Projekt starten
Starten Sie das Projekt indem Sie im Menü Ausführen den Punkt Start klicken oder die Taste F9 drücken. Winguard schaltet automatisch in die Testumgebung und startet dann das Programm. Da nur ein Text ausgegebenwird ist das Programm recht schnell abgearbeitet. Im Messprotokoll finden Sie dann den ausgegebenen Text:

Testschrittname	Untergrenze	Wert	Obergrenze	Bewertung
Hallo Welt				

Anmerkung
Neben dem auszugebenden Text können Sie für den Befehl TextOut noch weitere Parameter festlegen. Eine vollständige Auflistung aller Befehle und aller Parameter finden Sie in Kapitel 6.

5.1.4 Einlesen von Informationen

Mitunter ist es sinnvoll vom Bediener zusätzliche Informationen abzufragen. Dazu existiert in Winguard der Befehl InputBox.

In diesem Projekt werden wir vom Bediener die Eingabe einer Zahl verlangen. Dazu öffnen wir ein kleines Fenster in das die Zahl eingegeben werden soll. Danach geben wir die Zahl im Messprotokoll aus.

Ein neues Projekt erstellen

Erstellen Sie ein neues Projekt wie oben beschrieben.

Eine Bedienerabfrage durchführen

Suchen Sie in der Kategorie Eingabe den Befehl Inputbox und ziehen Sie diesen in den Skripteditor. Füllen Sie den erscheinenden Dialog aus. Der Text den Sie unter Überschrift eintragen erscheint als Titel des Fensters. Der Eingabetext stellt die Aufforderung an den Bediener dar. Der Wert der unter Defaultwert eingetragen wird ist der Standardwert. Er wird verwendet wenn der Bediener den Dialog ohne weitere Eingabe schließt.

Der letzte Punkt gibt den Namen der *Variable* an, in der die Eingabe des Benutzers gespeichert wird. Eine Variable verknüpft einen Wert (hier die Eingabe) mit einem Namen. Variablen müssen vor ihrer Benutzung definiert werden. Jedes Projekt erhält bei Erstellung allerdings eine Menge von vordefinierten Variablen. Geben Sie die Variable `Messwert` in das letzte Feld ein. Eine weiterführende Erklärung zu Variablen finden Sie unter Abschnitt 5.4.

Die Eingabe ausgeben

Fügen Sie einen TextOut-Befehl nach dem Eingabebefehl ein.

Da der Wert in der Variablen `messwert` gespeichert ist, müssen Sie diese als Text eintragen. Um die Ausgabe etwas interessanter zu gestalten geben Sie zusätzlich noch einen Text (wieder in Anführungszeichen) aus. Um beide Ausgaben zu verknüpfen wird der *Operator* + verwendet. Dieser hängt den Wert der Variablen `messwert` an den Text an.

Starten Sie das Programm

Starten Sie das Programm wie im letzten Abschnitt beschrieben. Winguard wird Sie nun zur Eingabe einer Zahl auffordern und danach diese Zahl im Messprotokoll ausgeben.

5.1.5 Auswertung von Messdaten

In diesem Beispiel werden wir vom Benutzer die Eingabe einer Zahl zwischen 10 und 20 verlangen. Anschliessend überprüfen wir die Eingabe des Benutzers indem wir einen Testschritt anlegen.

Neues Projekt erstellen
Erstellen Sie ein neues Projekt.

Eingabe der Zahl
Fordern Sie den Bediener zur Eingabe einer Zahl zwischen 10 und 20 auf. Speichern Sie das Ergebnis der Eingabe in der Variablen `messwert`.

Auswertung der Eingabe
Suchen Sie nun in der Kategorie Auswertung den Befehl Ergebnis und fügen Sie diesen in Ihr Programm ein. Tragen Sie in das Feld Testschrittname (Text) den Text `"Eingabe"` ein. Dieser Text erscheint im Messprotokoll als Name des Testschrittes.

Geben Sie unter Ergebnis (Wert) den Namen der Variablen ein, die den eingelesenen Wert enthält, also `Messwert`.

Nun tragen Sie neben Untergrenze / Obergrenze `10` und `20` ein. Der Wert aus dem Ergebnis-Feld wird dann später mit diesen Grenzen verglichen. Liegt er außerhalb der Grenzen wird der Test als fehlerhaft gewertet ansonsten gilt der Test als bestanden.

Starten des Programms
Starten Sie das Programm wie gehabt. Geben Sie in den Eingabedialog einen Wert ein, der innerhalb der Grenzen liegt, beendet der Test erfolgreich. Liegt der Wert hingegen außerhalb des erwarteten Bereichs wird der Test als fehlgeschlagen betrachtet.

5.1.6 Ablaufsteuerung

Wir wollen nun den Benutzer zu einer korrekten Eingabe zwingen. Dazu möchten wir das Eingabefenster sooft neu anzeigen, bis eine gültige Eingabe vorliegt.

Neues Projekt erstellen
Erstellen Sie wie gewohnt ein neues Projekt.

Ein Label definieren
Suchen Sie in der Kategorie Programmablauf den Befehl Label. Ziehen Sie den Befehl in den Editor und erstellen Sie ein Label mit dem Namen `Einlesen`.

Ein Label kann von einer anderen Stelle des Programms durch Verwendung des Befehls GOTO *angesprungen* werden. Die Ausführung des Programms wird dann an der Position des Labels fortgesetzt.

Eingabe der Zahl
Erfragen Sie vom Benutzer die Eingabe einer Zahl größer als 10 und speichern Sie das Ergebnis in der Variablen `messwert`.

Überprüfung der Eingabe und Anspringen des Labels

Suchen Sie in der Kategorie Programmablauf den Befehl GOTO Label und fügen Sie diesen hinter der Eingabe ein. Es öffnet sich der Eingabedialog für den Befehl GOTO.

Die Liste zeigt alle Label die angesprungen werden können. Wählen Sie hier das einzig mögliche Label `Einlesen` aus. Setzen Sie nun den Haken unter Bedingung und tragen Sie dann in das Eingabefeld (B) die Bedingung `Messwert <= 10` ein.

Wird der Befehl ausgeführt und ist der Wert der Variablen `Messwert` kleiner oder gleich 10, so wird die Ausführung des Programmes am Label `Einlesen` fortgesetzt, d.h. der Benutzer wird erneut um eine Zahl gebeten.

Starten des Programms

Starten Sie das Programm.

Das Programm fordert solange die Eingabe einer Zahl, bis diese größer als 10 ist. Die einzige Möglichkeit das Programm ohne korrekte Eingabe zu beenden, ist im Eingabefenster die Schaltfläche Abbrechen zu klicken. In diesem Fall wird der Test allerdings als Abgebrochen gewertet.

Sie können Label immer dann verwenden, wenn der Ablauf des Prüfprogrammes nicht geradlinig sein soll, also zum Beispiel Wiederholungen enthält.

5.1.7 Ansprechen der Guardian-Hardware

Winguard wurde in erster Linie dazu entwickelt Prüfprogramme auf dem Guardian-Testsystem auszuführen. In diesem Abschnitt werden wir zuerst die Guardian-Stromversorgung einschalten und danach die ausgegebene Spannung messen. Sie benötigen ein angeschlossenes Guardian-Testsystem. Eine Anleitung zur Installation erhalten Sie unter Abschnitt 1.6.

Ein neues Projekt erstellen

Erstellen Sie ein neues Projekt.

Einschalten der Stromversorgung

Suchen Sie in der Kategorie Guardian den Befehl PSU-Karte und ziehen Sie diesen ins Skript. Im Einstellungsdialog wählen Sie DC und tragen Sie ins Feld Spannung `5` und ins Feld Strombegrenzung `0.1` ein. Damit wird die PSU-Karte 5V mit maximal 0,1A ausgeben. Schliessen Sie den Dialog mit OK.

Messen der Spannung

Suchen Sie in der Kategorie Guardian den Befehl ADX-Karte und ziehen Sie ihn ins Skript unter dem `PowerSupply`-Befehl. Wählen Sie die Messart DCV und den Messbereich 20V. Schliessen Sie den Dialog mit OK.

Auswertung des Messwertes
Der ADX-Befehle legt den Messwert in der Variablen `messwert` ab. Überprüfen Sie den Wert dieser Variablen mit Hilfe des Befehls `Ergebnis` aus der Kategorie `Auswertung`. Verwenden Sie als Grenzwerte 4 und 6.

Abschalten der Spannung
Abschliessend sollten Sie die Stromversorgung wieder abschalten. Verwenden Sie dazu in der Unit FINALIZATION ebenfalls den Befehl `PSU-Karte`, wählen Sie diesmal aber die Einstellung `Aus`.

Hardwareaufbau
Bereiten Sie nun den Guardian für die Messung vor. Verbinden Sie dazu die Ausgänge `PSU1+` und `PSU1-` an der Frontseite des Testsystems mit den Eingängen `ADX+` und `ADX-` (ebenfalls an der Frontseite).

Starten des Programmes
Starten Sie nun das Programm. Mit dem ersten Befehl wird eine Spannung von 5V auf der Guardian-Stromversorgung erzeugt. Diese 5V werden über die im letzten Schritt hinzugefügten Verbindungen an die Messkanäle der ADX-Karte angelegt. Mit dem Messbefehl wird diese Spannung abgefragt, um anschließend vom Befehls `Ergebnis` in das Messprotokoll geschrieben zu werden. Wenn Ihr Guardiansystem korrekt funktioniert sollte der Test mit `Pass` bewertet werden.

5.1.8 Units und Prozeduren

Ein Winguard-Prüfprogramm ist in *Units* und *Prozeduren* unterteilt. Eine Prozedur ist ein abgeschlossener Teil des Prüfprogrammes, der von anderen Teilen aufgerufen werden kann und nach dessen Abarbeitung zu der aufrufenden Stelle zurückgekehrt wird. Prozeduren werden durch ihren Namen eindeutig bestimmt. Außerdem können einer Prozedur mehrere Parameter übergeben werden, die die Art der Abarbeitung beeinflussen können.

Eine Unit fasst mehrere Prozeduren zu einer logischen Einheit zusammen. Units werden ebenfalls durch ihre Namen identifiziert.

Units und Prozeduren dienen der Organisation des Prüfprogramms und sollen die Wiederverwendbarkeit von Programmcode erleichtern. Im folgenden Beispiel werden wir eine neue Unit anlegen, in dieser Unit eine Prozedur definieren und diese dann im Hauptprogramm aufrufen. Weiterführende Informationen zu Units und Prozeduren finden Sie in den Abschnitten Abschnitt 5.2.7 und Abschnitt 5.3.

In diesem Beispiel erstellen wir eine Unit die eine Prozedur zur Ausgabe von Zahlen enthält. Die Zahl soll der Prozedur dabei als Parameter übergeben werden.

Neues Projekt erstellen

Erstellen Sie ein neues Projekt.

Neue Unit erstellen

Wählen Sie im Menü Projekt den Punkt Neue Unit. Winguard fragt Sie daraufhin nach dem Namen der neuen Unit. Geben Sie einen Namen für die neue Unit ein, zum Beispiel `MeineUnit` und klicken Sie OK.

Winguard erstellt daraufhin eine neue Unit und wechselt automatisch in die neue Unit. Alle Befehle die Sie jetzt erzeugen, werden in die neue Unit eingefügt.

Eine neue Prozedur erstellen

Suchen Sie in der Kategorie Unterprogramme den Befehl Neu Deklarieren und ziehen Sie ihn in die neue Unit.

Geben Sie unter Name der Routine den Namen `Ausgabe` des neuen Unterprogramms ein. Klicken Sie als nächstes in die obere Tabelle und geben Sie den Namen des Parameters `Wert` ein. Wechseln Sie in die nächste Spalte. Durch Klick auf den Pfeil oder Alt+↓ öffnen Sie eine Auswahlliste für den Typ des Parameters. Der Typ schränkt die Art der Daten ein, die beim Aufruf der Prozedur als Parameter verwendet werden können. Wählen Sie Real um nur Zahlen als Parameter zuzulassen.

Schließen Sie nun den Dialog mit OK. Im Skripteditor erscheinen nun zwei neue Zeilen. Die Zeile `procedure Ausgabe(Wert : Real);` wird *Prozedurkopf* genannt. Sie enthält alle wichtigen Informationen zu der Prozedur wie ihren Namen und die Liste der Parameter. Die Prozedur endet mit der Zeile `end procedure`. Wird die Prozedur aufgerufen werden alle Zeilen zwischen dem Kopf und dem Ende der Prozedur nacheinander abgearbeitet.

Die Prozedur füllen

Ziehen Sie den Befehl TextOut der Kategorie Ausgabe zwischen den Kopf und das Ende der Prozedur.

In der Prozedur können Sie die Parameter der Prozedur wie eine gewöhnliche Variable verwenden. Erzeugen Sie nun die Ausgabe des Parameters.

Aufruf der Prozedur

Zuletzt müssen Sie die Prozedur noch aufrufen. Dazu müssen Sie zuerst in das Hauptprogramm wechseln.

Mithilfe der Reiter können Sie zwischen Units hin- und herschalten. Wählen Sie hier die Unit Main-Unit aus.

Suchen Sie nun den Befehl Aufruf in der Kategorie Unterprogramme und ziehen Sie ihn in den Editor.

Im oberen Eingabefeld muss die aufzurufende Prozedur ausgewählt werden. Wir wählen die (einzige) Prozedur `Ausgabe` aus. In der Tabelle darunter werden die Parameter

der aktuellen Prozedur angezeigt. In der Spalte Wert können wir den Wert des Parameters eintragen. Tragen Sie hier eine beliebige Zahl ein und schließen Sie den Dialog mit OK.

Wenn wir unser Programm ausführen erscheint im Messprotokoll die Ausgabe `Die Zahl hat den Wert 12`.

Wenn wir die Prozedur mehrmals in unserem Programm aufrufen reicht es aus, die Prozedur anzupassen um alle Ausgaben zu verändern. Prozeduren helfen somit Programme übersichtlich und wartbar zu halten. Um die Lesbarkeit Ihrer Programme zu erhöhen sollten Sie ihren Prozeduren „sprechende" Namen geben, also `Ausgabe` statt `x` oder `StromversorgungEinschalten` statt `On`. Verwenden Sie in einem Programm mehrere Prozeduren, so organisieren Sie diese in Units. Vergeben Sie auch an Ihre Units sprechende Namen.

5.2 Quelltexteditor

Winguard skripte werden mit einem Zeileneditor bearbeitet. In diesem Abschnitt werden die Funktionen des Quelltexteditors beschrieben.

5.2.1 GUI-Module

Winguard bietet zu jedem Befehl einen Einstellungsdialog um den Befehl zu konfigurieren. Diese Dialog werden *GUI-Module* genannt. Das GUI-Modul zu einem Befehl öffnet sich, wenn Sie den entsprechenden Befehl aus dem Befehlsbrowser in den Editor ziehen. Für bereits erstellte Befehle können Sie das GUI-Modul öffnen indem Sie die Zeile des Befehls doppelt klicken. Sie können auch die entsprechende Zeile im Quelltexteditor markieren und die Taste F2 drücken.

Das GUI-Modul soll Sie beim Editieren der Befehlseinstellungen unterstützen. Ist das Modul geöffnet steht Ihnen durch Drücken der Taste F1 die Online-Hilfe mit einer Erläuterung aller Einstellungen des aktuellen Befehls zur Verfügung.

GUI-Module erkennen außerdem syntaktische Fehler und weisen Sie auf diese bereits zur Entwicklungszeit hin. Das fehlerhafte Feld wird dabei ausgewählt.

5.2.2 Erweitertes Editieren

Zur schnellen Eingabe kurzer Befehle erlaubt Winguard die direkte Eingabe von Befehlszeilen über die Tastatur. Durch Drücken der Buchstabentasten oder der Leertaste öffnet sich für

die auswählte Zeile eine Texteingabe in die Sie den Quelltext direkt eintippen können. Der Quelltext wird in den Quelltexteditor übernommen sobald Sie ↵ drücken.

Während Sie tippen wird ihnen zusätzlich eine Liste von Vorschlägen angezeigt, die den von Ihnen getippten Text enthalten. Mit den Tasten ↑ und ↓ können Sie in dieser Liste einzelne Befehle auswählen und deren GUI-Modul durch Drücken von ↵ öffnen beziehungsweise den Vorschlag in die Befehlszeile übernehmen.

Anmerkung
Im Gegensatz zu GUI-Modulen übernimmt die erweiterte Eingabe den von Ihnen eingetippten Quelltext ohne Überprüfung. Sollte der so eingegebene Quelltext Fehler enthalten, werden diese erst beim Programmstart angezeigt. Außerdem können Sie unter Umständen zu einem so erzeugten fehlerhaften Befehl das dazugehörige GUI-Modul nicht öffnen.

5.2.3 Verschiedene Funktionen

Um zu den Zeilen im Editor die jeweilige Zeilennummer anzeigen zu lassen, wählen Sie `Ansicht/Zeilennummern`.

Der Editor verwaltet zu jeder Unit eine Undo-Liste. Damit können alle Änderungen an der jeweiligen Unit bis zum Laden des Programms rückgängig gemacht werden. Eine Redo-Funktion ist zur Zeit nicht verfügbar.

5.2.4 Kontextmenü

Das Kontextmenü öffnet sich, wenn Sie mit der rechten Maustaste in den Editor klicken. Die Aktionen des Menüs beziehen sich auf die geklickte Zeile:

Kopieren
: Kopieren der aktuellen Zeile

Ausschneiden
: Ausschneiden der aktuellen Zeile

Einfügen
: Einfügen einer zuvor ausgeschnittenen oder kopierten Zeile

Haltepunkt
: Breakpoint für die aktuelle Zeile umschalten

Zeile (De)aktivieren
aktuelle Zeile aktivieren/deaktivieren

Pocedure-Explorer
Öffnen des Prozedurexplorers

Zur Prozedur
Nur für Zeilen mit Prozeduraufruf: springe zur Prozedurdefinition

5.2.5 Syntaxhervorhebung

Um die Lesbarkeit von Programmen zu erleichtern hebt Winguard die einzelnen Elemente des Quelltextes optisch unterschiedlich hervor.

- Schlüsselwörter wie `procedure` oder `if` werden **fett** geschrieben
- Werte wie Zahlen und Zeichenketten werden rot dargestellt.
- Kommentare werden in *blauer kursiver* Schrift dargestellt
- Bezeichner werden ohne Textdekorationen dargestellt

5.2.6 Editieren von Projekten

Während des Editierens im Quelltexteditor ist die aktuelle Zeile automatisch markiert. Sie erkennen die Zeile an dem blauen Balken. Viele Operationen des Quelltexteditors können auf mehrere Zeilen gleichzeitig angewendet werden. Dazu muss eine Gruppe von Zeilen, ein sogenannter *Block* markiert werden.

Markieren von Blöcken
Um einen Block zu markieren wählen Sie zuerst die erste Zeile des Blockes mit Hilfe der Tasten ↑ und ↓. Halten Sie nun die Taste ⇧ gedrückt und benutzen Sie wiederum die Tasten ↑ und ↓ um mehrere Zeilen zu markieren. Alternativ können Sie einen Block mit der Maus markieren. Klicken Sie dazu auf die erste Zeile des Blocks und halten Sie die Maustaste gedrückt. Ziehen Sie nun die Maus bis alle gewünschten Zeilen markiert sind.

Kopieren, Ausschneiden, Einfügen und Löschen
Blöcke können über die gängigen Tastenkombinationen kopiert, ausgeschnitten, eingefügt und gelöscht werden. Der Editor bietet diese Aktionen außerdem im Kontextmenü an, das Sie über einen Rechtsklick auf den Editor öffnen können. Die Tastenkombinationen sind:

Strg+C	Kopiert den aktuellen Block in die Zwischenablage
Strg+X	Kopiert den aktuellen Block in die Zwischenablageund löscht ihn aus dem Editor (Ausschneiden).
Strg+V	Fügt den Inhalt der Zwischenablage vor der aktuellen Zeile ein.
Entf	Löscht den aktuellen Block.

Einrücken von Blöcken

Blöcke und Zeilen können Sie durch Drücken der Taste ↹ um vier Zeichen nach rechts einrücken. Um die Einrückung aufzuheben und den markierten Block vier Zeichen nach links zu schieben drücken Sie ⇧+↹.

Deaktivieren von Blöcken

Möchten Sie einen Teil des Prüfprogramms vorübergehend deaktivieren, jedoch den entsprechenden Code nicht aus dem Quelltext löschen, können Sie den gewünschten Teil durch drücken von F12 deaktivieren. Er wird dann im Editor in *grauer kursiver* Schrift dargestellt. Deaktivierte Schritte können durch erneutes drücken von F12 wieder aktiviert werden.

5.2.7 Navigation in Units und Prozeduren

Übersicht der Projektdateien

Um eine Übersicht über alle Projektdateien zu erhalten, öffnen Sie über das Menü Projekt/Units... den Projekt-Units-Dialog.

Neben dem Pfad und dem Namen des Projektes finden Sie hier eine Auflistung aller Units die von dem Projekt verwendet werden. Zu jeder Unit wird außerdem der Pfad angezeigt.

Umschalten zwischen Units

Um zwischen den einzelnen Units des Projektes zu wechseln, verwenden Sie die Reiter oberhalb des Skripts.

Um zur nächsten bzw. vorherigen Unit zu springen, können Sie außerdem die Tastenkombination Strg+↹ bzw. Strg+⇧+↹ verwenden.

Die Standardunits sind außerdem direkt über das Menü Ansicht und das Untermenü Gehe zu Unit erreichbar. Außerdem können Sie folgende Tastenkombinationen verwenden, um schnell zu einer der Standardunits zu wechseln:

Strg+F9 Main-Unit

Strg+F10 Globale Variablen

Strg+F11 Initialisierung

Strg+F12 Finalisierung

Zu Prozeduren springen

Um bei vielen Prozeduren den Überblick zu behalten empfiehlt sich die Verwendung des *Prozedurexplorers*. Diesen können Sie über das Kontextmenü des Editors öffnen. Der Prozedurexplorer bietet Ihnen eine Liste aller Prozeduren die in den Units des Projektes definiert wurden.

Durch Doppelklick auf einen Prozedurnamen springt der Quelltexteditor zum entsprechenden Prozedurkopf.

Möchten Sie zu einem Prozeduraufruf die aufgerufene Prozedur einsehen, suchen Sie im Kontextmenü des Editors den Punkt Prozedur beim Cursor suchen. Alternativ finden Sie im GUI-Modul des Aufrufs eine Schaltfläche Zum Quelltext springen die ebenfalls zu der aufgerufenen Prozedur springt.

Der Quelltexteditor erlaubt zusätzlich die erweiterte Navigation mittels Cursortasten. Ist die Zeile eines Prozeduraufrufs markiert, können Sie durch drücken von Strg+← die aufgerufene Prozedur besuchen. Durch drücken von Strg+→ gelangen Sie wieder zum Aufruf zurück. Wenn Sie mehrmals hintereinander zu einer Prozedur springen, können Sie auch mehrmals zurückspringen.

5.3 Prozeduren

Dieser Abschnitt beschreibt den Dialog zur Prozedurdefinition.

Oben findet sich der Name der Prozedur, die Unit in der die Prozedur definiert wurden und die Zeilennummer des Prozedurkopfes.

In der Mitte werden die Übergabeparameter der Prozedur definiert. Parameter können in der Prozedur wie Variablen angesprochen werden. Die Reihenfolge in der Tabelle spiegelt die Reihenfolge beim Aufruf wider. Zu jedem Parameter muss der Name, der Datentyp sowie die Art der Übergabe angegeben werden. Klicken Sie dazu auf dem Pfeil in der entsprechenden Tabellenzelle. Es öffnet sich eine Auswahl der Typen bzw. der Übergabearten.

Der Datentyp eines Parameters kann String für Zeichenketten, Real für Zahlen oder Array of ... für eine Liste von Zeichenketten oder Zahlen sein.

Der Übergabetyp gibt an, wie der Parameter übergeben wird. Die Übergabe als Wert erzeugt bei Aufruf der Prozedur eine neue Variable der der übergebene Wert zugewiesen wird. Diese Art der Übergabe lässt auch die Verwendung von Ausdrücken zu.

Im Gegensatz dazu wird bei der Übergabe als Referenz eine bereits existierende Variable unter dem Namen des Parameters verfügbar gemacht. Beim Aufruf muss daher eine Variable des gleichen Typs angegeben werden. Änderungen die in der Prozedur an der Variable vorgenommen werden, ändern auch deren Wert in der aufrufenden Prozedur. Auf diese Art können Informationen aus der Prozedur an ihren Aufrufer zurückgegeben werden.

Unten werden lokale Variablen definiert. Lokale Variablen können nur in der Prozedur verwendet werden, in der sie definiert wurden. Lokale Variablen verdecken globale Variablen mit dem gleichen Namen.

5.4 Variablen und Konstanten

Dieser Abschnitt beschreibt den in Winguard integrierten Variablenbrowser. Es wird erklärt, wie Variablen angelegt, gelöscht und editiert werden können. Außerdem werden Konstanten und ihre Verwendung vorgestellt.

5.4.1 Variablenbrowser

Um den Variablenbrowser zu öffnen verwenden Sie Bearbeiten/Variablen oder F10.

Der Variablenbrowser bietet eine Übersicht über alle im aktuellen Projekt definierten Variablen. Neue Projekte enthalten nur vordefinierte Systemvariablen. Diese werden durch ein SYS vor dem Variablennamen markiert. Die Anzeige sortiert die Variablen nach ihren Typen und nach ihrem Gültigkeitsbereich (global oder lokal). Variablen werden durch ihren Namen, gefolgt von einer Bemerkung, dargestellt.

5.4.1.1 Erstellen von Variablen

Um eine neue Variable zu erstellen klicken Sie auf die Schaltfläche Neu. Daraufhin öffnet sich der *Variableneditor*.

Geben Sie oben den Namen der Variablen ein. Dieser muss den Regeln für gültige Variablennamen entsprechen und darf von keiner anderen Variablen verwendet werden.

Die Bemerkung kann die Verwendung der Variablen beschreiben. Sie erscheint im Variablenbrowser neben dem Variablennamen. Ihre Angabe ist optional.

Der Defaultwert wird der Variablen beim Programmstart zugewiesen. Hier muss ein gültiger Zahlenwert oder eine Zeichenkette angegeben werden. Die Angabe eines Defaultwertes ist ebenfalls optional. Ist kein Defaultwert angegeben wird der Variablen je nach Typ eine leere Zeichenkette, die Zahl Null oder ein leeres Array zugewiesen.

Darunter wird der Typ und der Gültigkeitsbereich der Variablen festgelegt. Wählen Sie Real für Zahlenwerte, String für Zeichenketten und die Array-Varianten, falls Sie mehrere Werte des entsprechenden Typs unter einem Variablennamen speichern wollen. Weiterführende Informationen zu Typen finden Sie unter Abschnitt 7.1.1.

Der Gültigkeitsbereich (entweder Global oder Lokal) gibt an, wo die Variable bekannt sein soll. Globale Variablen können im gesamten Programm verwendet werden, lokale Variablen nur in der Prozedur, in der sie definiert wurden. Daher kann ein lokaler Gültigkeitsbereich nur ausgewählt werden, wenn im Editor die Zeile einer Prozedur aktiviert ist. Andernfalls sind diese Optionen gesperrt.

Unten können zusätzliche Eigenschaften der Variablen festgelegt werden. Die Option Variable nicht zurücksetzen übernimmt den Wert der Variable aus dem vorherigen Prüfablauf. Damit kann beispielsweise eine Benutzereingabe für eine Prüfserie einmal am Anfang gespeichert und bei weiteren Prüflingen wiederverwendet werden. Ist die Option Überwachter Ausdruck aktiviert, so erscheint die Variable während dem Debuggen des Programms in den überwachten Ausdrücken (siehe Abschnitt 5.6.3).

Um die Variable endgültig anzulegen klicken Sie die Schaltfläche OK. Der Variableneditor wird geschlossen und Sie finden die Variable im Variablenbrowser wieder.

5.4.1.2 Löschen von Variablen

Um eine Variable zu löschen, wählen Sie im Variablenbrowser die Variable aus. Der Name der Variable erscheint im Textfeld Gewählte Variable. Um die ausgewählte Variable zu löschen klicken Sie die Schaltfläche Löschen.

5.4.1.3 Editieren von Variablen

Sie können einige der Eigenschaften einer Variablen nachträglich ändern. Wählen Sie dazu zuerst die Variable aus. Klicken Sie dann Bearbeiten um den Variableneditor zu öffnen.

5.4.2 Konstanten

Konstanten ordnen Bezeichnern Werte zu, können jedoch zur Laufzeit nicht verändert werden. Durch diese Zuordnung erhöhen Konstante die Lesbarkeit und die Wartbarkeit von Prüfprogrammen.

Konstanten werden in Winguard im Konstanteneditor verwaltet. Der Editor lässt sich über das Menü Bearbeiten Menüpunkt Konstanten oder über die Tastenkombination Shift-F10 öffnen.

Es werden alle Konstanten in der Tabelle dargestellt. Der Wert kann durch anklicken editiert werden. Beachten Sie, dass Zeichenketten in Anführungszeichen angegeben werden müssen. Um eine neue Konstante anzulegen, klicken Sie die Schaltfläche Neu. Es öffnet sich ein Eingabefenster in das Sie den Name der Konstanten eingeben. Sie können den Wert durch = getrennt eingeben: `Foo=1` legt eine Konstante `Foo` mit dem Wert `1` an. Die Eingabe von `Bar` erzeugt eine Konstante ohne Wert. Sie sollten in diesem Fall einen Wert nachträglich eingeben. Konstanten die keinen Wert besitzen, werden beim Schliessen des Editors gelöscht.

Um eine Konstante explizit zu löschen, aktivieren Sie die Zeile der Konstante und klicken Sie die Schaltfläche Löschen.

5.5 Projektparameter

Für bauähnliche Prüflinge unterscheiden sich Prüfabläufe meistens nur in wenigen Details, wie zum Beispiel Toleranzgrenzen oder Art der Stromversorgung. Um Mehraufwand bei Entwicklung und Pflege solcher Testfälle zu vermeiden, können sogenannte Projektparameter verwendet werden. Projektparameter werden in einer Tabelle gespeichert und anhand einer Produkt-ID identifiziert. Dazu wird beim Start des Prüfprogrammes der gewünschte Datensatz aus der Tabelle gelesen. Die Felder des Datensatzes können dann im weiteren Programmablauf in Ausdrücken verwendet werden. Im Folgenden wird anhand eines Beispiels die Verwendung der Projektparameter erläutert.

Wir möchten einen Prüfablauf entwickeln mit dem wir verschiedene baugleiche Relais testen können. In unserem Beispiel möchten wir drei verschiedene Relaistypen namens Rel5, Rel12 und Rel24 mit Nennspannungen von 5V, 12V und 24V prüfen. Dabei sind die Anschlüsse der Relaisspule mit der Stromversorgung (PSU1) verbunden. Die Messkanäle 1 und 2 sind verbunden mit den Kontakten des Relais, so dass wir durch eine Widerstandsmessung an diesen Kanälen feststellen können, ob das Relais geschlossen ist.

Wir entwerfen zuerst einen statischen Prüfablauf. Wir beginnen damit das Netzteil einzuschalten und die Messkanäle zu verbinden. Danach messen wir den Widerstand auf dem Relaiskontakt und werten diesen aus. Zuletzt wird die Stromversorgung abgeschaltet.

```
// Netzteil einschalten
```

```
PowerSupply(PowerOn:True, Nr:1, AC:False, Current:0.1, Voltage:0);

// Messkanäle 1 und 2 schalten
MSU(DiPol:True, IdA:1, IdB:2, Supply:[]);

// Widerstandsmessung
ADX(Time:20, Detect:False, Type:3, Range:1, ADXMode:1);

// Auswertung
Ergebnis(Max:150, Value:Messwert, Unit:"Ohm", Text:"Relaiskontakt  ↩
    geschlossen", Min:0);

// Netzteil abschalten
PowerSupply(PowerOn:False, Nr:1, AC:False, Current:0, Voltage:0);
```

Wir wollen jetzt das Programm so anpassen, dass die anzulegende Spannung aus der Projektparametertabelle gelesen wird. Dazu öffnen wir zuerst den Einstellungsdialog für Projektoptionen: Optionen/Einstellungen/Aktuelles Projekt. Hier finden sich die Einstellungen zu den Projektparametern.

Zuerst müssen wir den Pfad zur Parametertabelle angeben. Geben wir eine Datei an, die noch nicht existiert, so wird eine neue Tabelle angelegt. Die Tabelle wird im CSV-Format gespeichert und kann mit einem beliebigen Texteditor eingesehen werden. Um Änderungen an der Tabelle über einen Texteditor zu verhindern kann ausserdem der Schalter Schütze Einträge gegen manuelle Änderung gesetzt werden. Wir legen eine neue Tabelle namens *Projektparameter* an.

Wurde für das Projekt eine Tabelle definiert, können wir diese über den integrierten Tabelleneditor editieren. Dieser kann über den Menüpunkt Bearbeiten/Prüfparametertabelle geöffnet werden. Eine neue Tabelle wird immer mit den vordefinierten Feldern *ID*, *Version*, *Datum*, *Benutzer* und *Kommentar* erstellt. Das Feld *ID* enthält eine eindeutige Bezeichnung (zum Beispiel Artikelnummer) des Datensatzes. Sie wird einmalig beim Erstellen eines neuen Eintrages abgefragt. Die Felder *Datum* und *Benutzer* geben an, durch welchen Benutzer der Eintrag wann zuletzt geändert wurde. Die Versionsnummer wird bei jeder Änderung um eins erhöht. Das Feld *Kommentar* kann beliebig gefüllt werden und sollte eine kurze Beschreibung des Datensatzes enthalten. Bis auf das Feld *Kommentar* kann keines dieser Felder vom Benutzer verändert werden.

Durch Neuer Parameter kann zu den bestehenden Feldern ein neues Feld hinzugefügt werden. Dazu ist die Eingabe eines Feldnamens nötig. Alle Feldnamen müssen eindeutig sein. Wurde der Feldname vergeben, wird eine neue Spalte in der Tabelle angelegt. Diese kann später vom Programm aus abgefragt werden. Wir legen eine Spalte mit der Bezeichnung *Nennspannung* an.

Nun legen wir für jeden unserer Prüflinge einen neuen Datensatz an. Um einen Datensatz

anzulegen klicken wir auf Neue ID. Ein Dialog fragt nach der ID für den neuen Datensatz. Hier geben wir *Rel5* für das erste Relais ein.

Die Felder *Kommentar* und *Nennspannung* können durch anklicken editiert werden. Hier tragen wir einen sinnvollen Kommentar und die Nennspannung ein. Wir wiederholen diese Schritte für die beiden anderen Relaistypen, diesmal mit anderer ID und Nennspannung. Zum Schluss sollte die Tabelle so aussehen:

ID	Version	Datum	Benutzer	Kommentar	Nennspannung
Rel5	1	12.12.2022 16:09:05	GTS	Relais mit 5V Nenn-spannung	5
Rel12	1	12.12.2022 16:10:16	GTS	Relais mit 12V Nenn-spannung	12
Rel24	1	12.12.2022 16:10:23	GTS	Relais mit 24V Nenn-spannung	24

Beim Start des Prüfprogramms müssen wir den aktuellen Datensatz auswählen. Dazu verwenden wir die Funktion `SelectParamDataSet`, der wir die ID als String übergeben. Wird eine nicht vorhandene ID übergeben erhalten wir eine Fehlermeldung. Wir erfragen die ID beim Benutzer durch ein Eingabefeld und speichern das Ergebnis in der Variablen `RXDString`. Danach rufen wir die Funktion `SelectParamDataSet` auf und übergeben als Parameter die eingelesene ID.

```
// Abfrage der Datensatz-ID
repeat
    EingabeBox(Prompt:”Rel5, Rel12 oder Rel24”, Variable:RXDSTRING, ↩
        Caption:”Relaistyp”);
until SelectParamDataSet(RXDSTRING)
```

Zuletzt müssen wir den Wert der Spannung aus der Tabelle lesen. Dazu stehen uns zwei Funktionen `GetStrParam` und `GetRealParam` zur Verfügung, wobei `GetStrParam` den Wert des Feldes direkt zurückgibt, während `GetRealParam` versucht, aus dem Feldinhalt eine Zahl auszulesen. Wir verwenden diese um die Spannung für das Netzteil anzugeben:

```
// Netzteil einschalten
PowerSupply(PowerOn:True, Nr:1, AC:False, Current:0.1, Voltage: ↩
    GetRealParam(”Nennspannung”));
```

Führen wir das Programm aus, wird das Netzteil abhängig von der Relais-ID beschaltet. Die Tabelle kann beliebig viele Felder verwalten.

5.6 Debugger

Es kommt vor, dass ein Prüfablauf nicht wie geplant funktioniert. Der Debugger kann Ihnen bei der Fehlersuche helfen, indem das Testprogramm zwischendrin angehalten und untersucht wird.

5.6.1 Aktivieren des Debuggers

Das Programm kann mit dem Debugger auf verschiedene Weisen angehalten werden:

- Ausführen/Unterbrechen unterbricht ein laufendes Testprogramm an der als nächstes auszuführenden Zeile.
- Ausführen/Einzelne Anweisung oder F7 setzt ein unterbrochenes oder nicht gestartetes Testprogramm für nur eine Zeile fort.
- Ausführen/Gesamte Routine oder F8 setzt ein unterbrochenes oder nicht gestartetes Testprogramm für einen Unterprogrammaufruf oder eine Zeile fort.
- Ausführen/Bis Cursorposition oder F4 setzt ein unterbrochenes oder nicht gestartetes Testprogramm fort, bis es die gerade im Editor ausgewählte Zeile erreicht.
- Mit Ausführen/Haltepunkt oder F6 kann die gerade im Editor ausgewählte Zeile mit einem Haltepunkt markiert werden. Erreicht das Testprogramm diese Zeile, wird das Programm angehalten, bevor die Zeile ausgeführt wird. Eine Zeile mit Haltepunkt wird in der ersten Spalte mit einem roten Punkt markiert.
- Der Knopf Debug im Laufzeitfehlerdialog.

Der Zustand des Testprogramms kann mit Ansicht/Überwachte Ausdrücke oder Strg+W angezeigt werden.

5.6.2 Lokale Ausdrücke

Die Tabelle *lokale Ausdrücke* zeigt alle Variablen an die im aktuellen Kontext von Interesse sind. Das sind die lokalen Variablen und die Parameter der aktuellen Prozedur sowie alle Variablen die im aktuellen und im zuletzt ausgeführten Befehl verwendet werden.

Das Feld *Scope* gibt an, aus welchem Grund die Variable in der Liste aufgeführt wird. Ein *P* steht dabei für Prozedur und bedeutet, dass die Variable entweder ein Parameter oder eine lokale Variable ist. Ein *A* zeigt an, dass die Variable innerhalb eines Ausdruckes gefunden wurde, also dem aktuellen oder dem letzten Befehl entstammt.

5.6.3 Überwachte Ausdrücke

Die *Überwachten Ausdrücke* sind ähnlich zu den lokalen Ausdrücken, allerdings können die Einträge frei gewählt werden. Name kann hier ein beliebiger Variablenname aber auch ein kompletter Ausdruck sein. Sie können damit zur Laufzeit Berechnungen direkt überprüfen.

Um einen Ausdruck hinzuzufügen oder zu Editieren klicken Sie mit der Maus auf ein Feld in der Spalte *Name*. Sie können den Ausdruck dann direkt eingeben. Kann der Ausdruck nicht ausgewertet werden, erscheint sowohl im Feld *Type* als auch im Feld *Wert* der Text „Fehler bei der Auswertung".

5.6.4 Aufrufstack

Der *Aufrufstack* zeigt während der Ausführung eine Liste der aufgerufenen Prozeduren an. Damit können Sie nachvollziehen, von welcher Stelle aus die aktuelle Prozedur aufgerufen wurde. Hinter dem Unitnamen steht die Zeilennummer in der Unit und hinter dem Prozedurnamen die Zeilennummer in der Prozedur selbst.

Klicken Sie einen Eintrag in der Liste mit der Maus doppelt an, springt die Anzeige zu dieser Aufrufstelle.

Kapitel 6

Befehle

6.1 Ausgabe

6.1.1 Infobox

Der Befehl „InfoBox“ erlaubt Ihnen, Text in einem Fenster auszugeben. Das Fenster verfügt über keine Schaltflächen und wird durch das Programm geöffnet und geschlossen. Die Fenstergröße wird an den anzuzeigenden Text angepasst.

Fenster schließen
Schliesst eine zuvor geöffnete Infobox. Die Infobox wird ausserdem geschlossen wenn der Testlauf beendet wird.

Blinkend
Mit dieser Option blinkt der Hintergrund des Fensters gelb. Dies erzeugt eine größere Aufmerksamkeit.

Schriftgröße
Legt die Schriftgröße fest.

XPos, YPos
Legt die Position des Fensters in Prozent fest (50/50 = Mitte).

Ohne Rahmen
Das Fenster erhält keinen Rahmen.

6.1.2 Meldungsbox

Erzeugt einen Dialog auf dem Bildschirm und wartet, bis der Benutzer eine Schaltfläche anklickt. Es können mehrere Schaltflächen kombiniert werden. Nachdem der Befehl ausgeführt wurde, steht in der Systemvariablen `BUTTONRESULT` der Name der angeklickten Schaltfläche in Großbuchstaben. Eine Ausnahme bildet die Schaltfläche Abbrechen. Diese führt sofort zum Abbruch des Programmablaufes.

Meldungstyp
Gibt an, welches Symbol in dem Meldungsfenster eingeblendet wird.

Buttons
Hier können Sie die Schaltflächen auswählen, die im Meldungsfenster erscheinen sollen. Dies kann eine beliebige Auswahl der Standardknöpfe sowie ein Array mit individuellen Beschriftungen sein.

Position
Hier stellen Sie die Position des Fensters ein. Ist „Standard“ gewählt, wird das Fenster in der Mitte des Bildschirms angezeigt.

6.1.3 Statusleiste

Mit diesem Befehl können Sie Informationen in die Statusleiste von Winguard schreiben. Diese Zeile befindet sich ganz unten im Fenster.

6.1.4 TextOut

Gibt eine Zeile Text im Messprotokoll aus, ohne Messwert oder Grenzen. Die eingestellte Schriftgröße ist relativ zur normalen Schriftgröße des Messprotokolls, Schriftgröße 10 wird so groß dargestellt wie die Zeilen des Befehls Ergebnis.

6.2 Auswertung

6.2.1 Ergebnis

Der Ergebnisbefehl wird dazu verwendet, die im Feld Ergebnis angegebene Variable mit einer Ober- und Untergrenze zu vergleichen und entsprechend zu bewerten (Pass/Fail). Diese Informationen werden auch in das Messprotokoll geschrieben.

Testschrittname
Dieser Text erscheint in der ersten Spalte des Messprotokolls.

Ergebnis (Wert)
Hier wird die Variable oder der Ausdruck eingegeben, deren Inhalt bewertet werden soll (im Folgenden Messwert genannt).

Einheit
Die Maßeinheit des Messwertes als Stringkonstante, z.B. `"KOhm"`

Untergrenze / Obergrenze
Liegt der Messwert außerhalb dieser Werte, wird eine rote Zeile mit dem Status FAIL erzeugt. Anderenfalls wird eine grüne Zeile mit dem Status PASS erzeugt.

Sollwertvorgabe
Hier kann als Grenzwert ein Sollwert mit Toleranzangabe in % eingegeben werden.

Immer Pass
Nützlich, wenn nur ein Wert ins Protokoll aufgenommen werden soll.

Immer Fail
Hiermit kann eine Fehlermeldung ins Protokoll geschrieben werden.

Offset, Faktor
Der Messwert kann vor der Bewertung noch skaliert und Offset-kompensiert werden. Dies berechnet sich nach folgender Formel: Messwert = Messwert × Faktor + Offset. Als Vorgabe stehen in diesen Feldern 0 und 1, so dass der Messwert nicht verändert wird.

Nachkommastellen
Bestimmt die Darstellung der Zahl im Protokoll.

Messen bis Wert innerhalb der Grenzen
Die Messung wird für die angegebene Zeit wiederholt, bis der Messwert innerhalb der Grenzen liegt. Die Messung wird zusätzlich wiederholt, wenn in `Stabilität` ein Wert größer als 0 angegeben ist und die Differenz zum vorherigen Messwert größer ist als dieser Wert. Hiermit kann in einem Prüfablauf auf stabile Messwerte gewartet werden, z.B. nach einem Einschwingvorgang. Siehe auch Abschnitt 6.2.3.

Abgleichen
Diese Funktion dient zum Einstellen bzw. Abgleichen des Prüflings. Die Messung wird wiederholt, bis der Benutzer die Schaltfläche `Start` anklickt. Der Messwert wird dabei zusätzlich auf einem Abgleichbalken dargestellt.

Im Fehlerfall Testlauf aufgeben
Diese Funktion dient dazu, den Testlauf bei bestimmten Fehlerzuständen abzubrechen, wenn z.B. die Fortsetzung des Tests nicht sinnvoll ist oder um den Prüfling vor Zerstörung zu schützen.

Nur im Fehlerfall anzeigen
Wird diese Checkbox aktiviert, erscheint die Zeile nur im Messprotokoll, wenn die Bewertung FAIL ist.

Grenzen in Prüfparametertabelle
Die Grenzwerte werden aus der Spalte `Parametername` in der Prüfparametertabelle gelesen, anstatt aus dem Script. In `ID` kann eine Zeile ausgewählt werden, in die mit Klick auf `OK` die Grenzen gespeichert werden.

Der zulässige Toleranzbereich für den Abgleich wird grün dargestellt. Seine Breite kann in Prozent eingestellt werden. Darunter werden die Grenzwerte angezeigt (UG/OG).

6.2.2 Gridergebnis

Veranlasst die sofortige Bewertung der Messergebnisse und das Hochsetzen der Geprüft/-Gut/Fehler Zähler. Die Funktion eignet sich, wenn man zwei oder mehrere Prüfling in einem Ablauf prüfen möchte.

6.2.3 Mess-Schleife

Dieser Befehl findet Anwendung, wenn Sie im Modul Ergebnis die Option Messen bis innerhalb der Grenzen oder Abgleich gewählt haben. Es handelt sich hierbei um ein spezielles Label, das vom Befehl Ergebnis angesprungen wird, solange Messungen wiederholt werden müssen. Alle Zeilen, die zwischen Mess-Schleife und Ergebnis stehen, werden dabei ausgeführt, um eine kontinuierliche Messung oder Abfrage zu erreichen. Beispiel:

```
MessSchleife;
    ADX(Time:100, Detect:False, Type:1, Range:3, ADXMode:1);
Ergebnis(Max:6, Flags:4, Value:messwert, Unit:”V”, TimeOut:1000, Text ↩
    :”Spannung”, Min:4);
```

6.2.4 Report drucken

Dieser Befehl erzeugt den Ausdruck des Messprotokolls.

6.2.5 Report speichern

Dieser Befehl bewirkt, dass das Messprotokoll gespeichert wird.

Sie können auswählen, in welchem Format dies geschehen soll. Für neue Entwicklungen ist die Winguard-eigene Datenbank oder das CSV-Format empfehlenswert, die anderen Formate sind zur Kompatibilität mit bestehenden Lösungen.

6.2.6 Report zurücksetzen

Der `ClearGrid`-Befehl bewirkt, dass alle Einträge im Messprotokoll gelöscht werden. Ein bereits gespeichertes Protokoll wird nicht verändert.

Dies ist nützlich, um mehrere Protokolle in einem Prüfablauf zu erstellen. Zu diesem Zweck sollte außerdem mit `Bewertung := 1` und `FAIL_COUNTER := 0` die Gesamtbewertung zurückgesetzt werden, da ansonsten nach einem Protokoll mit einem Fehler auch alle nachfolgenden als fehlgeschlagen markiert werden.

6.3 Bilder

Die Bildauswertung beruht auf einem Vergleich mit Referenzbildern. Diese werden in Dateien gespeichert, entweder aus dem Bild-Holen-Dialog, oder mit dem Bild-Speichern-Befehl während einem gesonderten Prüfablauf zur Bilderstellung.

Während des Prüfablaufs wird zuerst ein Bild mit dem Bild-Holen-Befehl erfasst und dann mit dem Bild-Auswerten-Befehl mit einen Referenzbild verglichen. Für jedes zu erfassende Detail sollte ein separater Befehl verwendet werden, damit Übereinstimmung in einem Bereich nicht die Abweichung in einem anderen Bereich überschattet.

6.3.1 Bild Anzeigen

Mit dem Modul `Picturebox` können Sie zur Bedienerführung oder als Fehlerortanzeige Grafiken während des Testlaufs anzeigen lassen. Die Schaltfläche [...] hinter dem Feld Dateiname öffnet einen Dateiauswahldialog für die Grafiken. Unterstützt werden die Formate JPG, BMP, ICO, EMF und WMF.

Wenn kein Dateiname angegeben ist, wird das Bild im internen Puffer angezeigt.

Bildfenster schließen

Genau wie die Infobox muss auch die Picturebox mit einem eigenen Befehl geschlossen werden.

Automatische Größenanpassung
Wenn diese Option aktiv ist, wird das Fenster automatisch an die Größe der Grafik angepasst.

Im Vordergrund belassen
Mit dieser Option bleibt das Fenster immer im Vordergrund und wird von keinem anderen Fenster verdeckt

In % zur Originalgröße
Ermöglicht die Skalierung der Grafik.

Bild im Mittelpunkt des Bildschirms zentrieren
Positionierung in der Mitte.

X-/Y-Ausrichtung
Positionierung des Fensters an beliebiger Stelle.

Titelleiste anzeigen
Das Fenster wird mit der Titelleiste angezeigt und kann damit verschoben und geschlossen werden.

Statustext
Text, der hier eingegeben wird, erscheint in der Statuszeile des Fensters

6.3.2 Bild und Symbole Anzeigen

Mit diesen Befehlen können Bilder und übergelagerte Symbole angezeigt werden.

6.3.3 Bild Erfassen

Mit `ImageCapture` wird ein Bild mit einer Kamera gemacht. Die Kamera kann anhand des Namens oder Gerätepfads ausgewählt werden. Letzteres ist nützlich, falls mehrere baugleiche Kameras angeschlossen sind. Der Belichtungs-Parameter `Exposure` gibt die Verschlusszeit in Sekunden an. Der Weißabgleich-Parameter `WhiteBalance` gibt die Farbtemperatur in Kelvin an.

Das Bild wird im internen Puffer gehalten.

6.3.4 Bild Laden

Mit `ImageLoad` wird ein Bild aus einer Datei geladen. Das Bild wird im internen Puffer gehalten.

6.3.5 Bild Speichern

Mit `ImageSave` wird das Bild im internen Puffer in einer Datei gespeichert.

6.3.6 Bild Ändern

Mit `ImageChange` kann die Helligkeit des Bilds im internen Puffer normalisiert werden. Dabei wird jeder Farbkanal separat aufgehellt oder abgedunkelt. `Brightness` gibt die Soll-Helligkeit in Prozent an.

Außerdem kann das Bild in Graustufen umgewandelt werden.

Der zu verändernde Bereich kann als Array im `Rect`-Parameter angegeben werden. Das Format ist `[X, Y, Breite, Höhe]` in Pixeln von links oben.

6.3.7 Bild Auswerten

Mit `ImageEval` wird das Bild im internen Puffer mit einer Referenz aus einer Datei verglichen.

Der zu vergleichende Bereich kann als Array im `Rect`-Parameter angegeben werden. Das Format ist `[X, Y, Breite, Höhe]` in Pixeln von links oben.

Ein Versatz zwischen den beiden Bildern kann mit dem Positionstoleranz-Parameter `Displacement` ignoriert werden. Er enthält den maximalen Versatz in Pixeln. Es wird der Versatz mit der größten Übereinstimmung für die Berechnung verwendet. Dabei wird der Ausschnitt im Bild im internen Puffer verschoben.

Das Ergebnis wird in Prozent in die in `Result` angegebene Variable geschrieben.

6.4 Eingabe

6.4.1 Inputbox

Mit diesem Befehl können Sie ein Eingabefenster erzeugen, in dem der Benutzer einen Text eingeben kann. Klickt der Benutzer auf Abbrechen, wird der Testlauf abgebrochen, ansonsten wird der eingegebene Wert in der angegebenen Variablen gespeichert.

Überschrift
: Der Titel des Eingabefensters

Eingabetext
Text der Eingabeaufforderung

Defaultwert
Der Text, der am Anfang in der Eingabe steht.

Ergebnis in Variable
Gibt die Variable an, in der das Ergebnis gespeichert werden soll.

6.4.2 Dateiauswahldialog

Mit diesem Befehl kann der Benutzer aufgefordert werden, eine Datei auszuwählen, wahlweise zum lesen oder zum schreiben. Dazu wird dem Benutzer ein Standard-Dateiauswahldialog angezeigt.

Der Parameter FileName bestimmt die Variable, in der der Pfad zur gewählten Datei abgelegt wird. InitDir ist das Verzeichnis, das der Dialog zuerst anzeigt. Caption ist der Titel des Dialogs. Extension ist die Dateierweiterung (Der Teil des Dateinamens hinter dem letzten Punkt).

6.4.3 Buttonleiste einblenden

Mit diesem Modul kann während des Testlaufs eine Leiste mit Schaltflächen eingeblendet werden. Es gibt 20 Ebenen mit je 7 Schaltflächen, die vom Bediener angeklickt werden können. Durch Anklicken der Schaltflächen im Entwicklungsmodus (siehe Abb.), öffnet sich ein weiterer Dialog, mit dem die Eigenschaften eingestellt werden können.

Der gedrückte Knopf lässt sich durch Auslesen der Variable USERBUTTONS ermitteln.

6.4.3.1 Buttonleiste dynamisch ändern

```
function SetUserButtonOptions(ID: Real; Caption: string; Enabled:  ↩
    Boolean; Visible: Boolean): Boolean;
```

Diese Funktion ändert den Knopf Nr. `ID` zur Laufzeit.

6.5 Formulare anzeigen und auswerten

6.5.1 Ein Formular erstellen

Formulare können mit einem beliebigen HTML-Editor erstellt werden. Als Beispiel wird hier BlueGriffon verwendet.

Ändern sie den Zeichensatz des Dokuments auf „UTF-8“ (Format / Seiteneigenschaften).

Um den Benutzer die Möglichkeit zu geben, einen Text einzugeben, fügen sie ein Formular-Element ein (Einfügen / Formular / Formular). Geben sie einen beliebigen Formularnamen an, als URL den Dateinamen dieses Dokuments, als Methode POST, und belassen sie die übrigen Einstellungen.

Geben sie dann in das frisch erstellte Formular Text ein, um dem Benutzer mitzuteilen, was eingegeben werden soll.

Danach kommt in das Formular ein Textfeld (Einfügen / Formular / Eingaben... / Textfeld). Als Name des Textfeldes geben sie einen Winguard-Variablennamen an.

Um die Eingabe abzuschließen, braucht das Formular noch eine Schaltfläche (Einfügen / Formular / Schaltfläche) des Typs „Abschicken“.

Speichern sie das HTML-Dokument als Datei im Winguard-Projektverzeichnis.

6.5.2 Ein Formular anzeigen

Der `FormShow`-Befehl öffnet das Formular-Fenster und bestimmt das angezeigte Formular.

6.5.3 Ein Formular auswerten

Der `FormResult`-Befehl holt Benutzereingaben ab und/oder schließt das Formular. Die Benutzereingaben können entweder als Array in der angegebenen Variable abgelegt werden, oder automatisch auf mehrere Winguard-Variablen anhand des Namens der Formularelemente verteilt werden.

6.6 Guardian

6.6.1 Adapter Startkontakt

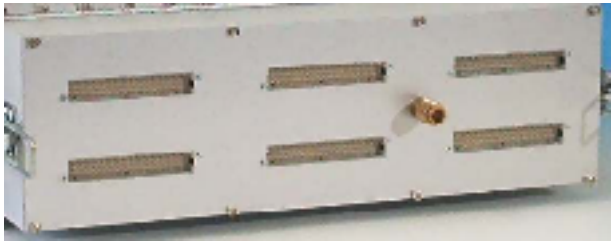


Abbildung 6.1: Guardian-Systeminterface

Winguard unterstützt das Starten eines Prüfablaufes durch ein externes Startsignal, z.B. ein Mikroschalter, der beim Schließen des Prüfadapters schaltet. Mit diesem Befehl wird ein Digitaleingang (Bit 6 des Digitalports der ersten Relaiskarte, siehe Lastrelais-Karte) gescannt. Bei einem high/low Flankenwechsel wird die Schleife beendet. Dieser Befehl hat keine Parameter.

6.6.2 ADX-Karte

Die ADX-Karte ist das interne Digitalmultimeter des Guardian Systems. Es misst potentialfrei mit einer Auflösung von 16 Bit und hat eine integrierte Scopefunktion.

6.6.2.1 Einzelmessung Multimeter

In diesem Bereich stehen folgenden Messarten zur Verfügung: Gleichspannung (DCV), Wechselspannung (ACV), Widerstand (Ohm) und Diodenmessung (DIO). Mit den Schaltflächen unter dem grünen Display wählen Sie die Messart aus. Die Schaltflächen auf der rechten Seite sind für die Messbereiche. Mit der Checkbox Autorange wird die automatische Bereichswahl aktiviert. Bei Time wird die Messzeit im Bereich von 1-1000 ms eingestellt. Grundsätzlich gilt: je länger die Messzeit, desto stabiler der Messwert.

Mit der Schaltfläche Messen können Sie Messungen direkt von dem Modul aus starten. Eine Übergabevariable, wie in den anderen Modulen, braucht hier nicht angegeben zu werden; der aktuelle Messwert wird über Systemvariable `messwert` abgefragt.

Die Einheit des Messwerts steht in der Systemvariable `adx_dim`. Dies ist hauptsächlich für die automatische Bereichswahl interessant.

6.6.2.2 Scope

Auflösung
8 oder 16 Bit

Triggerart
Positiv, negativ, Software (Freilauf)

Abtastrate
200 Hz bis 50 KHz

Triggerpegel
+/- 200mV bis 200V je nach Messbereich

Bereich
200mV, 2V, 20V oder 200V

Sweep
Länge der Aufzeichnung. Die max. Länge hängt von der Abtastrate und der Auflösung ab. Die Speichergröße ist 1024 Werte bei 8 Bit und 512 Werte bei 16 Bit.

Timeout
Zeitlimit bis zum Eintreffen der Triggerflanke (nur bei Triggerart pos und neg)

Modus
Betriebsart der Scopefunktion. Es besteht die Auswahl zwischen „Triggern“ (Starten der Aufzeichnung), „Daten lesen“ (Datenübertragung vom ADX-Modul in den Speicher) und „Triggern & Daten einlesen“.

Anmerkung
Nach dem Lesen der Daten können Sie diese mit dem Scope-Befehl auswerten.

6.6.3 RC-Messung mit ADX-Karte

Mit diesem Befehl können ein Kondensator und ein parallel geschalteter Widerstand gemessen werden. Dazu wird der Kondensator mit einer Konstantstromquelle aufgeladen, und die dabei entstehende Spannung aufgezeichnet und ausgewertet.

Die nötige Stromstärke und Aufladedauer berechnet sich aus der Kapazität des Kondensators und dem parallelen Widerstand. Diese müssen daher als Messbereich angegeben werden.

Die Einheitenprefixe für Widerstand und Kapazität werden für Messbereiche und Ergebnisse verwendet.

6.6.4 BUS-Befehle

Dieses Modul ist für die low-level Programmierung der Komponenten des Guardian Testsystems vorgesehen. Hiermit kann z.B. jedes einzelne Relais des Messstellenumschalters gezielt ein- oder ausgeschaltet werden (siehe MSU Datenblatt).

6.6.5 Guardian Hardware initialisieren

Führt eine Initialisierung der Guardian Hardware durch. Dies ist eine Sicherheitsmaßnahme, bei der alle Stromversorgungen und Relais ausgeschaltet werden. Für diesen Befehl gibt es keine weiteren Einstellungen.

6.6.6 Kurzschlusstest

Dieses Modul testet einen Prüfling auf Kurzschlüsse und unterbrochene Verbindungen. In der linken Hälfte des Fensters finden sich diverse Optionen und der Dateiname, unter dem die Einstellungen gespeichert werden.

Der untere Schwellwert legt fest, wie viel Widerstand eine Verbindung maximal haben darf, bevor sie als unterbrochen bewertet wird.

Der obere Schwellwert ist der minimale Widerstand, unter dem ein Kurzschluss festgestellt wird. Wenn nicht angegeben, sind beide Schwellwerte identisch.

Spannungskontrolle bestimmt, ob der Prüfling vor dem Kurzschlusstest auf Spannung überprüft wird. Dies ist empfehlenswert, da das Testsystem durch Spannung beschädigt werden kann. Mit der Einstellung „Entladung“ wird für die eingestellte Wartezeit versucht, die Spannung abzuleiten. Wenn dadurch die Spannung hinreichend abfällt, kann die Prüfung fortgesetzt werden.

Maximale Anzahl Fehler begrenzt die im Protokoll ausgegebenen Fehlermeldungen.

Mit einer Lastrelais-Karte kann ein Abschlusswiderstand zugeschaltet werden.

Wählen sie mit Kanäle aus, an welchen MSU-Kanälen Messpunkte angeschlossen sind.

In der rechten Hälfte wird eingestellt, zwischen welchen Messpunkten Verbindungen bestehen sollen und welche Messpunkte ausgespart werden. Diese Informationen können automatisch ermittelt werden. Drücken sie dafür den Lernen-Knopf. Dies führt einen der Prüfung ähnlichen Vorgang durch, und trägt die gefundenen Verbindungen und unter Spannung stehenden Messpunkte in die entsprechenden Felder ein.

Tragen sie in Übersprungene Verbindungen Paare von Messwerten ein, zwischen denen nicht direkt eine Widerstandsmessung durchgeführt werden soll, auch wenn ein sie an einem Kurzschluss beteiligt sind.

Während des Prüfablaufs gibt der Kurzschlusstest für jeden gefundenen Fehler (Kurzschluss, fehlende Verbindung, unerwartete Spannung) eine Ergebniszeile aus und je eine Zeile mit der Anzahl der Verbindungen und Kurzschlüsse.

6.6.7 Lastrelais-Karte

Mit diesem Modul werden die Relaiskarten des Testsystems gesteuert. Im Feld Relaiskarte wird die Nummer der gewünschten Karte eingestellt. Der Zustand eines Relais wird verändert, wenn die Checkboxen neben den virtuellen Schaltern aktiviert sind. Mit diesen Schaltern wird der Zustand der Relaiskontakte gewählt. Die Relais können zur besseren Übersicht mit Namen versehen werden. Die Namen können auch mit der Schaltfläche Import aus den INI-Dateien von anderen Projekten übernommen werden. Die Relais können auch über Variablenwerte geschaltet werden, indem Sie die Relaisnummer (1-18) und den Schaltzustand (0 oder 1) angeben. Die Relaiskarten haben neben den Relais zusätzlich einen 8 Bit Digital Port (siehe Datenblatt). Die Bits 0-5 werden normalerweise zum Lesen der Adapternummer (0-31) verwendet. Der Status dieses Ports kann durch Eingabe einer Variable gelesen werden.

6.6.8 PIO-Karte

Bei auf High geschalteten Ports sind beide Anschlüsse des Ports miteinander verbunden. Optional können sie außerdem mit dem internen 5V-Netzteil verbunden werden.

Um die Frequenzgeneratorfunktion zu nutzen, tragen Sie die gewünschten Parameter ein. Bei High-Schaltend schaltet der gewählte Ausgang zwischen High und Tristate um, bei Low-Schaltend zwischen Low und Tristate, und bei Push-Pull zwischen High und Low. Die Dauer des zuerst genannten Zustands ist die Impulslänge. Früher eingestellte Parameter werden durch die neuen ersetzt, es kann immer nur eine Frequenz erzeugt werden. Die Generatorfunktion bleibt aktiv bis sie explizit abgeschaltet wird oder die gewählte Anzahl Impulse erzeugt wurden.

6.6.9 PLD-Karte

Auf der linken Seite des Dialogs befinden sich zwei Spalten mit Bedienelementen für jeden Ausgang der Karte. In die Textfelder können Beschriftungen für die Ausgänge eingetragen werden. Mit Checkboxen wird ausgewählt, welche Ausgänge bearbeitet und welche unverändert gelassen werden. Unten befinden sich Textfelder mit einer numerischen Repräsentation, die benutzt werden können um von Programmcode berechnete Werte zu verwenden. Beispielsweise könnte eine Schleife die Zweierpotenzen von 1 bis 128 in einer Variable ablegen um acht Ausgänge hintereinander zu schalten.

Um einen Ausgang an- oder abzuschalten, versehen sie den Ausgang mit einem Häkchen und stellen den Schalter für den Ausgang in die gewünschte Stellung. Alle Ausgänge ohne Häkchen werden unverändert belassen.

Um die Frequenzgeneratorfunktion zu nutzen, tragen sie die gewünschte Frequenz und Impuls- zu Pausenzeitenverhältnis ein. Alle mit Häkchen versehenen Ausgänge werden mit der gewählten Frequenz angesteuert. Ausgänge, deren Schalter auf Aus steht werden invertiert. Das normale an- und abschalten ist in diesem Modus nicht verfügbar, dafür kann ein separater Aufruf der Funktion verwendet werden. Die Generatorfunktion bleibt aktiv bis eine andere Frequenz eingestellt oder sie explizit abgeschaltet wird oder die gewählte Anzahl Impulse erzeugt wurden.

Die PLD-Karte besitzt einen Überstromschutz. Wenn die gewählte Stromstärke überschritten wird, wird ein Flag gesetzt dass mit Hilfe des Rückgabeparameters abgefragt werden kann.

6.6.10 MSU-Karte

Mit diesem Modul werden die MSU-Karten (Messstellenumschalter) des Testsystems gesteuert. Die MSU-Karten sind mit je 48 Reedrelais bestückt und stellen die Verbindung von einer Messstelle zu einer Messeinrichtung über den Analogbus her. Jede Messstelle ist mit zwei MSU-Karten verbunden, mit vier Karten können also 96 Messstellen bedient werden etc. Auf diese Weise kann eine Messung zwischen beliebigen Messstellen erfolgen.

Um einen Messkanal einzuschalten, doppelklicken Sie auf das entsprechende Feld. Es öffnet sich ein Eigenschaftsfenster, in dem Sie dem Kanal einen Namen zuordnen können. Desweiteren können Sie wählen, ob der Messpunkt mit dem Plus- oder Minuseingang der Messeinrichtung verbunden werden soll.

Warnung
Achten Sie unbedingt darauf, dass über die eingeschalteten Reedrelais kein Strom von mehr als 500mA fließt, da sonst die Relaiskontakte zerstört werden. Dies kann auch bei der Entladung von Kondensatoren passieren und lässt sich z.B. durch Strombegrenzungswiderstände vermeiden (siehe Datenblatt).

Bei Ausführung dieses Befehls werden zuerst alle Verbindungen der MSU-Karten getrennt, um Kurzschlüsse zu vermeiden. Danach wird jeweils eine MSU-Karte mit den Sense-Analogbus-Kanälen verbunden, und mit der Option ”4 Pol” außerdem mit den Drive-Analogbus-Kanälen. Dies ist so gut wie immer die richtige Wahl.

Danach wird jede der beiden Karten mit einer Messstelle verbunden, so dass eine Messstelle mit Kanal A und eine mit Kanal B verbunden ist. Optional können dann weitere Messstellen mit Kanal A verbunden werden. Dazu dient die ”Einspeisung”-Option.

6.6.11 PSU

Mit diesem Befehl werden die programmierbaren Stromversorgungen (oder auch Power Supply Units = PSU) des Testsystems gesteuert. Die Ausgangsspannung und die Strombegrenzung werden in den entsprechenden Feldern als Wert oder Variable angegeben. Mit dem Ein/Aus Schalter wird das Relais am Ausgang des Netzteils gesteuert. Mit dem Befehl PSU U/I Messung kann der angegebene Strom gemessen werden. Neben dem normalen DC Betrieb bietet der AC-Modus die Ausgabe einer 10–100Hz Sinus Schwingung. In dieser Betriebsart ist das Messen von Spannung oder Strom (U/I Messung) nicht möglich. Technische Daten entnehmen Sie bitte dem Datenblatt.

Wenn das Testsystem eine Trafo-Karte hat, wird diese ebenfalls gesteuert. Dabei wird die PSU-Karte so eingestellt, dass nach der Transformation in etwa die gewünschte Spannung ausgegeben wird. Prinzipbedingt muss diese jedoch nachgeregelt werden. Beispiel:

```
PowerSupply(PowerOn:True, Nr:1, AC:True, Current:2.5, Ratio:2,  ↩
    Voltage:soll);
Netzteilmessung(Result:ist, Voltage:True, Nr:1, AC:True, Trafo:True);
PowerSupply(PowerOn:True, Nr:1, AC:True, Current:2.5, Ratio:2,  ↩
    Voltage:soll * soll / ist);
```

Anmerkung
Der Trafo ist nur für 50–60Hz spezifiziert. Die Strombegrenzung bezieht sich immer auf den Ausgang der PSU-Karte. In den meisten Fällen sollte deshalb 2.5A angegeben werden. Winguard begrenzt den Strom auf den für den Transformator zugelassen Wert.

Wenn das Testsystem mit der Erweiterung für PSU-Synchronisation ausgestattet ist, können mehrere PSU Wechselspannung synchron ausgeben. Dazu müssen zuerst ein oder mehrere Taktnehmer eingestellt werden, und dann der Taktgeber. Bevor der Taktgeber eingestellt ist, liefern die Taktnehmer noch keinen Strom. Eine Nachregelung muss deshalb erfolgen, nachdem alle beteiligten PSU-Karten eingeschaltet wurden.

6.6.12 PSU U/I-Messung

Dieses Modul misst die Spannung oder den abgegebenen Strom der programmierbaren Netzteile mit einer Auflösung von 16 Bit. Im Feld Netzteilnummer wird das gewünschte Netzteil ausgewählt. Der gemessene Wert wird in der angegebenen Variable und in der Variable `messwert` abgelegt.

Wenn das Testsystem eine Trafo-Karte hat, kann dieser Befehl eine ADX-Karte verwenden, um den transformierten Strom zu messen. Die Messstellen werden dafür automatisch abgeschaltet, da sie die gleiche Verbindung zur ADX-Karte verwenden.

Das Ergebnis ist immer in Volt bzw. Ampere.

6.6.13 Farb-Messung

Mit diesem Befehl können an einer UMB2-Karte angeschlossene Farbsensoren ausgewertet werden.

Als ersten Schritt klicken Sie auf Messen, und dann auf Angeschlossene Aktivieren. Damit wird der Befehl so konfiguriert, dass alle angeschlossenen Sensoren ausgelesen werden.

Die ausgelesenen Farbwerte werden wenn angegeben in die Ergebnisvariable geschrieben. Es muss eine Real-Array-Variable angegeben werden. Für jeden Farbsensor werden vier Werte im Array abgelegt - Helligkeit, Rot, Grün und Blau.

Zusätzlich können die ausgelesenen Werte sofort ausgewertet werden. Normalerweise wird das Ergebnis in einzelnen Ergebniszeilen im Protokoll ausgegeben. Alternativ kann eine Bits-Ergebnisvariable angegeben werden, in die das Ergebnis als Bitmaske geschrieben wird.

6.6.13.1 Dunkelvergleich

Hiermit kann überprüft werden, dass eine Lichtquelle ausgeschaltet ist.

Dazu muss unter Helligkeit für jeden Sensor mit dieser Aktion eine maximal zulässige Lichtstärke angegeben werden.

Es wird dann überprüft, dass die gemessene Lichtstärke unter der Grenze liegt.

6.6.13.2 Sollwertvergleich

Hiermit kann überprüft werden, dass eine Lichtquelle mit der richtigen Farbe und Helligkeit leuchtet.

Für alle Sensoren mit dieser Aktion muss unter Helligkeit eine minimal erforderliche Lichtstärke, unter Soll-Farbe der gewünschte Farbton und unter Toleranz eine Farbabweichung (in Prozent) angegeben werden.

Es wird dann überprüft, dass die Lichtquelle die geforderte Lichtstärke erreicht und den richtigen Farbton hat.

Die Soll-Farbe kann mit Messen, Farbe Übernehmen von einer Referenzmessung übernommen werden.

6.6.13.3 Umgebungslichtausgleich

Diese Option ist nützlich, falls die Lichtquellen nicht ausreichend vom Umgebungslicht abgeschirmt sind.

Führen Sie am Anfang des Prüfablaufs vorm Anschalten der Lichtquellen eine Farb-Messung durch, bei der Sie die Sensorwerte in einer separaten Ergebnisvariable speichern. Diese Ergebnisvariable geben sie dann bei weiteren Farbmessungen im Feld Umgebungslichtausgleich an. Die gespeicherten Sensorwerte werden dann vom Messwert abgezogen.

6.6.14 DA-Wandler-Karte

Die DA-Wandler-Karte kann auf 8 Kanälen verschiedene Spannungen ausgeben.

6.6.15 WFG-Karte

Die WFG-Karte erzeugt Sinus-, Dreieck- und Rechtecksignale mit einem optionalen Gleichspannungsanteil.

6.6.16 Sinusgenerator

Der Sinusgenerator wurde zur Prüfung von Audiokomponenten konzipiert. Er erzeugt eine potentialfreie Sinusschwingung mit sehr geringem Klirrfaktor. Die Frequenz ist im Bereich von 0,01 Hz bis 20 KHz und die Amplitude von 0 bis 20 Vss programmierbar.

6.6.17 Relaismatrix

Mit diesem Modul werden die Signalwege zwischen dem Prüfling und vier externen Geräten geroutet. An der Rückseite des Guardian Systems befinden sich vier isoliert montierte BNC Buchsen, an die beliebige Messgeräte oder Quellen angeschlossen werden können. Da die in der Relaismatrix verwendeten Reedrelais nur mit max. 500mA belastet werden dürfen, liegt aus Sicherheitsgründen in jeder der acht Leitungen zu den BNC Buchsen ein Strombegrenzungswiderstand von 110 Ohm, um eine Überlastung zu vermeiden. Bei Geräten mit einem hochohmigen Eingang haben diese Widerstände keine Auswirkung. Bei Einspeisung eines Signals z.B. mit einem Funktionsgenerator kann dies möglicherweise relevant sein.

Beim ersten Anklicken eines Kreuzungspunktes zweier Linien wird ein Punkt gesetzt, beim zweiten Anklicken verwandelt sich der Punkt in ein diagonales Kreuz und beim dritten Anklicken verschwindet das Symbol wieder. Ein Punkt bedeutet, dass bei der Programmausführung der Ausgang der entsprechenden MSU-Karte mit dem Anschluss eines externen Gerätes

verbunden wird. Das Kreuz bedeutet, dass die Verbindung getrennt wird. Ohne Symbol bleibt der Status unverändert.

In die Textfelder auf der linken Seite, werden die Kanal Nummern eingetragen die geschaltet werden sollen. Durch Komma getrennt können auch mehrere Kanäle auf einer Karte gleichzeitig eingeschaltet werden. Bitte beachten Sie, dass hierdurch die Reedrelais überlastet werden können, wenn die Kanäle z.B. zwischen den Plus- und Minuspol einer Stromversorgung liegen! Zum Ausschalten eines einzelnen Kanals geben Sie die entsprechende Nummer mit negativem Vorzeichen an.

6.6.17.1 Zurücksetzen

Vor dem Einschalten von Relais können Sie in dem oberen Panel die Matrix bzw. Teile der Matrix zurückzusetzen, d.h. die entsprechenden Relais ausschalten. Nach dem letzten Ausschaltbefehl wird eine Pause von 2ms eingehalten, damit die Relais ausreichend Zeit zum Öffnen haben und es nicht zu einer Überlappung mit neuen Verbindungen kommt. Dies gilt auch für die negativ angegebenen Kanalnummern.

Alle Kanäle
Die Kanäle 1-48 auf allen Karten werden ausgeschaltet

Analogbus
Alle Verbindungen zu den externen Geräten werden getrennt

Gesamte Matrix
Alle Relais der gesamten Matrix werden ausgeschaltet. Dieser Befehl wird am schnellsten ausgeführt!

ADX Karte trennen
Hiermit werden die Relais des internen Multimeters abgeschaltet.

6.6.18 UMB1-Karte

UMB steht für Universal Microcontroller Board. Diese Systemkomponente wurde in zwei Bestückungsvarianten hergestellt: Als reine Digital I/O Karte und als Multifunktionskarte mit digitalen und analogen Funktionen. Technische Daten entnehmen Sie bitte dem Datenblatt. Im Feld Kartennr. wird die Karte adressiert. Mit der Schaltfläche Test kann die Funktion aus dem Modul heraus direkt ausgeführt werden.

6.6.18.1 Registerkarte Digital Port

Hiermit steuert das Modul den Zugriff auf die 32 digitalen Kanäle. Wählen Sie bei Richtung aus, ob gelesen oder geschrieben werden soll. Bei Modus wird die Art des Zugriffs gewählt:

Bit

Zum Selektieren I/O Kanäle klicken Sie bei gehaltener Alt-Taste auf die gewünschen Bits. Diese färben sich rot und erscheinen unten in der Auswahl (`P0_1 = Port + Bit-Nr`). Auf diese Weise können Sie sich Gruppen von beliebigen Bits zusammen stellen, auf die mit einem max. 32 Bit breiten Datenwort zugegriffen wird. Die Reihenfolge bestimmt die Stelle in dem binären Datenwort (MSB zuerst).

Byte

Zum Selektieren Ports klicken Sie bei gehaltener Alt-Taste auf die entsprechenden Portflächen. Diese färben sich rot und Erscheinen unten in der Auswahl als Portnummer. Auch hier bestimmt die Reihenfolge die Wertigkeit der Ports.

6.6.18.2 Beispiele

Die Auswahl zeigt im Bytemodus den Wert `P0+P1` an und sie geben den Wert `384` (`128+256`) aus. Auf Port 0 wird dann das Bit 0 gesetzt und auf Port 1 das Bit 7.

Die Auswahl zeigt im Bitmodus den Wert `P0_7+P0_6+P0_1+P0_0` (also die ersten und die letzten beiden Bits von Port 0). Wenn Sie den Wert 6 (binär 0110) schreiben wird Bit 6 und Bit 1 des Ports 0 gesetzt, Bit 0 und Bit 7 bleiben low.

6.6.18.3 Übergabeparameter

Beim bitweisen Schreiben können Sie die einzelnen Bits durch Doppelklick invertieren. Das entsprechende Datenwort wird im Bereich Data hexadezimal und dezimal dargestellt. Bei Eingabe einer Variablen in das Feld Übergabeparameter wird der Wert der Variablen ausgegeben. Beim Lesen wird der Zustand der Bits in der hier eingegebenen Variable abgelegt.

6.6.18.4 Registerkarte Generator

Mit dieser Funktion können Sie an den digitalen Kanälen ein Taktsignal mit einstellbarem Tastverhältnis ausgeben. Das Bit, auf dem der Takt ausgegeben werden soll, ist im Bereich Ausgabe Bit einstellbar. Die Taktfrequenz ist im Bereich von 10 Hz bis 4000 Hz programmierbar. Wenn das Tastverhältnis 50:50 beträgt, können Sie einen zweiten Takt auf einem anderen Bit ausgeben lassen.

6.6.18.5 Registerkarte Analog

Die vollbestückte UMB-Karte verfügt über einen 12 Bit A/D Wandler mit achtfach Multiplexer. Der EingangsSpannungsbereich ist in vier Stufen umschaltbar: ±2,5V, ±5V, 0–5V und 0–10V. Der D/A Wandler hat ebenfalls eine Auflösung von 12 Bit und kann zwischen ±2V und 0–4V umgeschaltet werden. Diese Spanne von 4V kann bei Bedarf hardwaremäßig bis 10V erweitert werden (siehe Datenblatt).

Wählen Sie zunächst mit den drei Radiobuttons auf der linken Seite die gewünschte Betriebsart aus. Im Bereich D/A Wandler geben Sie gewünschte Spannung im Feld Spannung als Konstante oder Variable an. Im Bereich A/D Wandler können Sie neben der Variablen, mit der der Analogwert zurückgegeben wird, auch die Messzeit einstellen. Innerhalb dieses Zeitintervalls wird fortlaufend gemessen und der Mittelwert gebildet.

6.6.18.6 Encoder

Die UMB-Karte hat zwei Interrupt Eingänge, an die ein Encoder (Weg- bzw. Positionsgeber) mit zwei um 90 Grad versetzen Ausgängen angeschlossen werden kann. Die Flankenwechsel werden in einem 16 Bit Positionsregister gezählt. Es werden Frequenzen bis ca. 20 KHz verarbeitet. Im Bereich A/D Wandler gibt es eine Checkbox mit der Bezeichnung Encoder als Kanal. Ist diese Funktion aktiviert, wird anstelle eines Analogwertes das Positionsregister ausgelesen. In der unten beschriebenen Kalibrierdatei kann ein Faktor hinterlegt werden, um diesen Wert z.B. in mm umzurechnen.

6.6.18.7 4 Kanal-Scope

Diese Funktion arbeitet in drei Schritten, die bei Modus gewählt werden:

Auslösen
: Diese Funktion zeichnet vier aufeinander folgende Analogkanäle auf. Der erste dieser Kanäle 0 bis 4 (für Kanal 0–3, 1–4, … 4–7) ist einstellbar. Die Anzahl der Messwerte ist von 16–4080 und die Abtastrate von 1–5000 Hz einstellbar. Die Daten werden zunächst in dem 32 KByte großen Speicher der Karte abgelegt.

Messwerte lesen
: Diese Funktion überträgt die Daten aus dem Speicher der UMB-Karte über die RS-232 Schnittstelle zum PC. Die Übertragung des gesamten Speichers dauert mehrere Sekunden. Wählen Sie die Anzahl der Messwerte daher nur so groß wie nötig.

Messwerte zuordnen
Hier werden die vier Kurven in Winguard Arrays umgewandelt. Geben Sie in den Feldern jeweils ein Real-Array an. Mit der Scope-Funktion können die Kurven grafisch dargestellt und ausgewertet werden.

6.6.18.8 Kalibrierung

Der Treiber für die UMB-Karte verwendet die Kalibrierdatei `UMBAnalog.kal` (im Winguard Programmverzeichnis), um die Ein- und Ausgangssignale zu skalieren. Das Feld auf der rechten Seite zeigt den Aufbau der Datei. Verwenden Sie einen beliebigen Editor, um die Faktoren einzugeben. Um diese Funktion zu aktivieren, klicken Sie auf die entsprechenden Checkboxen.

6.6.19 UMB/UMC GPIO

Die UMC-Karte hat 32 digitale Allzweckeingabe/-ausgabe-Anschlüsse. Die UMB-Karte hat 16 digitale Allzweckeingabe/-ausgabe-Anschlüsse. Sie können unabhängig voneinander konfiguriert und beschaltet werden. Um fest vorgegebene Werte auszugeben, kann für jeden Anschluss der Zustand gewählt werden. „--“ bedeutet, dass der Anschluss den vorigen Zustand beibehält.

Um einen programmatisch gewählten Zustand auszugeben, können in die Felder `Maske`, `Wert` und `Open Drain` Variablen eingetragen werden. Beispielsweise kann mit `Maske = 2` und in `Wert` eine Variable, die `0` oder `2` enthält, der zweite Anschluss umgeschaltet werden.

Beim Auslesen wird der Zustand aller 32 Anschlüsse in einer Bitmaske in die gewählte Variable geschrieben.

Die Zustände haben folgende Bedeutung:

Pull Low
Der Ausgang wird mit Masse verbunden.

Push High
Der Ausgang wird mit 5V (UMC) bzw. 3,3V (UMB) verbunden.

High Impedance
Der Ausgang wird nicht verbunden. Bei der UMB funktioniert in diesem Zustand auch der digitale Eingang nicht.

Digital Input
Der Ausgang wird über einen ca. 6kΩ (UMC) bzw. ca. 100kΩ (UMB) Widerstand mit 5V (UMC) bzw. 3,3V (UMB) verbunden. Bei der UMC ist dies nicht der bevorzugte Zustand für digitale Eingänge.

6.6.20 UMC Scope

Die UMC-Karte hat vier Analog-Eingänge, die mit maximal 15kHz (4 Kanäle) bis 21kHz (1 Kanal) aufgezeichnet werden können.

Die Daten werden beim Auslesen in das Scope übertragen und können dort ausgewertet werden.

Wenn kein Kanal als Trigger ausgewählt ist, wird sofort mit der Aufzeichnung begonnen.

6.6.21 Widerstandsgeber

Die Widerstandsgeber-Karte ist eine programmierbare Dekade. Die gewünschten Widerstandswerte werden eingestellt, indem 24 Relaiskontakte eine Reihenschaltung von binär gestaffelten Widerständen (1, 2, 4, 8, 16 Ω usw.) überbrücken. Durch das Schalten der Kontakte können Widerstandswerte im Bereich von 0 Ω bis 16 MΩ eingestellt werden. Die Strombelastbarkeit und Spannungsfestigkeit hängen von der Kombination der aktiven Widerstände ab. Die einzelnen Widerstände sind Metallschichtwiderstände mit einer Genauigkeit von ±1%, einer max. Leistung von 0.6W. Der Temperaturkoeffizient ist ±50 ppm/°C.

Bei Werten bis 63 Ω wird der unerwünschte Übergangswiderstand der Relaiskontakte durch einen zuschaltbaren Abgriff klein gehalten. Der programmierte Widerstandswert kann wahlweise an den Analogbus 3+4 des Guardian Systems oder an X2, Pin 1+2 geschaltet werden. X2, Pin 3 ist mit SYSGND verbunden, um ggf. den Schirm einer Leitung anzuschließen.

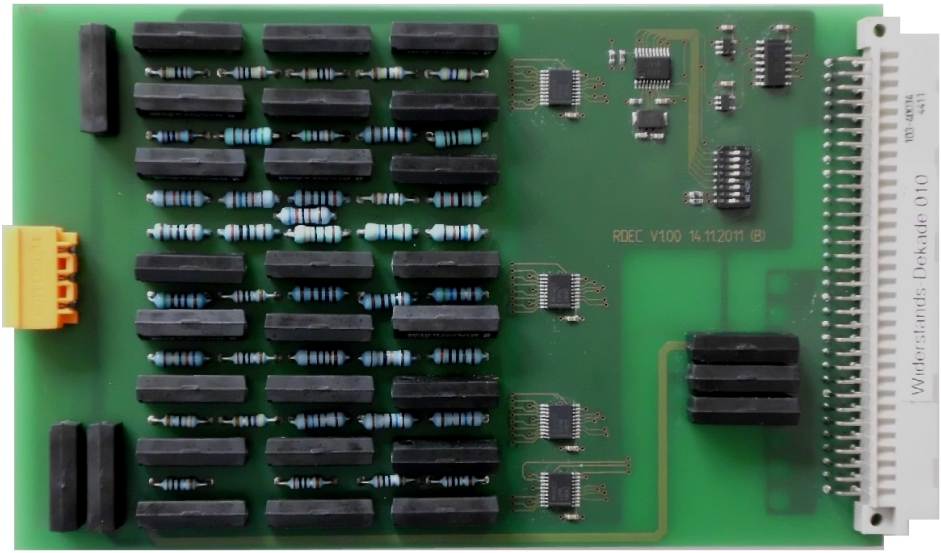


Abbildung 6.2: Widerstandsgeber-Karte, links X2

6.7 Kommunikation

6.7.1 Comport: Eigenschaften

Über das Comport Modul öffnen, konfigurieren und schließen Sie die seriellen Schnittstellen (RS-232) Ihres PCs. Der Comport, an dem sich das Guardian Testsystem befindet, steht für den Prüfablauf nicht zur Verfügung und wird von Winguard automatisch verwaltet.

Die Checkbox Parityumschaltung ohne Initialisierung ermöglicht das Ändern des Parity-Wertes, ohne den Comport zu schließen und erneut zu öffnen. Diese Funktion wird wesentlich schneller ausgeführt und häufig bei der Ansteuerung von Guardian Komponenten verwendet.

Anmerkung
Der zu verwendende Comport wird durch einen Zahlenwert (angefangen bei dem Wert 0 für COM1) bestimmt. Es wird empfohlen im Feld Port, anstelle von Zahlenwerten globale Variable zu verwenden und diesen an einer zentralen Stelle einen Wert zuzuweisen. Auf diese Weise sind die Programme einfacher auf andere PCs portierbar.

6.7.2 Comport: Empfangen

Erlaubt das Lesen von Daten von der seriellen Schnittstelle in eine Variable. Sie müssen die Schnittstelle zuvor mit dem Befehl Comport Eigenschaften geöffnet haben.

PORT
Die Nummer des Comports, beginnend mit 0 = COM1.

Eingang/Ausgang löschen
Löscht den Empfangs- bzw. Sendepuffer.

Inhalt in Variable
Hier wird eine Stringvariable angegeben.

Anzahl der zu lesenden Zeichen
Anzahl der zu lesenden Zeichen in Bytes.

Timeout
Die Operation bricht ab, wenn die hier angegebene Zeit überschritten ist. Die Variable `RXDTIMEOUT` wird dann auf „1“ gesetzt, und auf „0“ wenn die Operation fertiggestellt wurde.

Leseform

DCD/DSR/CTS/RI
liest das Statusregister des Comport und gibt die Signale als Bitkombination zurück.

String
liest die seriellen Daten des Comports

Warten auf Antwort
Wenn angegeben, wird nicht eine bestimmte Anzahl Bytes gelesen, sondern solange gelesen bis die angegebenen Zeichen empfangen wurden. Um auf CR + LF zu warten, kann `[13, 10]` verwendet werden. Die angegebenen Zeichen werden **nicht** in den Ergebnisstring übernommen.

6.7.3 Comport: Senden

Dieses Modul gibt Daten auf der seriellen Schnittstelle aus.

Da nach dem Senden von Daten oftmals auf eine Antwort gewartet werden muss, wurde diese Funktion gleich in dieses Modul integriert. Sie müssen die Schnittstelle zuvor mit dem Befehl Comport Eigenschaften geöffnet haben.

PORT
Die Nummer der Schnittstelle.

Ein-/Ausgang löschen
Löscht den Empfangs- bzw. Sendepuffer.

String
Stringkonstante oder Stringvariable, deren Inhalt auf die Schnittstelle ausgegeben werden soll. Jedes Zeichen des Strings wird als ein windows-1252-kodiertes Byte gesendet.

`+ CR` und `+ LF`
Hängt an den zu sendenden Text ein Carrige return (ASCII 13) und/oder ein Linefeed (ASCII 10) an.

Warten auf Antwort
Zeichen oder Zeichenkette, auf die gewartet werden soll

Timeout
Zeitvorgabe in ms, in der die Antwort eintreffen soll. Bei einer Zeitüberschreitung wird der Vorgang abgebrochen und die Systemvariable `RXDTIMEOUT` wird auf den Wert 1 gesetzt.

6.7.4 TCP

Dieser Befehl kann eine einfache Anfrage-Antwort-Kommunikation (beispielsweise HTTP) mit einem anderen Rechner über TCP durchführen.

6.7.5 UMB/UMC I²C

Mit diesem Befehl können per I²C angeschlossene Geräte angesteuert werden.

Die Adresse des Gerätes muss ohne das Lese-/Schreib-Bit angegeben werden.

Die zu sendenden Daten müssen als `Array of Real` angegeben werden. Es kann ein Array-Literal (`[1, 2, 3]`) verwendet werden. Sollen nur Daten gelesen werden, muss ein leeres Array (`[]`) angegeben werden.

Sollen nur Daten geschrieben werden, muss als Anzahl der zu lesenden Bytes `0` angegeben werden.

Wenn sowohl zu sendende Daten als auch zu lesende Daten angegeben sind, wird zuerst ein Schreib-Paket gesendet, und dann ohne Stop-Zustand direkt ein Lese-Paket angefordert. Ansonsten wird ein Paket gesendet bzw. angefordert.

In der Schreib-Ergebnisvariable wird die Anzahl tatsächlich geschriebener Bytes zurückgegeben. Damit kann überprüft werden, ob der Schreibvorgang erfolgreich ausgeführt wurde.

In der Lese-Ergebnisvariable werden die gelesenen Bytes als `Array of Real` zurückgegeben.

UMB-Karten mit Firmwareversion vor 1.12 unterstützen nur den Fast-Mode. Ab Firmwareversion 1.12 wird auch der Standard-Mode mit 100 kbit/s unterstützt.

6.7.6 UMB/UMC SPI

Mit diesem Befehl kann von einem SPI-Bus gelesen und geschrieben werden.

Nur die UMB-Karte unterstützt den 4-Wire-Slave-Modus, und nur die UMC-Karte unterstützt mehrere Chip-Select-Leitungen - bei der UMB-Karte muss stattdessen der 3-Wire-Modus und der UMB-GPIO-Befehl verwendet werden.

6.8 Programmablauf

6.8.1 case of

Eine Case-Anweisung besteht aus zwei Teilen, dem Anweisungskopf mit dem Selektor (`case Ausdruck of`) und dem Verzweigungsteil, in dem die einzelnen Werte behandelt werden (`select Wert:`). In Abhängigkeit von dem Wert des Selektors springt der Interpreter zu den verschiedenen Verzweigungen. Die letzte Verzweigung kann ein `else` sein, sie wird immer ausgeführt wenn kein voriger Selektor passte. Die Case-Anweisung wird mit `end case` abgeschlossen.

Sie können eine Case Anweisung z.B. auch verwenden, um abzufragen, welche Schaltfläche der Benutzer bei Verwendung des „UserButton“-Befehls angeklickt hat.

```
case UserButtons of
    select ”1x”:
        TextOut(Text: ”Eins”);
    select ”2x”:
        TextOut(Text: ”Zwei”);
    ...
end case;
```

Anmerkung
Im Gegensatz zu anderen Sprachen, erlaubt Winguard die Eingabe von Ausdrücken in den Selektoren. Es wird jedoch immer nur der erste passende Zweig ausgeführt.

6.8.2 if then else

Mit diesem Befehl werden andere Befehle abhängig von einer Bedingung ausgeführt oder nicht ausgeführt. Die Befehlszeilen unter `if Bedingung then` werden ausgeführt, wenn die Bedingung zutrifft, andernfalls wird der Bereich unterhalb von `else` ausgeführt. Die Zeile `else` ist optional und kann gelöscht werden, wenn keine Befehle ausgeführt werden sollen, wenn die Bedingung nicht zutrifft. `end if` beendet die Verzweigung.

Eine Bedingung ist erfüllt, wenn sie einen wahrer logischer Ausdruck, ein nicht-leerer String oder eine Zahl ungleich 0 ist.

```
if True then            // immer erfüllt.
if "" then              // nie erfüllt.
if messwert > 10 then   // Wenn der in 'Messwert' gespeicherte Wert ↩
    größer als
                        // 10 ist, ist die Bedingung erfüllt
if b = 1 then           // Die Bedingung ist erfüllt, wenn 'b' den ↩
    Wert 1 hat
if b - 1 then           // Die Bedingung ist erfüllt, wenn 'b' einen ↩
    Wert
                        // ungleich 1 hat
```

Steht zwischen `if` … `then` und `end if` die Anweisung `else`, so werden alle Zeilen über `else` ausgeführt, falls die Bedingung erfüllt ist, alle darunterstehenden falls die Bedingung nicht erfüllt ist.

```
if messwert > 10 then
    TextOut(Text: "Messwert ist größer als 10!");
else
    TextOut(Text: "Messwert ist zu klein!");
end if
```

6.8.3 Schleifen

Es werden drei verschiedene Arten von Schleifen unterstützt: For-Schleife, While-Do-Schleife und Repeat-Until-Schleife.

6.8.3.1 For-Schleife

Eine For-Schleife wird verwendet um den Wert einer Zahlenvariablen hochzuzählen. Die Variable muss zuvor deklariert worden sein. Eine For-Schleife hat die Form `for Variable :=`

`Startwert to Abbruchwert do`, der auszuführende Block wird mit `end for` abgeschlossen.

Bei Ausführung wird der Variablen der Startwert zugewiesen. Dieser wird solange um eins erhöht, bis der Abbruchwert erreicht ist. Vor jeder Erhöhung wird der Schleifenrumpf oder Block ausgeführt. Der Wert der Schleifenvariable ist nach dem Durchlauf gleich `Startwert + 1`.

```
for i := 1 to 10 do
    TextOut(Text: i);
end for
```

Das Beispiel gibt zeilenweise die Zahlen 1 bis 10 aus. Der Wert von `i` ist nach dem Durchlauf 11.

6.8.3.2 While-Do-Schleife

Bei While-Do-Schleifen wird der Schleifenrumpf solange ausgeführt, solange eine gegebene Bedingung wahr (Nichtnull) ist. Sie wird in Winguard durch folgendes Konstrukt eingeleitet `while Bedingung do` und beendet mit `end while`.

```
while a < b do
    a := a + 1
    b := b / 2
end while;          // erhöhe a um 1, teile b durch 2 bis a >= b ist

while True do
end while;          // diese Schleife wird nicht verlassen (Programm ↩
    hängt!)

while False do
end while;          // diese Schleife wird nicht durchlaufen
```

6.8.3.3 Repeat-Until-Schleife

Repeat-Until-Schleifen werden mindestens einmal durchlaufen. Die Abbruchbedingung steht am Ende der Schleife, die Schleife wird solange ausgeführt, solange diese Bedingung nach falsch auswertet. Der Schleifenkopf besteht aus einem einfachen `repeat`, das Schleifenende enthält die Abbruchbedingung: `until Bedingung;`.

```
repeat
    a := a + 1
```

```
    b := b / 2
until a >= b;       // diese Schleife ist nicht identisch mit der  ↩
    obigen
                    // while-Schleife!
```

6.8.4 goto

Dieses Modul erzeugt bedingte oder unbedingte Programmsprünge. Im oberen Feld wird das Label eingegeben, zu dem gesprungen werden soll. Ist die Checkbox Bedingung angeklickt, wird der Sprung nur ausgeführt, wenn die eingegebene Bedingung zutrifft.

In den meisten Fällen sind die anderen Kontrollstrukturen besser lesbar.

6.8.5 Label

Label sind so genannte Sprungmarken. Damit der Interpreter von einer Skriptzeile zu einer anderen springen kann, muss diese durch ein Label benannt werden (vgl. goto).

6.8.6 HALT

Mit diesem Befehl wird die Ausführung des Prüfprogrammes beendet und der Testlauf ordnungsgemäß abgeschlossen. Die Zeilen in der Unit FINALIZATION werden allerdings noch ausgeführt, um Hardware ggf. zurückzusetzen.

6.8.7 Pause

Dieser Befehl fügt Wartezeiten in den Testlauf ein. Wenn Sie die Option visible aktivieren, erscheint während der Verzögerung ein Countdown in der Mitte des Bildschirms.

6.8.8 Kommentarzeile

Dieser Befehl hat keinen Effekt auf die Ausführung des Programms. Er ermöglicht Ihnen das Einfügen von Kommentaren in Ihren Quelltext, damit dieser besser verständlich ist.

6.9 Sequentieller Dateizugriff

6.9.1 Öffnen/Schließen

Mit diesem Modul werden ASCII Dateien zum Schreiben oder Lesen geöffnet.

6.9.1.1 Zugriffsart

Lesen
Die Datei wird nur zum Lesen geöffnet. Alle Schreibzugriffe auf die Datei schlagen fehl.

Überschreiben
Die Datei wird zum Schreiben geöffnet. Der alte Inhalt der Datei wird komplett gelöscht, bevor Daten geschrieben werden. Wenn die Datei nicht existiert, wird sie neu angelegt.

Anhängen
Die Datei wird zum Schreiben geöffnet oder neu angelegt, falls sie noch nicht existiert. Der alte Inhalt der Datei bleibt erhalten. Schreibzugriffe hängen Daten an.

Schließen
Die Datei wird geschlossen. Wenn eine Datei geschlossen wird, werden sämtliche Änderungen gesichert und nachfolgende Lese- oder Schreibzugriffe schlagen fehl.

Wenn Sie die Option „Prüfen ob Datei von Fremdanwendung geöffnet ist“ aktiv ist, wird mit dem Öffnen solange gewartet, bis entweder keine andere Anwendung mehr auf die Datei zugreift oder die angegebene Timeoutzeit erreicht ist. In diesem Fall wird die Datei nicht geöffnet.

6.9.2 Lesen

Dieser Befehl liest eine Zeile aus der Datei und setzt den Dateizeiger auf die nächste Zeile. Die Zeile kann mehrere Teile enthalten, die durch ein Trennzeichen getrennt sind. Die einzelnen Teile werden den Variablen, die bei Parameter angegeben sind, zugeordnet. Enthält die Zeile mehr Trennzeichen als Variablen, enthält die letzte Variable den Rest der Zeile.

6.9.3 Schreiben

Schreibt den Inhalt der Variablen als Textzeile in eine zuvor geöffnete Datei.

6.10 Sonstiges

6.10.1 EXE starten

Dieser Befehl startet eine externe Anwendung. Sie haben die Möglichkeit, an die Anwendung mehrere Parameter zu übergeben. Geben Sie an, in welchem Verzeichnis das Programm gestartet werden soll.

Wählen Sie die Option Warten bis Programm beendet wurde, damit die Ausführung des Skripts erst fortgesetzt wird, wenn das gestartete Programm beendet wurde. Mit Timeout gleich `-1` kann unbegrenzt gewartet werden.

Sie können den mit Exit Code das Ergebnis des Programms in einer Variablen speichern lassen. Falls das Programm erfolgreich beendet wurde, wird `0` geschrieben, andernfalls `ExitCode - 65536`. Wenn die Wartezeit abgelaufen ist, wird in Exit Code `-2` geschrieben.

Wenn Sie die Option Winguard während der Ausführung ausblenden aktivieren, wird Winguard minimiert, solange das andere Programm läuft. Mit Programm minimiert ausführen wird hingegen das neu gestartete Programm minimiert.

6.10.2 BEEP

Ein akustisches Signal wird über PC-Lautsprecher bzw. die Soundkarte ausgegeben. Wenn kein Dateiname angegeben ist, wird der Windows-Standard-Ton ausgegeben.

6.10.3 IniFiles

Dieses Modul ermöglicht den Zugriff auf Konfigurationsdateien im INI Format. Sie können Einträge mit verschiedenen Formaten schreiben, lesen, löschen oder auf Vorhandensein prüfen.

Konfigurationsdateien sind in einzelne Abschnitte unterteilt, so genannte Sections. In jedem Abschnitt sind mehrere Identifier und deren Werte gespeichert. Geben Sie beim Lesen eines Wertes im Feld Result eine Variable an, in die das Ergebnis gespeichert wird.

Aktion	Beschreibung	Ergebnis	Wert	Eintrag

Read	Liest den Wert, der im angegebenen Abschnitt unter dem Namen `Eintrag` gespeichert ist, aus der Datei und sichert ihn in der Variable, die in `Ergebnis` angegeben ist.	Ja	-	Ja
Write	Sichert den `Wert` in die Datei im angegebenen `Abschnitt` unter `Eintrag`.	-	Ja	Ja
Delete	Löscht den `Eintrag` oder `Abschnitt` aus der Datei.	-	-	bei Value
Section Exists	Sichert eine ”1” in der in `Ergebnis` angegebenen Variablen falls der angegebene `Abschnitt` in der Datei existiert.	Ja	-	-
Ident Exists	Sichert eine ”1”, falls im angegebenen `Abschnitt` ein `Eintrag` mit dem Namen existiert.	Ja	-	Ja

6.10.4 JSON

Liest bzw. schreibt Winguard-Script-Werte von bzw. in einen String oder Datei im JSON-Format.

Da Winguard keinen Nullwert hat, können JSON mit `null` nicht unverändert eingelesen werden. Standardmäßig gibt es eine Fehlermeldung. Mit der Option Ignoriere "null" werden Nullwerte stattdessen entfernt — bei Arrays werden die nächsten Elementn vorgerückt, bei Objekten wird der betroffene Schlüssel entfernt, und bei einer alleinstehenden `null` wird die Ergebnis-Variable nicht verändert. Mit der Option Ersetze "null" mit können Nullwerte stattdessen mit einem beliebigen anderen Wert ersetzt werden.

6.10.5 Systemutilities

Dieses Modul stellt Ihnen Funktionen zum Zugriff auf das Dateisystem zur Verfügung. Nach erfolgter Operation wird über die Variable im Ergebnisfeld eine 1 zurückgegeben. Konnte die Funktion nicht ausgeführt werden, wird eine 0 zurückgegeben.

Im Feld Dateiname wird die Datei oder das Verzeichnis eingetragen, mit dem die Operationen ausgeführt werden. Mit der gepunkteten Schaltfläche kann ein Dateiauswahldialog geöffnet werden.

FileExists
: Gibt eine 1 zurück, wenn die angegebene Datei existiert, ansonsten den Wert 0.

DeleteFile
: Löscht die angegebene Datei.

FileCopy
: Erzeugt eine Kopie einer Datei unter anderem Namen.

FileMove
: Verschiebt eine Datei.

FileSize
: Gibt die Größe der Datei in Bytes zurück.

DirectoryExists
: Prüft, ob ein Verzeichnis existiert.

CreateDir
: Erzeugt ein Verzeichnis.

File in use
Prüft, ob eine Datei in Benutzung ist.

GetFilePath
Extrahiert den Pfad aus einem Dateinamen.

6.10.6 Scope

Der Scope-Befehl dient zur Auswertung von Kurven.

6.10.6.1 Die Registerkarte Data IN

Im Bereich Input wählen Sie die Datenquelle für die Kurvendarstellung aus. Es können bis zu vier Kurven gleichzeitig dargestellt werden. Wählen Sie die Datenquellen mit den ... -Knöpfen am rechten Rand.

Alte Scopedaten löschen
Alle dargestellten Kurven werden gelöscht. Die Option sollte ausgewählt werden, wenn eine neue Kurve gezeichnet wird. Ansonsten werden aktuelle Daten an vorherige Kurvendarstellungen angehängt.

Scope anzeigen
Wenn dieser Befehl ausgeführt wird, wird ein Fenster mit den Kurven angezeigt.

Scope belassen
Das Fenster mit den Kurven wird weiterhin angezeigt, falls vorher ein Befehl mit Scope anzeigen ausgeführt wurde.

Scope ausblenden
Das Fenster mit den Kurven wird gegebenenfalls geschlossen.

Auf Taste >Weiter< warten
Nach der Anzeige einer Kurve hält das Programm an und wartet auf das Anklicken der Schaltfläche Weiter bzw. die Betätigung der ⏎-Taste. Um diesen Programmstop flexibel zu gestalten, kann er mit dem folgenden kombiniert werden:

Nur auf Taste >Weiter< warten, falls Variable [...] <> 0 ist
Um während der Entwicklungsphase eines Testprogramms zu überprüfen, ob ein Kurvenverlauf aussieht wie erwartet oder die Triggerschwellen richtig gesetzt sind, ist es sinnvoll das Programm nach einer Scopemessung stoppen zu lassen. Durch Verwendung einer globalen Variable in dieser Funktion können mehrere Stopps durch zuweisen des Wertes 0 bzw. 1 an einer zentralen Stelle gemeinsam aktiviert oder aufgehoben werden.

Text X, Text Y
Hier können Maßeinheiten für die Achsenbeschriftung eingegeben werden. Wenn Sie den Prefix Modus im Register Skalierung aktivieren, werden vor den Maßeinheiten Kennbuchstaben wie m (Milli) oder µ (Mikro) ausgegeben.

6.10.6.2 Quellenauswahl

In diesem Dialog geben Sie die Quelle für die anzuzeigenden Daten an. Folgende Datenquellen stehen Ihnen dabei zur Auswahl:

Datei
Hier können Sie X/Y Werte aus einer ASCII-Datei laden. Diese müssen paarweise, getrennt durch den angegebenen Separator, jeweils in einer Zeile vorliegen. Zum Beispiel:

```
0,005|1,2
0,006|1,3
0,007|1,4
...
```

Der erste Wert entspricht dem X-Wert, der zweite dem Y-Wert. Gibt es nur eine Spalte, werden die Daten als Y-Werte interpretiert und für die X-Werte ein Vorgabewert verwendet.

Array
Hier können Sie den Inhalt eines Real-Arrays aus Winguard darstellen. Die Werte des Arrays werden auf der Y-Achse dargestellt. Die dazugehörigen X-Werte werden durch die angegebene Frequenz ermittelt. Alternativ kann ein Array mit X-Werten angegeben werden, das genau so lang wie das für die Y-Werte ist.

Punkt
Hier können Sie einzelne Punkte in das Koordinatensystem eintragen. Jeder einzelne Punkt wird bei der Darstellung mit dem vorherigen Punkt verbunden.

ADX Aufzeichnung
Hier werden die zuletzt mit der ADX-Karte aufgezeichneten Werte übernommen.

Interner Bild-Puffer
Ein einzelner Farbkanal einer Reihe oder Spalte von Pixeln des Bildes im internen Puffer wird übernommen.

Durch Eingabe eines Wertes in `Faktor X` oder `Faktor Y` kann die Kurve optional in X bzw. Y Richtung skaliert werden. Durch Eingabe eines Wertes in `Offset X` oder `Offset Y` kann die Kurve optional in X bzw. Y Richtung verschoben werden. Die Kurve wird zuerst skaliert, dann verschoben.

6.10.6.3 Die Registerkarte Funktionen

Zur Auswertung von Kurven stehen mehrere Funktionen zur Verfügung. Die jeweils benötigten Parameter sind editierbar. In Feld Kanal wird die zu verwendende Kurve (1-4) eingestellt. Das Ergebnis einer Auswertung wird in die Variable geschrieben, die im Feld Ergebnis-Variable angegeben ist. Durch Eingabe von Werten oder Variablen in die Felder von / bis kann aus der gesamten Kurve ein Teilbereich selektiert werden.

Offset (Nulldurchgang)
Berechnet den Mittelwert einer Kurve vom ersten positiven Nulldurchgang bis zum letzten positiven Nulldurchgang.

Effektivwert
Berechnet den Effektivwert einer Kurve (Wurzel aus dem Mittelwert der Quadrate). Der Parameter Offset bezieht den angegebenen Offsetwert mit in die Berechnung ein. Mit dem Parameter Nulldurchgang berücksichtigen werden nur ganze Schwingungszüge ausgewertet. Dies erhöht die Genauigkeit, wenn eine Kurve nur wenige Schwingungszüge enthält.

Mittelwert
Berechnet den Mittelwert einer Kurve.

Klirrfaktor
Berechnet den Klirrfaktor in % (Summe der Oberwellen im Verhältnis zur Grundschwingung).

dB Sinad
Berechnet das Signal / Rauschverhältnis in dB. Hier müssen folgende Parameter angegeben werden: Abtastrate, Nutzsignal, Bandbreite, FFT-Fenster (Square, Parsen, Hanning, Welch, Hamming)

Zeitdifferenz
Berechnet die Zeitdifferenz zwischen zwei Flanken. Parameter: Y1 / Y2 Sollwerte der ersten bzw. zweiten Flanke, Flanke Y1 / Y2 (positiv / negativ / positiv Kurvenanfang / negativ Kurvenanfang): Triggerung der unter Y1 / Y2 angegebene Werte bei steigender oder fallender Flanke. Optional kann als erste Flanke bereits der Kurvenanfang genommen werden, wenn der Y1 Sollwert bereits überschritten ist.

Frequenz
Ermittelt den Kehrwert des Mittelwerts der Zeitdifferenz zwischen aufeinanderfolgenden Flanken. `Y1` gibt den Schwellwert dieser Flanken an und `Flanke Y1` die Richtung. Um Fehlmessungen durch Rauschen zu vermeiden, muss zwischen jeweils zwei Flanken eine weitere mit `Y2` und `Flanke Y2` angegebene Flanke erscheinen. Dazu sollte der Abstand zwischen `Y1` und `Y2` größer sein als die Störung durch Rauschen.

Impulszahl

Zählt die Anzahl der Flanken. Die Ermittlung der Flanken funktioniert genau wie oben bei Frequenz beschrieben.

Tastgrad

Ermittelt Flanken wie oben bei Frequenz beschrieben. Das Ergebnis ist das Verhältnis der Zeit zwischen einer „Y1"-Flanke und der folgenden „Y2"-Flanke zur Zeit zwischen zwei „Y1"-Flanken in Prozent. Dies wird auch „Duty-Cycle" genannt, und ist nützlich zum Testen einer Pulsweitenmodulation („PWM").

Peak

Sucht den kleinsten, größten oder den Unterschied zwischen dem kleinsten und größten Wert einer Kurve.

Wenn „lokale Extrema = mit" angegeben ist, wird stattdessen ein lokales Minimum oder Maximum gesucht. Es wird der Y-Wert des ersten Extrema zurückgegeben.

Die Parameter `Epsilon X` und `Y` dienen der Rauschunterdrückung. Innerhalb eines Bereiches `Epsilon X` um das lokale Extrema muss ein Wert liegen, der mehr als `Epsilon Y` vom Extrema abweicht. Die `Epsilon`-Parameter sind optional, ohne sie wird jedes Extrema gefunden. `Epsilon Y` sollte größer als die Störung durch Rauschen sein, und `Epsilon X` so groß, dass die Steigung vor oder nach dem Extrema gerade so `Epsilon Y` erreicht.

Phasenverschiebung

Berechnet die Phasenverschiebung zwischen zwei Kurven in Grad. Mit den Parametern Kurve1 / Kurve2 wird bestimmt zwischen welchen Kurven die Verschiebung berechnet werden soll.

X/Y-Abfrage

Gibt zum abgefragten X-Wert den dazugehörigen Y-Wert und umgekehrt zurück. Parameter: Kurvenwert für die Abfrage.

Differenzkurve

Erzeugt eine Kurve aus den Differenzen von aufeinander folgenden Momentanwerten. Bei Ergebnis-Variable muss hier ein Real-Array angegeben werden. Parameter: Skalierungsfaktor der einzelnen Differenzen zur Originalkurve. Wenn die Punkte ihrer Kurve alle den gleichen Abstand in X-Richtung haben (wie etwa Kurven aus einem Array oder von der ADX-Karte), können Sie durch Angabe der Frequenz die Ableitung der Kurve berechnen.

In Array kopieren

Die Kurve wird ohne weitere Bearbeitung in die Ergebnis-Variable kopiert. Diese muss vom Typ `array of Real` sein.

Hüllkurve

Die Funktion stellt fest, ob eine Kurve in einem Toleranzband liegt, d.h. es wird Wert für Wert verglichen, ob die erfasste Kurve die untere Kurve unterschreitet oder die obere Kurve überschreitet. Liegt die Messkurve im Toleranzband wird der Wert 1 in die Ergebnis-Variable geschrieben, andernfalls der Wert 0. Im Feld Hüllkurvendatei wird in Anführungszeichen der Name Hüllkurvendatei incl. Pfad oder eine Variable mit diesem Inhalt angegeben.

6.10.6.4 Erstellen von Toleranzbändern

Zeichnen Sie eine Kurve auf, ohne das Scope auszublenden, und halten Sie das Programm danach an. Klicken Sie auf die Schaltfläche Hüllkurve. Unter dem Koordinaten System wird eine Toolbox zum Erzeugen von Hüllkurvendateien eingeblendet. Wählen Sie zuerst, ob Sie die untere oder die obere Hüllkurve erzeugen wollen. Nach Anklicken der Schaltfläche Zeichnen können Sie dann mit der Maus auf die gewünschten Positionen im Koordinatensystem zeigen und mit der linken Maustaste einen Punkt setzen. Fahren Sie die nächste Position an und klicken Sie erneut. Zwischen den beiden Punkten wird jetzt eine Linie gezeichnet. Alternativ können Sie auch eine Hüllkurve mit der Funktion Punkt setzen durch Eingabe von X/Y Koordinaten erzeugen. Sind beide Toleranzbänder erstellt, sichern Sie diese mit der Schaltfläche Speichern als Datei mit der Endung HKV in das Projektverzeichnis.

Wechseln Sie in den Endwicklungsmodus und tragen Sie den Dateinamen incl. Pfad in das Feld Hüllkurvendatei ein. Durch Voranstellen der Systemvariable (z.B. PROJEKTPFAD+"Envelope.HKV") ist das Testprogramm leichter auf andere Laufwerke oder Verzeichnisse portierbar. Da die Hüllkurvendateien in einem einfachen ASCII Format vorliegen, können sie auch softwaremäßig mit Winguard oder einem anderen Programm erzeugt werden. In dem mitgelieferten Beispielskript Hüllkurve finden Sie eine entsprechende Vorlage.

Das interne Format von Hüllkurvendateien

```
[UpperCurve]
Values=10
XValue0=0
YValue0=1,174
XValue1=1
YValue1=1,348
XValue2=2

usw ...

[LowerCurve]
Values=10
XValue0=0
```

```
YValue0=-0,825
XValue1=1
YValue1=-0,651
XValue2=2
```

6.10.6.5 Die Registerkarte Farben

In diesem Bereich können Sie individuelle Farbeinstellungen zum Scope vornehmen. Damit die Einstellungen gespeichert und im Skriptablauf übernommen werden, aktivieren Sie bitte die Checkbox Farben in Befehlszeile übernehmen. Wir empfehlen, diese Einstellungen nur einmal am Anfang eines Testprogramms vorzunehmen. Damit sind sie im Nachhinein leicht änderbar.

Um Ihren Drucker nicht unnötig zu belasten, wird beim Ausdrucken der Kurven der Hintergrund des Scopes automatisch in weiß geändert. Wenn Sie für die Kurven dunklere Farben und für den Hintergrund einen hellen Farbton einstellen ist auf dem Ausdruck alles gut erkennbar.

6.10.6.6 Die Registerkarte Skalierung

In diesem Bereich stellen Sie die Skalierung (Differenz X, Differenz Y), den Offset und die Position der Achsen ein. Wählen Sie die gewünschte Funktion mit den Radiobuttons aus und stellen Sie den gewünschten Wert mit dem Drehregler ein. Wenn Sie den Drehregler anklicken, können die Werte auch mit den Cursortasten eingestellt werden. Mit der Schaltfläche Clr Offset werden die Offsetwerte auf Null gestellt.

Mit der Funktion Autoscale wird das Koordinatensystem so skaliert, dass alle angezeigten Kurven vollständig und in maximaler Größe dargestellt werden.

Der Prefix Modus erzeugt, entsprechend der gewählten Skalierung vor den Maßeinheiten, ggf. einen Kennbuchstaben wie m (Milli) oder µ (Mikro), wie das bei Oszilloskopen üblich ist.

Damit die Einstellungen gespeichert und in das Skript übernommen werden, aktivieren Sie bitte die Checkbox Skalierungswerte in Befehlszeile übernehmen. Dadurch können Sie in Ihren Testschritten unterschiedliche Skalierungen aufrufen und die Messkurven optimal darstellen.

6.10.6.7 Das Scopefenster während des Testablaufs

Falls die Option Scope anzeigen gewählt wurde, sieht der Bediener nach der Ausführung einer Scopemessung dieses Fenster. Die Kurven werden entsprechend der Einstellungen skaliert und automatisch ausgewertet. Die Triggerpunkte werden zur Kontrolle mit roten

Kreuzen markiert. Das Ergebnis der gewählten Funktion wird neben Messwert angezeigt und der eingetragenen Variablen zugewiesen. Die Skalierung kann mit den Schiebereglern oder dem Mausrad geändert werden, die Position mit der linken Maustaste. Diese Einstellung bleibt solange erhalten, bis sie durch das Programm neu gesetzt wird (siehe Checkbox Skalierungswerte in Befehlszeile übernehmen). Außerdem kann die Intensität des Rasters eingestellt werden. Mit den oberen Schaltflächen mit den Kurvennamen können die 4 Kurven ein- und ausgeschaltet werden.

Mit der Schaltfläche Kurvenmessung können Sie die einzelnen Punkte (Y-Werte) einer Kurve anzeigen lassen. Es erscheint dazu unter der Kurve ein Schieberegler, mit dem die Position auf der X-Achse per Maus eingestellt werden kann. Selektieren Sie die gewünschte Kurve, indem Sie auf einen der Drehregler (CH1-CH4) klicken.

Mit der Funktion Autoscale wird das Koordinatensystem so skaliert, dass alle Kurven vollständig und in maximaler Größe dargestellt werden. Die Funktion bezieht sich nur auf die eingeschalteten Kurven.

Mit der Schaltfläche Save können Sie die Kurven im ASCII Format speichern. Nach der Auswahl der Kurvennummer (1-4) erscheint ein Dateiauswahldialog. Die Dateiendung ist „*.Kur“. Das Format ist so aufgebaut, dass die Kurven von dem Modul auch wieder gelesen werden können (siehe Quellenauswahl).

Mit der Schaltfläche Print können die Kurven ausgedruckt werden. Sie haben dabei die Möglichkeit, einen Kommentar einzugeben und den Drucker zu wählen.

6.10.7 PicoScope

Liest Spannungsverläufe mit einem PicoScope.

6.10.8 PicoScope

Gibt Spannungsverläufe mit einem PicoScope aus.

Die Frequenz gibt an, wie viele der Kurven pro Sekunde ausgegeben werden.

Die Wellenform muss ein Array mit der Kurvenform in V enthalten.

Mit Wiederholungen kann die Kurve mehrmals ausgegeben werden.

Falls die ausgegebene Kurve mit demselben Pico-Scope gemessen werden soll, muss Bei nächster Pico-Scope-Messung starten verwendet werden, da eine neue Kurvenform nicht während einer Messung konfiguriert werden kann.

6.10.9 Schrittmotorsteuerung

Mit dem `MotorControl`-Modul können Motorsteuerungen von Trinamic gesteuert werden.

Es können mehrere Motorsteuerungen konfiguriert werden. Der Steuerungs-Parameter `Module` bestimmt, an welche Steuerung ein Befehl geschickt wird. Der Befehls-Parameter `Command` bestimmt, welcher TMCL-Befehl gesendet wird. Die Parameter `Axis`, `Param`, `Value` entsprechen den TMCL-Befehls-Parametern. In die in `Result` angegebene Variable wird die Antwort der Motorsteuerung geschrieben.

Die Achsenparameter next target position, actual position, snapshot position, last reference position, encoder position, maximum encoder deviation, next target speed, actual speed, maximum positioning speed, minimum speed, referencing search speed, referencing switch speed, actual acceleration, und maximum acceleration werden automatisch zwischen TMCL-Einheiten und Umdrehungen, Umdrehungen pro Sekunde und Umdrehungen pro Sekunde pro Sekunde umgerechnet. Dabei wird der Steuerungsparameter Einheiten pro Umdrehung berücksichtigt. Um beispielsweise die Position einer Achse, die mit der Motorachse über eine 2:1 Übersetzung verbunden ist, in Grad zu bestimmen, müssen 180 Einheiten pro Umdrehung eingestellt werden.

Die Option „Warten bis Position erreicht“ blockiert den Programmablauf, bis ein Position-Anfahren-Befehl die gewünschte Position erreicht hat. Wenn dabei ein Problem auftritt, wird eine Exception ausgelöst.

6.10.10 DLL.Call

Dieser Befehl dient nur der Abwärtskompatibilität mit früheren Versionen.

6.10.11 CallNamedPipe

Der `CallNamedPipe`-Befehl entspricht der gleichnamigen Windows-API-Funktion. Er dient zur Kommunikation mit darauf vorbereiteten Programmen.

6.11 Unterprogramme

6.11.1 Aufruf

Dieses Modul ruft ein zuvor deklariertes Unterprogramm mit Übergabeparametern auf. Wählen Sie im Dropdown-Menü das entsprechende Unterprogramm aus. Tragen Sie in der Zeile Variable / Wert die Parameter ein, die beim Aufruf übergeben werden sollen.

6.11.2 Exit Procedure

Verlässt das aktuelle Unterprogramm und springt zurück zu der Befehlszeile, von der der Aufruf kam. Dieser Befehl benötigt kein GUI-Modul.

6.11.3 Neu deklarieren

Prozeduren dienen der Organisation der Messprogramme. Sie fassen Programmteile unter einem Namen zusammen. Neben dem Namen der Prozedur wird hier auch Anzahl und Typ der Übergabeparameter festgelegt. Prozeduren können nicht in den Units Main, Initialization, Finalization und Globale Projektvariablen deklariert werden. Verwenden Sie ggf. eine andere Unit oder erzeugen Sie eine neue Unit (Menü Datei/neue Unit). Beim Schließen des Dialogs mit OK werden die Skriptzeilen `procedure name();` und `end procedure;` erzeugt. Fügen Sie beliebige Befehlszeilen dazwischen ein; diese werden dann als beim Aufruf der Prozedur ausgeführt.

Anmerkung
Es darf kein Code ausserhalb von Prozeduren eingefügt werden. Selbst der in den Units Main, Init und Finalisierung eingefügte Code wird intern in Prozeduren verpackt. Ausserdem können Prozeduren nicht geschachtelt werden, d.h. es kann keine Prozedur in einer anderen Prozedur definiert werden.

Einer Prozedur können beliebig viele Parameter übergeben werden. Parameter werden innerhalb der Klammern hinter dem Prozedurnamen angegeben. Eine Parameterdefinition ist immer der Form `Name : Typ`. Mehrere Parameter werden voneinander durch Semikolons getrennt. Die Parameter sind nur in der Prozedur definiert und können innerhalb dieser wie Variablen angesprochen werden.

```
// eine Prozedur mit einem Parameter namens 'Bar' vom Typ 'String'
procedure Foo(Bar: String)
end procedure;

// eine Prozedur mit zwei Parametern (durch Semikolon getrennt)
procedure Foo(Bar: String; Baz: Number);
end procedure;
```

Anmerkung
Parameternamen verdecken gleichnamige globale Variablen. Definieren Sie beispielsweise einen Parameter `messwert` so haben Sie innerhalb der Prozedur keinen Zugriff auf die globale Variable `messwert` (und damit auch nicht auf die Messergebnisse der ADX-Karte).

Neben Parametern können in einer Prozedur auch lokale Variablen definiert werden. Diese werden — wie Parameter — wie Variablen verwendet. Auch sie verdecken gleichnamige globale Variablen. Der Unterschied zu Parametern ist jedoch, dass lokale Variablen nicht vom Benutzer übergeben werden. Ihr Wert ist bei Aufruf der Prozedur undefiniert, das heisst, ihnen muss am Anfang der Prozedur ein Wert zugewiesen werden.

Lokale Variablen können momentan nur über den Einstellungsdialog einer Prozedurdefinition editiert werden.

6.12 Variablen

6.12.1 ArrayToString

Mit diesem Modul ist es möglich, die einzelnen Elemente eines String- oder Real-Arrays in einen String zu wandeln.

Der `Start`-Parameter gibt dabei den Index des ersten Elements an.

Ein Beispiel: Die ersten 5 Elemente des Real-Arrays Tab enthalten die ASCII Werte 48 bis 52. Wird das Array in dem Modul eingesetzt, liefert es einen String mit dem Inhalt `”01234”` zurück.

Weiteres Beispiel: Das Array `[”a”, ”b”, ”c”]` wird mit einem `Start`-Parameter von `1` in den String `”bc”` umgewandelt.

6.12.2 Stringbearbeitung

Dieses Modul ermöglicht die Umwandlung und Bearbeitung von Zeichenketten. Der Eingang ist ein Ausdruck vom Typ „String“. Der Ausgang ist abhängig von der verwendeten Operation. Die Befehlsbearbeitung erfolgt in Flussrichtung.

Konvertiere String in Großbuchstaben
: Wandelt die Kleinbuchstaben innerhalb des Strings in Großbuchstaben um.

Position
: Liefert die erste Position, an der die angegebene Zeichenkette in dem String gefunden wurde. Hierbei wird zwischen Groß- und Kleinschreibung unterschieden.

Länge
: Liefert die Anzahl der Zeichen des Strings.

Fragment
Schneidet einen Teil des Strings aus oder löscht ihn.

Beispiel: Ein String hat den Inhalt ”1234567890”.

Die Funktion Ausschneiden, Links, Länge 3, Pos. --- ergibt ”123”

Die Funktion Ausschneiden, Rechts, Länge 3, Pos. --- ergibt ”890”

Die Funktion Löschen, Mitte, Länge 3, Pos. 4 ergibt ”123890”

Segment
Liefert des x-ten, durch den Separator getrennten Teilstring zurück.

Umwandeln in Zahl
Wandelt einen String mit numerischem Inhalt in eine Real-Variable um.

Runden
Wandelt einen String mit numerischem Inhalt in eine Real- Variable um und rundet dabei auf ganze Zahlen.

Leerzeichen löschen
Löscht die Leerzeichen am Anfang oder am Ende des Strings.

Ersetze ab Position
Alle Zeichen ab der angegebenen Position werden durch den angegebenen String ersetzt.

Anzahl von Zeichen
Liefert die Anzahl der hier angegebenen Zeichen innerhalb des Strings

6.12.3 StringToArray

Dieses Modul schreibt die Zeichen eines Strings in die einzelnen Elemente des angegebenen Arrays. Bei Real Arrays wird das Zeichen zuvor in eine Ordinalzahl umgewandelt.

Dabei wird optional am Anfang des Arrays ein Lücke gelassen.

Beispiel: Der String ”HALLO” wird je nachdem in [”H”, ”A”, ”L”, ”L”, ”O”] oder [72, 65, 76, 76, 79] umgewandelt.

6.12.4 Zahlenkonvertierung

Binär zu Integer
Behandelt die eingegebene Zahl als Binärzahl und rechnet diese in eine Dezimalzahl um.

Integer zu Binär
Wandelt eine Dezimalzahl in eine Binärzahl um. Die Anzahl der Stellen ist wählbar. Das Ergebnis wird als String zurückgegeben.

Integer zu Hex
Wandelt eine Dezimalzahl in eine Hexzahl um. Die Anzahl der Stellen ist wählbar. Das Ergebnis wird als String zurückgegeben.

Hex zu Integer
Wandelt eine Hexzahl in eine Dezimalzahl um.

4ByteHex zu Float
Wandelt eine 4 Byte Hexzahl (FF00FF00) in eine Real-Variable

Zufallszahl von IN
Errechnet eine Zufallszahl zwischen 1 und der Eingangszahl.

String zu Ord
Liefert den ASCII-Wert des erstens Zeichens eines Strings.

ASCII zu String
Liefert das zu einem ASCII-Wert gehörende Zeichen.

Max Array Index
Liefert die Größe eines Arrays.

Σ Real Array
Liefert die Summe aller Element eines Real-Arrays

High Byte
Liefert das Highbyte (die oberen acht Bits) eines 16bit Wortes.

Low Byte
Liefert das Lowbyte (die unteren acht Bits) eines 16bit Wortes.

Runden
Rundet eine Realzahl auf x Nachkommastellen

6.12.5 Zuweisen

Mit diesem Modul können Sie einer Variablen einen Wert zuweisen. Mit einem Doppelklick kommen Sie wie bei den meisten Feldern in den Variablenbereich. Der Wert kann eine einfache Zahl, ein komplexer mathematischer Ausdruck oder auch ein String sein.

Das Ergebnis eines Vergleiches wie z.B. a>b oder x=1 ist von Typ Boolean und kennt nur die beiden Zustände TRUE und FALSE, die bei Zuweisungen automatisch in die Zahlenwerte 0 und

1 umgewandelt werden. Zwei Vergleiche können zusätzlich mit den Schlüsselwörtern AND, OR und XOR logisch verknüpft werden.

Beispiel: `Result := (Messwert > 1) and (Messwert < 2)`

Der Variable Result wird der Wert 1 zugewiesen, wenn die Variable Messwert einen Wert zwischen 1 und 2 hat. Andernfalls erhält Result den Wert 0.

Kapitel 7

Scriptsprache

In diesem Kapitel wird der Aufbau der Winguard-eigenen Scriptsprache erläutert. Seine Lektüre ist vor allem dann sinnvoll, wenn Sie den Winguard-Zeileneditor verwenden möchten. Im Gegensatz zu GUI-Modulen überprüft der Zeileneditor die Eingabe nicht auf Fehler. Die Informationen in diesem Kapitel können Ihnen helfen, Fehler zu vermeiden, oder einmal gemachte Fehler zu finden und zu beheben.

7.1 Ausdrücke

In den meisten GUI-Modulen können Sie statt konstanten Werten auch Ausdrücke angeben. Der Wert eines Ausdruckes wird von Winguard zur Laufzeit des Programmes ermittelt und kann damit beispielsweise aus bereits gemessenen Werten oder Benutzereingaben ermittelt werden. Damit wird eine weitaus höhere Flexibilität in der Programmentwicklung erreicht. Im folgenden Abschnitt finden Sie eine eingehende Erläuterung der von Winguard akzeptierten Ausdrücke.

7.1.1 Jeder Ausdruck hat einen Typ

Winguard unterscheidet Werte anhand ihres Typs. Der Typ eines Wertes bestimmt unter Anderem die Art der Berechnungen mit diesem Wert. So können zwei Zahlen `a` und `b` addiert werden, indem man den Ausdruck `a+b` schreibt, für zwei Zeichenketten `a` und `b` entspricht der Ausdruck `a+b` jedoch dem Aneinanderhängen (Konkatenation).

```
5 + 4       // entspricht dem Wert 9
”5” + ”4”   // entspricht dem Wert ”54”
```

7.1.1.1 Einfache Typen: Zahlen und Zeichenketten

Der Typ `Number` oder auch `Real` bezeichnet alle Werte, die Zahlen sind. Es findet keine Unterscheidung zwischen Integer- und Fliesspunktzahlen statt.

Der Typ `String` umfasst alle Zeichenketten.

7.1.1.2 Arraytypen

Arraytypen fassen mehrere Werte eines Typs zu einem neuen Wert zusammen. So speichert eine Variable vom Typ `array of Real` beliebig viele Zahlwerte unter einem einzigen Namen.

Auf die *n*te Stelle eines Arrays *array* kann mit `array{n}` zugegriffen werden.

Zu jedem Typ kann ein Arraytyp erzeugt werden. Winguard erkennt also den Typen `array of String` genauso wie den Typen `array of array of String`, auch wenn der letztere nicht von den Variablendialogen unterstützt wird.

7.1.1.3 Map-Typ

Der Typ `Map` speichert Werte anhand eines Strings. Beispielsweise ergibt `m{”A”}` den unter `”A”` gespeicherten Wert aus der `Map` `m`.

7.1.1.4 Bool’scher Typ

Der Typ `Boolean` kennt genau zwei Werte: `True` und `False`. Der Typ Boolean tritt nur als Ergebnis von Vergleichen auf und kann daher ebenfalls nicht im Variablendialog ausgewählt werden.

7.1.2 Konstante Werte

Die einfachste Art eines Ausdruckes ist ein konstanter Wert, also eine Zahl oder ein String. Die Art, wie Sie Werte angeben können hängt von dem Typ des Wertes ab.

7.1.2.1 Eingabe von Zahlenwerten

Winguard versteht unterschiedliche Zahlenformate. Einer Zahlenvariable `x` können Sie den Wert `120` einfach durch deren Angabe zuweisen:

```
x := 120
```

Enthält der Zahlenwert einen Nachkommaanteil so wird dieser durch einen Punkt vom Ganzzahlteil abgetrennt:

```
x := 120.5
x := -22.7
```

Besonders große Zahlen können in Exponentialschreibweise kürzer angegeben werden:

```
x := 1E9        // entspricht der Zahl 1000000000 (9 Nullen)
x := 1205E-1    // entspricht 120.5
```

Zahlen ohne Nachkommaanteil können in Hexadezimalschreibweise durch Voranstellen eines Dollarzeichens angegeben werden. Eine Schreibweise in Binärnotation wird durch Voranstellen eines Prozentzeichens erreicht:

```
x := $04ef          // entspricht der Zahl 1263 in Hexadezimalschreibweise ($04ef)
x := %10011101111   // entspricht der Zahl 1263 in Binärschreibweise
```

7.1.2.2 Eingabe von Strings

Strings müssen in Anführungszeichen angegeben werden:

```
y := ”Foobar”
```

Sie können zwei Strings zu einem zusammenfassen, indem Sie sie konkatenieren:

```
y := ”Foo” + ”bar”  // ergibt den String ”Foobar”
```

Anführungszeichen müssen verdoppelt werden:

```
Exec(ExePath:”C:\Windows\notepad.exe”, Params:”””” + PROJEKTPFAD + ” Datei mit Leerzeichen.txt””””);
```

Soll der String Sonderzeichen enthalten, die nicht mit der Tastatur eingegeben werden können, können Sie deren ASCII-Code in eckigen Klammern verwenden.

```
y := ”Foo” + [13] + ”bar”  // fügt ein 'Newline' zwischen ”Foo” und ” bar” ein
```

7.1.2.3 Eingabe von Arrays

Zur direkten Eingabe von Arrays werden die einzelnen Werte in eckigen Klammern durch Komma getrennt geschrieben.

```
y := [”foo”, ”bar”, ”baz”]  // ein String-Array mit drei Elementen
z := [1, 2, 3, 4]           // ein Number-Array mit vier Elementen
```

7.1.3 Variablen und Funktionen

Variablen können in Ausdrücken anstelle eines Wertes verwendet werden. Sie werden während der Ausführung durch ihren Wert ersetzt. Das folgende Skript weist der Variablen `x` den Wert `x + 1` zu, erhöht also ihren Wert um eins.

```
x := x + 1
```

Variablennamen beginnen mit einem Klein- oder Großbuchstaben oder einem Unterstrich (_), und können außerdem Zahlen enthalten.

Winguard definiert mehrere Funktionen, die in Ausdrücken verwendet werden können. Eine Funktion berechnet zu einem oder mehreren Parametern einen Ergebniswert. Parameter werden in Klammern hinter dem Funktionsnamen angegeben und können beliebige Ausdrücke enthalten. Einzelne Parameter werden durch Komma voneinander getrennt:

```
x := sin(90)                    // Berechne den Sinus von 90 Grad
y := ToNumber(”Foo”, x + 2);    // Konvertiere den String ”Foo” in ↩
    eine Zahl
```

Im Abschnitt 7.4 finden Sie eine vollständige Liste aller unterstützten Funktionen. Die Funktionen sind nach ihrem Anwendungsbereich sortiert. Zu jeder Funktion ist der Typ angegeben. Betrachten wir als Beispiel die Funktion *SubStr* die einen Teilstring aus einem String extrahiert:

```
function SubStr(Str : String; Start : Real; Laenge : Real) : String;
```

Kopiere aus dem String `Str` die Zeichenkette die an Position `Start` beginnt und `Laenge` Zeichen lang ist. Das erste Zeichen des Strings besitzt den Index 0.

Die Funktion erhält die drei Parameter *Str*, *Start* und *Laenge*. Der erste Parameter ist vom Typ String, alle weiteren vom Typ Real. Der Typ hinter dem letzten Doppelpunkt gibt den Ergebnistypen der Funktion an.

Wenn wir die Funktion `SubStr` aufrufen möchten, müssen wir ihr also drei Parameter übergeben. Dabei muss der erste Parameter eine Zeichenkette sein, die letzten beiden eine Zahl:

```
x := SubStr(”Foobar”, 3, 3);
```

Obiger Aufruf kopiert von Position 3 des Strings ”Foobar” drei Zeichen. Da das erste Zeichen des Strings mit 0 indiziert liefert der Aufruf als Ergebnis den String ”bar”. Der Typ der Variable muss mit dem Ergebnistypen der Funktion übereinstimmen. Ist dies nicht der Fall, versucht Winguard den Wert zu konvertieren. Wurde die Variable x als Stringvariable deklariert, wird der Ergebniswert einfach zugewiesen und die Variable erhält ebenfalls den Wert ”bar”. Ist die Variable vom Typ Number so wird versucht den String in eine Zahl umzuwandeln. Da dies nicht möglich ist, erhält die Variable den Wert 0. Die automatische Konvertierung wird im Abschnitt 7.1.5 beschrieben.

Im zweiten Beispiel ist für den dritten Parameter bereits ein Defaultwert vorgegeben. Wird bei Aufrufen der Funktion dieser Wert ausgelassen, wird stattdessen der Defaultwert verwendet.

```
function StrPos(HayStack : String; Needle : String; Offset : Number = ↩
    0) : Number
```

Sucht in einem String HayStack den String Needle, beginnend an Position Number. Wird der String gefunden, wird die Position des Strings zurückgegeben. Kommt Needle nicht in HayStack vor, gibt die Funktion 0 zurück.

Die folgenden beiden Aufrufe sind also äquivalent, da im ersten Fall der Defaultwert verwendet wird:

```
x := StrPos(”Foobar”, ”bar”);
x := StrPos(”Foobar”, ”bar”, 0);
```

7.1.4 Operatoren

Ähnlich zu Funktionen können Operatoren verwendet werden. Ein Operator wird in der Regel zwischen zwei Ausdrücken platziert und verknüpft beide Ausdrücke zu einem neuen Ausdruck. Daneben existieren auch Operatoren, die nur mit einem Ausdruck verknüpft werden. Als Beispiel für die erste Variante haben wir bereits den Operator + kennengelernt. Ein Operator der nur einen Parameter enthält ist beispielsweise der Operator not der zu einem Zahlwert das binäre Inverse berechnet:

```
x := y + 10
y := not x
```

7.1.4.1 Prioritäten von Operatoren

In dem Ausdruck `2 * x + y` soll zuerst der Teilausdruck `2 * x` berechnet werden um das Ergebnis dann zu `y` zu addieren (Punktrechnung geht vor Strichrechnung). Wir sagen der Operator * besitzt eine höhere Priorität als der Operator +. Operatoren mit einer höheren Priorität werden immer zuerst berechnet. Operatoren mit der gleichen Priorität werden von links nach rechts ausgewertet. Die Operatoren nach ihren Prioritäten absteigend sortiert:

1. Präfix -, Präfix +, `not`, ^, #, ++
2. *, /, `div`, `mod`, `shl`, `shr`, `and`
3. -, +, `or`, `xor`
4. <=, >=, =, <, >, <>, `in`

7.1.4.2 Operatortypen

Analog zu den Funktionen finden Sie im Anhang eine Auflistung der implementierten Operatoren. Die Operatoren sind nach ihrem Ergebnistyp sortiert. Im Unterschied zu den Funktionstypen werden hier keine Defaultwerte verwendet. Die Art der Operation hängt hier von den Typen der Parameter ab.

So führt beispielsweise die `function +(X : Real; Y : Real) : Real;` die Addition zweier Zahlen durch, während die `function +(S : String; I : Real) : String;` an den String `S` die zu einem String umgewandelte Zahl `I` anhängt. Der Wert des Ausdruckes `x + 5` hängt also davon ab, ob `x` eine Zahl ist oder ein String:

```
x := ”1” + 5;   // x = ”15”, da ”1” ein String ist
y := 1 + 5;     // y = 6, da 1 eine Zahl ist
```

7.1.5 Automatische Typanpassung

Stimmt der Typ eines Parameters nicht mit dem erwarteten Typ überein, nimmt Winguard automatisch eine Anpassung des Wertes vor. Betrachten wir die Funktion `ToArray`:

```
function ToArray(Str : String): array of Number;
```

Wandle den String in ein Zahlenarray. Dabei erscheint der ASCII-Wert jedes Zeichens als Element im Array.

Der einzige Parameter ist vom Typ String. Übergeben wir statt des Strings eine Zahl, wird Winguard diese erst in einen String konvertieren und dann die Funktion `ToArray` aufrufen.

Die Konvertierung einer Zahl in einen String liefert die textuelle Repräsentation der Zahl, also für die Zahl `123` den String `"123"`. Für einen Aufruf `ToArray(123)` wird Winguard zuerst eine Konvertierung der Zahl durchführen, so dass wir einen Aufruf `ToArray("123")` erhalten und dann anschließend die Funktion anwenden, was in einem Array `[49, 50, 51]` (die ASCII-Werte der Zeichen "1", "2" und "3") resultiert.

`array of Number` nach `String`
: Interpretiert die Zahlenwerte des Arrays als ASCII-Codes und erzeugt daraus einen String. Aus dem Array `[49, 50, 51]` wird der String `"123"`.

`String` nach `array of Number`
: Legt zu jedem Zeichen des Strings ein Arrayelement an, das den ASCII-Wert des Zeichens enthält. Aus `"123"` wird das Array `[49, 50, 51]`.

`String` nach `Number`
: Wandle den String in eine Zahl um. Es werden unterschiedliche Zahlrepräsentationen erkannt. Kann der String nicht konvertiert werden, wird `0` zurückgegeben. Beispiele: `"123"` wird zu `123`, `"$123"` wird zu `291`, `"ABC"` wird zu `0`

`Number` nach `String`
: Die Zahl wird in Dezimalschreibweise in einen String konvertiert, also `123` wird zu `"123"` und `$123` zu `"291"`. %$ otra vez

`Boolean` nach `Number`
: Liefert `1` falls der bool'sche Wert `True` ist und sonst `0`.

`array of X` nach `array of Y`
: Es wird elementweise von X nach Y konvertiert. Beispiel: Die Konvertierung eines `array of String` in ein `array of Number` versucht aus allen Strings Zahlen zu erzeugen, aus `["12", "23", "34"]` wird das Array `[12, 23, 34]`.

7.2 Kontrollstrukturen

7.2.1 Ausnahmebehandlung

Tritt während der Programmverarbeitung ein Fehler auf, wird dem Benutzer von Winguard ein Fehlerdialog präsentiert, der den Fehler genauer beschreibt. Durch setzen der Variable `SuppressErrors` auf einen Wert ungleich Null, kann dieser Fehlerdialog unterdrückt werden. In diesem Fall muss der Fehler vom Programmierer selbst behandelt werden. Dazu unterstützt WGScript Ausnahmebehandlung im Stil von `try-catch`.

Um für einen Block auf Fehler zu reagieren, müssen Sie diesen Block mit den Zeilen `try` und `except` einschliessen. Nach `except` fügen Sie Code an, um den Fehler zu behandeln:

```
SuppressErrors := 1;            // Zuerst die Fehlerbehandlung ausschalten
try
    messwert := messwert / 0    // Fehler provozieren
    TextOut(Text: ”Kein Fehler?”);
except
    TextOut(Text: ”Es ist ein Fehler aufgetreten”);
end;
```

Wird versucht, Messwert durch Null zu teilen, wird das Programm im `except`-Block weiter ausgeführt. Die Zeilen nach dem Fehler werden übersprungen. Die Fehlermeldung wird in der Variablen `ErrorMessage` gespeichert. Diese kann zusätzlich ausgegeben werden:

```
SuppressErrors := 1;
try
    messwert := messwert / 0
except
    TextOut(Text: ”Es ist folgender Fehler aufgetreten: ” +
        ErrorMessage);
end;
```

Die Fehlermeldung wird nun in das Messprotokoll geschrieben.

Anstatt des `except`-Konstruktes kann auch `finally` verwendet werden. Der Code des `finally`-Abschnittes wird immer ausgeführt, egal ob ein Fehler aufgetreten ist, oder nicht. Damit kann vor dem Verlassen einer Prozedur wieder ein definierter Zustand hergestellt werden. So wird bespielsweise die komplette Finalisierung in einen `finally`-Block eingebettet.

```
RelaisOeffnen();    // Relais oeffnen
try
    MessungDurchfuehren();
finally
    RelaisSchliesse(); // auch bei Fehler das Relais wieder schliessen
end;
```

Anmerkung
Während `except`-Blöcke nur aufgerufen werden, wenn die Variable `SuppressErrors` gesetzt ist, werden `finally`-Blöcke immer abgearbeitet.

7.3 Übersicht der Systemvariablen

7.3.1 Realvariablen

ABBRUCH
Setzen auf `1` bewirkt ein Abbrechen des Prüfablaufs.

BEWERTUNG
Gesamtergebnis des Testlaufs (`0`=Fehlerhaft, `1`=Fehlerfrei). Kann auch geschrieben werden.

DATETIME
Die Anzahl der Tage seit 1899-12-30, in der lokalen Zeitzone, mit Bruchteilen für die Stunden, Minuten, Sekunden und Millisekunden.

KEYPRESS
Enthält den ASCII-Wert der zuletzt gedrückten Taste. Auslesen dieser Variable setzt die Variable auf `0` zurück.

MESSWERT
Der aktuelle Messwert vom Multimeter bzw. PSU.

PWACCESS
Enthält den Wert `0`, falls die Entwicklungsumgebung durch Passwortschutz gesichert ist, sonst `1`.

RepeatCount
Die Anzahl der Durchläufe des Repeat-Modes wie sie im Einstellungsdialog angegeben ist.

Repetition
Nummer des aktuellen Durchlaufs im Repeat-Mode.

RXDTIMEOUT
Enthält den Wert `0` falls der letzte SerialCom-Befehl innerhalb der angegebenen Zeit von der seriellen Schnittstelle lesen konnte, und `1` wenn die Wartezeit ablief.

START
Wird auf `1` gesetzt wenn der Benutzer den Start-Knopf im Hauptfenster drückt oder StartButton-Befehl ausgeführt wird. Setzen auf `1` im Debugger simuliert das Drücken der Hardware-Start-Taste. Kann auch verwendet werden, um im laufenden Betrieb auf Drücken von Start zu warten.

TIMER
Stopuhrfunktion. Setzen auf `0` bewirkt Nullstellung. Der Timer enthält die Anzahl Sekunden seit dem Zeitpunkt, zu dem die Variable auf `0` gesetzt wurde.

TIMER2–4
Wie `TIMER`.

7.3.2 Stringvariablen

AUFTRAG
Fürs Protokoll: Ein frei wählbarer Bezeichner, nach dem Protokolle aus der Datenbank ausgewählt werden können. Muss aus Kompatibilitätsgründen vom Benutzer angelegt werden.

BUTTONRESULT
Rückgabewert: Zuletzt gedrückter Knopf vom Meldungsbox-Befehl. „OK“, „ABBRECHEN“, „WIEDERHOLEN“, „IGNORIEREN“, „JA“, „NEIN“ oder der Text eines individuellen Knopfs.

CommandLineParameters
Ein Array mit den Argumenten, die beim Start von Winguard übergeben wurden.

DATE
Gibt das aktuelles Systemdatum zurück.

GPIB_ERROR
Fehlermeldung vom letzten GPIB- Befehl.

NULLSTRING
String ohne Inhalt.

ProduktID
Fürs Protokoll: Ein frei wählbarer Bezeichner, nach dem Protokolle aus der Datenbank ausgewählt werden können.

ProduktID2
Fürs Protokoll: Ein frei wählbarer Bezeichner, nach dem Protokolle aus der Datenbank ausgewählt werden können. Muss aus Kompatibilitätsgründen vom Benutzer angelegt werden.

PROJEKTPFAD
Aktueller Datei-Pfad der Projektdatei inklusive abschließendem „\“.

RXDSTRING
Rückgabewert: Empfangsstring aus serieller Kommunikation.

SERIENNUMMER
Fürs Protokoll: Ein frei wählbarer Bezeichner, nach dem Protokolle aus der Datenbank ausgewählt werden können. Erscheint auch in der Testumgebung.

TESTER
Rückgabe: Name des eingeloggten Prüfers.

TIME
Rückgabe: Aktuelle Systemzeit.

TXDSTRING
Rückgabewert: Sendestring aus serieller Kommunikation.

USERBUTTONS
Rückgabewert: Name der gedrückten Funktionstaste vom GUI-Modul USERBUTTONS. Auslesen dieser Variable setzt die Variable auf "0" zurück.

USER_REPORT_FILENAME
Zuweisung des Protokolldateinamens

WINGUARD_PATH
Gibt das WinGuard Programmverzeichnis zurück.

7.4 Übersicht der Funktionen

7.4.1 Funktionen zur Zahlmanipulation

7.4.1.1 Betrag

```
function abs(X: Real): Real;
```

Berechnet den absoluten Betrag des Parameters X.

7.4.1.2 Konvertierung in Text

```
function inttostr(X: Real; Places: Real = 0): String;
function realtostr(X: Real; Digits = -1; Places: Real = -1;): String;
```

Wandelt die Zahl X in einen Text um. `Places` gibt dabei optional die Nachkommastellen an. Wenn die Anzahl der Nachkommastellen von X kleiner ist als `Places`, wird das Ergebnis mit Nullen aufgefüllt. `Digits` gibt die Anzahl der Vorkommastellen an, bzw. die Anzahl signifikanter Stellen falls `Places` -1 ist.

7.4.1.3 Konvertierung in Text (Hexadezimal)

```
function inttohex(X: Real): String;
```

Wandelt die Zahl X in einen String um, der die Repräsentation der Zahl in Hexadezimaldarstellung enthält.

7.4.1.4 Runden

```
function round(X: Real): Real;
```

Rundet die Zahl X auf die nächste ganze Zahl. Liegt X genau zwischen zwei Zahlen, wird die gerade Zahl zurückgegeben.

7.4.1.5 Kosinus

```
function cos(X: Real): Real;
```

Berechnet den Kosinus des Winkels X. X muss im Gradmaß angegeben werden.

7.4.1.6 Arkuskosinus

```
function arccos(X: Real): Real;
```

Berechnet den Arkuskosinus von X. Das Ergebnis ist im Gradmaß.

7.4.1.7 Logarithmus

```
function log(X: Real): Real;
```

Berechnet den Logarithmus von X zur Basis 10.

7.4.1.8 Natürlicher Logarithmus

```
function ln(X: Real): Real;
```

Berechnet den natürlichen Logarithmus von X.

7.4.1.9 Quadrieren

```
function sqr(X: Real): Real;
```

Berechnet das Quadrat des Parameters X. Der Aufruf dieser Funktion ist äquivalent zur Auswertung des Terms `x^2`.

7.4.1.10 Quadratwurzel

```
function sqrt(X: Real): Real;
```

Berechnet die Quadratwurzel von X. Diese Funktion ist äquivalent zu dem Ausdruck `x^(1/2)`. Zum ziehen der *n*-ten Wurzel kann auch `x^(1/`*n*`)` verwendet werden.

7.4.1.11 Sinus

```
function sin(X: Real): Real;
```

Berechnet den Sinus von X. Beachten Sie, dass X im im Gradmaß angegeben werden muss.

7.4.1.12 Arkussinus

```
function arcsin(X: Real): Real;
```

Berechnet den Arkussinus von X. Das Ergebnis ist im Gradmaß.

7.4.1.13 Tangens

```
function tan(X: Real): Real;
```

Berechnet den Tangens des Winkels X. X muss im Gradmaß vorliegen.

7.4.1.14 Arkustangens

```
function arctan(X: Real): Real;
```

Berechnet den Arkustangens von X. Das Ergebnis ist im Gradmaß.

7.4.1.15 Zufallszahl

```
function Random(Limit: Real): Real;
```

Liefert eine Zufallszahl zwischen 0 und Limit.

7.4.1.16 Fakultät

```
function fact(X: Real): Real;
```

Liefert X!.

7.4.2 Funktionen über Strings

7.4.2.1 Länge

```
function Length(Arr: Array): Real;
function Length(Str: String): Real;
```

Liefert als Ergebnisse die Anzahl der Elemente des Arrays Arr beziehungsweise der Zeichen des Textes Str.

7.4.2.2 Teilstring

```
function SubStr(Str: String; Start: Real; Laenge: Real): String;
```

Kopiere aus dem String Str die Zeichenkette die an Position Start beginnt und Laenge Zeichen lang ist. Das erste Zeichen des Strings besitzt den Index 0.

7.4.2.3 String aufteilen

```
function Split(Str: String; Separator: String): Array;
```

Teilt den String `Str` an den Vorkommnisen des Strings `Separator` auf und erzeugt daraus ein Array.

```
Bsp: Aus +Split(”1,2,3,4”, ”,”);+ wird das Array +[”1”, ”2”, ”3”, ”4”]+.
```

7.4.2.4 Array zusammenführen

```
function Join(Arr: Array; Separator: String): String;
```

Fügt alle Elemente des Arrays zu einem String zusammen. Je zwei Elemente werden durch den String `Separator` getrennt.

Beispiel: `Join([”1”, ”2”, ”3”, ”4”], ”,”);` erzeugt den String `”1,2,3,4”`.

7.4.2.5 Array zusammenführen

```
function Delete(Arr: Array; Position: Real; Length: Real): Array;
```

Erzeugt einen Array mit allen Elementen aus `Arr` bis auf die `Length` Elemente beginnend an `Position`.

7.4.2.6 In einem String suchen

```
function StrPos(HayStack: String; Needle: String; Offset: Number = 0) : Number
```

Sucht in einem String `HayStack` den String `Needle`, beginnend an Position `Number`. Wird der String gefunden, wird die Position des Strings zurückgegeben. Kommt `Needle` nicht in `HayStack` vor, gibt die Funktion 0 zurück.

7.4.2.7 String ersetzen

```
function StrReplace(HayStack: String; Needle: String; Replace: String ): String;
```

Ersetze alle Vorkommnisse von `Needle` in `HayStack` durch `Replace`.

7.4.2.8 Leerzeichen löschen

```
function Trim(Str: String): String;
```

Lösche alle Leerzeichen am Anfang und am Ende des Strings `Str`.

7.4.2.9 Wandle einen String in ein Zahlenarray um

```
function ToArray(Str: String): array of Number;
```

Wandle den String in ein Zahlenarray. Dabei erscheint der ASCII-Wert jedes Zeichens als Element im Array.

7.4.2.10 Wandle einen String in eine Zahl um

```
function ToNumber(Str: String; Default: Number = 0): Number;
```

Wandle einen String explizit in eine Zahl um. Sollte die Umwandlung fehlschlagen, wird der Defaultwert zurückgegeben.

Anmerkung
Normalerweise führt Winguard automatisch eine Umwandlung durch, falls einer Funktion, die ein Zahl als Parameter erwartet, ein String übergeben wird. Für Operatoren, die auf mehreren Typen arbeiten, kann diese Umwandlung mitunter nicht eindeutig bestimmt werden. Winguard meldet in diesem Fall „Die aufgerufene Funktion konnte nicht eindeutig bestimmt werden.“. In solch einem Fall, kann mit dieser Funktion die Konvertierung explizit vorgenommen werden.

7.4.2.11 Wandle einen String in eine Ganze Zahl um

```
function Int(Str: String; Default: Number = 0): Number;
```

Wandle einen String explizit in eine Ganze Zahl um. Sollte die Umwandlung fehlschlagen, wird der Defaultwert zurückgegeben.

Normalerweise kann die Funktion `ToNumber` verwendet werden, um Strings in Zahlen zu konvertieren. Die Konvertierung von Zahlen in Hexadezimalnotation wird von `ToNumber` allerdings verweigert. `Int` stellt diese Funktionalität zur Verfügung.

7.4.2.12 ASCII-Wert eines Zeichens

```
function ToAscii(Str: String): Number;
```

Liefert zu einem String den ASCII-Wert des ersten Zeichens oder -1 falls der String leer ist.

7.4.2.13 String aus ASCII

```
function FromAscii(Ascii: Number): String;
```

Liefert das Zeichen mit dem gegebenen Ascii-Wert.

7.4.2.14 Umwandeln in Großbuchstaben

```
function ToUpper(Str: String): String;
```

Wandelt den übergebenen String in Großbuchstaben um.

7.4.2.15 Umwandeln in Kleinbuchstaben

```
function ToLower(Str: String): String;
```

Wandelt den übergebenen String in Kleinbuchstaben um.

7.4.2.16 Konvertierung von Arrays in einen Text

```
function String(Arr: Array; Format: Real = 0; Laenge: Real = MaxInt;  ↩
    Sep: String = ”): String;
```

Wandelt `Arr` in einen Text um. Der Parameter `Laenge` gibt an, wie viele Elemente des Arrays maximal angezeigt werden. Ist der Parameter `Sep` nicht leer, wird der String zwischen den Einzelnen Elementen eingefügt. `Format` gibt an, wie Arraywerte vom Typ `Real` in einen String umgewandelt werden sollen:

0	Die Werte des Arrays werden als ASCII-Codes interpretiert.

1	Die Werte des Arrays werden als Dezimalzahl dargestellt.
2	Die Werte des Arrays werden als Hexadezimalzahl dargestellt.
3	Die Werte des Arrays werden als Fliesspunktzahl dargestellt.
4	Wie 2., jedoch ohne führendes $.

Die Parameter `Format`, `Laenge` und `Sep` sind optional.

7.4.2.17 Abfrage des Arbeitsverzeichnisses

```
function GetCWD(): String;
```

Git das aktuelle Arbeitsverzeichnis zurück.

7.4.3 Protokolle

Mit diesen Funktionen kann sowohl auf das im aktuellen Prüfablauf erstellte Protokoll als auch auf Protokolle aus dem Archiv zugegriffen werden. Dazu wird das gerade anzeigte Protokoll gewechselt. Auf das Archiv wird über die Position in der Liste von gegebenenfalls gefilterten Protokollen im Archiv zugegriffen.

7.4.3.1 Protokollfilter: Programmname

```
function ProtoFilterProgram(ProgId: String): Void;
```

Filtere nach dem Pfad des Prüfprogramms, welches das Protokoll erzeugt hat. Ein leerer String bedeutet, dass nicht nach Prüfprogramm gefiltert wird.

7.4.3.2 Protokollfilter: Programbewertung

```
function ProtoFilterValuation(Valuation: Number): Void;
```

Filtere nach Programmbewertung. Die Bewertung wird als Bit-Array angegeben, wobei die Bits folgende Bedeutung haben: 1=Fail, 2=Pass, 4=Abort

Beim Programmstart oder nach ProtoClearFilter entspricht dieser Filter dem Wert 7, es sind also alle Bits gesetzt.

7.4.3.3 Protokollfilter: Datum

```
function ProtoFilterDate(StartDate: String; EndDate: String): Void;
```

Nur auf Protokolle aus einem bestimmten Zeitraum zugreifen. Das Datum wird als String angegeben, das Format entspricht `Tag.Monat.Jahr Stunde:Minute:Sekunde`. Ein leerer String bedeutet, dass nicht nach dem entsprechenden Kriterium gefiltert wird.

7.4.3.4 Protokollfilter: Seriennummer

```
function ProtoFilterSerial(Serial: String): Void;
```

Nur Protokolle für eine bestimmte Seriennummer zulassen. Ein leerer String bedeutet, dass nicht nach Seriennummer gefiltert wird.

7.4.3.5 Protokollfilter: Prüflingsbezeichnung

```
function ProtoFilterIdent(Ident: String): Void;
```

Nur Protokolle für eine bestimmte Prüflingsbezeichnung zulassen. Ein leerer String bedeutet, dass nicht nach Prüflingsbezeichnung gefiltert wird.

7.4.3.6 Protokollfilter zurücksetzen

```
function ProtoClearFilter(): Void;
```

Löschen des Protokollfilters. Es wird wieder auf das gesamte Archiv zugegriffen.

7.4.3.7 Anzahl der Protokolle

```
function ProtoGetCount(): Number;
```

Liefert die Anzahl der Protokolle für den aktuell eingestellten Protokollfilter.

7.4.3.8 Protokoll anzeigen

```
function ProtoLoadProto(Index: Number): Void;
```

Lade das Protokoll mit der angegebenen Nummer aus der Protokollliste in das Messwertprotokoll. Das aktuelle Messprotokoll wird beim Beenden des Programms automatisch wieder angezeigt.

7.4.3.9 Aktuelles Protokoll anzeigen

```
function ProtoShowActive(): Void;
```

Zeige das im aktuellen Prüfablauf erzeugte Messprotokoll wieder an.

7.4.3.10 Text des Protokolls

```
function ProtoGetText(Index: Number): String;
```

Liefert den Text in der ersten Spalte des Messprotokolls in der angegebenen Zeile.

7.4.3.11 Werte im Protokoll

```
function ProtoGetValue(Index: Number): Number;
function ProtoGetValue(Text: String): Number;
```

Liefert den Wert in der angegebenen Zeile. Wenn Text angegeben ist, wird die erste Zeile verwendet, in der in der ersten Spalte der Text steht.

7.4.3.12 Werte im Protokoll

```
function ProtoGetUnit(Index: Number): String;
```

Liefert die Einheit in der angegebenen Zeile.

7.4.3.13 Grenzen im Protokoll

```
function ProtoGetUpperLimit(Index: Number): Number;
function ProtoGetLowerLimit(Index: Number): Number;
```

Liefert die verwendeten Grenzen in der angegebenen Zeile.

7.4.3.14 Bewertung im Protokoll

```
function ProtoHasFailed(Index: Number): Boolean;
function ProtoHasFailed(Text: String): Boolean;
```

Wenn die angegebene Zeile einen fehlgeschlagenen Testschritt repräsentiert, liefere `true`. Wenn Text angegeben ist, wird die erste Zeile verwendet, in der in der ersten Spalte der Text steht.

7.4.3.15 Ergebnisse im Protokoll

```
function ProtoIsResult(Index: Number): Boolean;
function ProtoIsResult(Text: String): Boolean;
```

Wenn die angegebene Zeile einen Testschritt repräsentiert, liefere `true`. Wenn Text angegeben ist, wird die erste Zeile verwendet, in der in der ersten Spalte der Text steht.

7.4.3.16 Metadaten des Protokolls

```
function ProtoGetTestStation(): String;
```

Liefert den Namen des Prüfplatzes.

7.4.4 Funktionen zum Konvertieren zwischen Datenformaten

7.4.4.1 IEEE-754 Single nach Byte-Repräsentation

```
function IEEE754SingleToByteArray(Value: Number): array of Number;
```

Wandle den Zahlenwert in eine Fliesspunktzahl nach IEEE-754 Single-Precision um und liefere den daraus resultierenden Speicherinhalt in einem Byte-Array.

Die Konvertierung kann zu Datenverlust führen! IEEE-754 definiert den Datentyp Single als 32bit-Fliesspunktzahl, mit einem 8bit-Exponenten und 23bit-Mantisse. Das resultierende Array besitzt vier Elemente.

7.4.4.2 Byte-Array nach IEEE-754 Single

```
function ByteArrayToIEEE754Single(Value: array of Number): Number;
```

Umkehrfunktion zu *IEEE754SingleToByteArray*: das übergebene Zahlenarray muss mindestens vier Werte aufweisen. Die Werte werden als Auszug eines Speichers, der ein IEEE-754-Single enthält, interpretiert. Zurückgegeben wird der Wert dieser Fliesspunktzahl.

7.4.4.3 Datum und Uhrzeit in Text im ISO8601-Format

```
function DateTimeToISO8601(DateTime: Real): String;
```

Konvertiert den Zeitstempel (im gleichen Format wie die DATETIME-Variable) in einen String im ISO8601-Format mit Zeitverschiebung.

7.4.4.4 Tag im Jahr von Datum und Uhrzeit

```
function DayOfTheYear(DateTime: Real): Real;
```

Gibt die Anzahl der Tage zwischen dem angegebenen Zeitstempel (im gleichen Format wie die DATETIME-Variable) und dem 31. Dezember des vorhergehenden Jahres zurück.

7.4.5 Verschiedenes

7.4.5.1 Datei lesen

```
function ReadFile(FileName: String): Array;
```

Lese die Datei mit dem Pfad `FileName` in ein Array ein.

7.4.5.2 Datei schreiben

```
function WriteFile(FileName: String; Contents: Array): Number;
```

Schreibe den Inhalt des Arrays zeilenweise in die Datei mit Namen `FileName`. Sollte die Datei existieren, wird der alte Inhalt gelöscht.

7.4.5.3 Skript ausführen

```
function Eval(Script: String): Misc
```

Führt das übergebene Skript aus und liefert den Ergebniswert. Bsp: `Eval(”1 + 2”) = 3`.

7.4.5.4 Umgebungsvariable abfragen

```
function GetEnv(VarName: String): String
```

Liest den Wert der Umgebungsvariable ”VarName”.

7.5 Auflistung der Operatoren

7.5.1 Operatoren mit Ergebnistyp Real

^	`function ^(X: Real; Y: Real): Real;`	Potenzierung zweier Zahlen: X^Y
*	`function *(X: Real; Y: Real): Real;`	Multiplikation zweier Zahlen
/	`function /(X: Real; Y: Real): Real;`	Division zweier Zahlen
+	`function +(X: Real; Y: Real): Real;`	Addition zweier Zahlen
	`function +(X: Real): Real;`	+X = X
-	`function -(X: Real; Y: Real): Real;`	Subtrahiere Y von X

	`function -(X: Real): Real;`	Negation des Vorzeichens
`div`	`function div(X: Real; Y: Real): Real;`	Ganzzahldivision, die Zahlen werden vor der Operation gerundet.
`mod`	`function mod(X: Real; Y: Real): Real;`	Rest der Ganzzahldivision
`shr`	`function shr(X: Real; Y: Real): Real;`	Bitweise Schiebeoperation nach Rechts. Die Zahlen werden gerundet.
`shl`	`function shl(X: Real; Y: Real): Real;`	Bitweise Schiebeoperation nach Links. Die Zahlen werden gerundet.
`and`	`function and(X: Real; Y: Real): Real;`	Bitweise Und-Verknüpfung.
`or`	`function or(X: Real; Y: Real): Real;`	Bitweise Oder-Verknüpfung.
`xor`	`function xor(X: Real; Y: Real): Real;`	Bitweise Exklusiv-Oder-Verknüpfung.

7.5.2 Operatoren mit Ergebnistyp String

`+`	`function +(S1: String; S2: String): String;`	Verknüpft zwei String miteinander (”abc” + ”def” = ”abcdef”)
	`function +(S: String; I: Real): String;`	Wandelt `I` in einen String um und hängt ihn an den String `S` an.
	`function +(I: Real; S: String): String;`	Wandelt `I` in einen String um und hängt an diesen den String `S` an.
	`function +(A: Array; S: String): String;`	Wandelt das Array `A` in einen String um und verknüpft die Strings.
	`function +(S: String; A: Array): String;`	Wandelt das Array `A` in einen String um und hängt ihn an den String `S` an.

*	`function *(S: String; I: Real): String;`	Schreibt den String S I-mal hintereinander.
	`function *(I: Real; S: String): String;`	Ebenso.

7.6 Fehlermeldungen

7.6.1 Syntaxfehler

Ein Syntaxfehler tritt immer dann auf, wenn die Skriptzeile fehlerhaft ist und nicht vom Interpreter gelesen werden kann. Dies kann passieren wenn ein Ausdruck keiner gültigen Form entspricht, weil in ihm zum Beispiel ein nicht erwartetes Zeichen gefunden wurde. Folgende Zeile führt zu einem Syntaxfehler, da "a :" keinen gültigen Ausdruck darstellt:

```
b := a :
```

Der Fehlerdialog wird neben der Zeile angezeigt, in der der Fehler aufgefallen ist. Die problematische Stelle wird dabei im Script rot hinterlegt. Die Ursache des Fehlers kann aber auch in einer früheren Zeile liegen, beispielsweise wenn eine schließende Klammer fehlt.

Um den Fehler zu beheben, editieren Sie die problematische Zeile und starten Sie dann das Programm erneut.

7.6.2 Laufzeitfehler

Wenn während des Programmablaufs ein Fehler auftritt, spricht man von einem Laufzeitfehler. Ein typischer Laufzeitfehler ist ein Zugriff auf ein nicht verfügbares Gerät, beispielsweise ein nicht eingeschalteter Guardian. Im Fehlerdialog erfahren Sie, an welcher Stelle im Programm der Fehler aufgetreten ist und um welchen Fehler es sich im Detail handelt. Ferner wird Ihnen die fehlerhafte Zeile angezeigt.

Falls sich der Fehler ohne weiteres beheben lässt, können Sie versuchen den entsprechenden Befehl erneut auszuführen, indem Sie Wiederholen klicken. Wenn Sie den Fehler Ignorieren wird einfach mit der nächsten Anweisung fortgefahren.

Um das Programm zu pausieren und zu der fehlerhaften Zeile in der Entwicklungsumgebung zu wechseln klicken Sie Debug.

Ein Klick auf Abbrechen bricht den Programmablauf ab. Gegebenenfalls wird dabei zunächst noch die Finalisierung abgearbeitet. Ist der Fehler in der Finalisierung auftreten, wird das Programm komplett beendet.