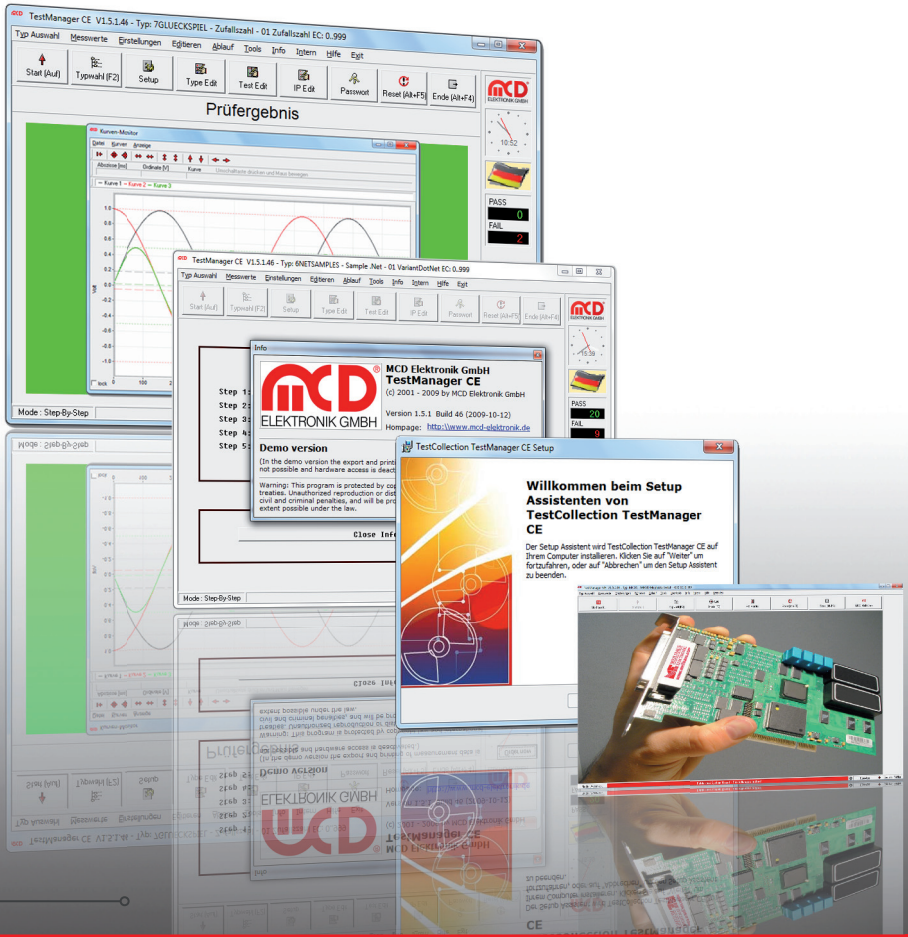
Messtechnologie bis ins Detail

MCD ELEKTRONIK GMBH

TestManager CE

Software-Entwicklungs-Umgebung für Prüfabläufe und Test



Das Einsteiger-Lehrbuch mit praxisnahen Übungen, Schritt für Schritt erklärt

TestManager Classic Edition

Testsequenzer inklusive
Entwicklungs-Umgebung für Prüfabläufe und Test.

MCD Elektronik TestManager CE • Software-Entwicklungs-Umgebung für Prüfabläufe und Test.
1. Auflage, für TestManager ab Version 1.5.1.46

Die Informationen in diesem Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise zu Fehlern ist der Herausgeber dankbar.
E-Mail: info(at)mcd-elektronik.de

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

Microsoft Windows, Exel und Explorer sind eingetragene Warenzeichen der Microsoft Corp.
DBASE ist ein eingetragenes Warenzeichen der Fa. Borland International, Inc.
MCD Elektronik ist ein eingetragenes Warenzeichen der MCD Elektronik GmbH.
Weitere Produkt- und Firmennamen sind zum Teil eingetragene Warenzeichen der jeweiligen Firmen.

© 2010 by MCD Elektronik GmbH
Carl-Zeiss-Str. 4 • 75217 Birkenfeld
www.mcd-elektronik.de
Alle Rechte vorbehalten
Printed in Germany

Inhalt

Einleitung 7

MCD TestManager CE Software 7

Grundgedanke des Buches 7

1. Einführung.....9

Download TestManager CE.....9

Die Installation.....9

Erste Anwendung.....11

Die Entwicklungsumgebung 11

■ **Übung 1-1.....12**

Starten eines vorhandenen Prüfablaufs 12

Testfenster.....14

2. Schritt für Schritt zum eigenen Prüfablauf.....17

Typfamilie.....17

■ **Übung 2-1.....17**

Anlegen einer neuen Typfamilie 17

Typvariante.....19

■ **Übung 2-2.....19**

Verwalten von Varianten 19

Prüfablauf.....21

Ablaufliste 22

Ablaufschritte 22

Prüfschritte 23

■ **Übung 2-3.....24**

Prüfablauf erstellen 24

Interpreter.....26

Interpretereditor 27

Interpreterhilfe 28
Debugger 28
■ **Übung 2-4.....28**
Interpreterschritt editieren 28

3. Screen-Objekte.....33
■ **Übung 3-1.....33**
Screen-Objekte 33

4. Globale Variable.....37
Datentypen wandeln.....37
Programmieren leicht gemacht.....37
■ **Übung 4-1.....38**
Glücksspiel 38

5. Tools.....45
■ **Übung 5-1.....45**
Verwendung von Tools 45

6. Files.....49
■ **Übung 6-1.....49**
Schreiben und Lesen von Files 49
■ **Übung 6-2.....53**
Schreiben, Lesen, Umbenennen von Files und Kennenlernen der Date-Time Funktionen 53
INI-File.....60
■ **Übung 6-3.....61**
Erstellen, Schreiben und Lesen von INI-Files 61

7. Hardware Zugriff.....67
Lizenz anfordern 67
Serielle Schnittstelle.....67

■ **Übung 7-1.....67**
Verwendung serieller Schnittstellen 67

8. Messdaten.....73

■ **Übung 8-1.....73**
Umgang mit Messdaten 73

9. Reports.....77

■ **Übung 9-1.....77**
Erstellen eines eigenen Reports 77

10. DLL.....85

■ **Übung10-1.....85**
Umgang mit DLLs 85

11. TestManager ME Messkarten Unterstützung.....91
Die Entwicklungsumgebung TestManager ME 91

Anhang.....93
System-Voraussetzungen 93
Interessante Web-Links für TestManager-User 94
Die wichtigsten TestManager CE Menüs 95
Unterschiede im Menü des TestManager ME 96

Einleitung

Haben Sie schon einmal mit einer Software-Entwicklungs-Umgebung oder einer der „klassischen“ Programmiersprachen wie z.B. Visual Basic, C#, C++ oder mit einer anderen C-Variante gearbeitet?
Dann sollten Ihnen wichtige Grundbegriffe der Programmiersprachen, wie z.B. Datentypen, Schleifen oder Syntax geläufig sein. Wenn Sie noch keine Erfahrungen mit dem Programmieren gesammelt haben und Ihnen die Grundbegriffe völlig fremd erscheinen, wird es Ihnen trotzdem leicht fallen, Prüfabläufe mit dem MCD TestManager zu erstellen.

MCD TestManager CE Software

Der MCD TestManager stellt einen Testsequenzer inklusive Entwicklungs-Umgebung für Prüfabläufe und Test dar. Die Umgebung ermöglicht das Anlegen und Verwalten von Typfamilien und deren Varianten, für die Prüfabläufe erstellt werden können. Der TestManager bietet ein großes Spektrum an Möglichkeiten, wie zum Beispiel die Anbindung an unterschiedliche Hardwarekomponenten. Es werden gängige Bussysteme, wie unter anderem der LIN, CAN, TCP/IP und IIC-Bus unterstützt. Dem Anwender werden eine Vielzahl von Tools und Einstellungsmöglichkeiten geboten.

Grundgedanke des Buches

Dieses Buch dient zur Erleichterung des Einstiegs in die MCD TestManager Entwicklungs-Umgebung. Mit Hilfe von Übungen soll der Zugang zu den Funktionen und Möglichkeiten des TestManagers CE erleichtert werden. Es kann jedoch nicht als allgemeines Nachschlagwerk oder als Handbuch angesehen werden.
Im Mittelpunkt des Buches stehen die Übungen, die sich stark an der Praxis orientieren. Sie sind so gewählt, dass das Gelesene direkt umgesetzt werden kann und der Anwender nützliche Tipps und weitere Informationen erhält.

Wie schon erwähnt stellt dieses Buch kein allgemeines Nachschlagwerk dar. Es können aufgrund von Umfang und Einfachheit lediglich einige Themen behandelt werden. Sollten jedoch Fragen und Probleme auftauchen, die eine tiefere Betrachtung der Dinge erfordern, finden Sie im Anhang Quellen, die ihren Wissensdurst stillen dürften.

1. Einführung

Download TestManager CE

Um das Buch effektiv nutzen und die Übungen durchführen zu können, laden Sie sich bitte den MCD TestManager CE aus dem Internet herunter. Die Software erhalten Sie über unsere Homepage unter **www.mcd-elektronik.de/tmce.html.** Downloaden Sie die ➡ **Trialversion** des TestManagers.

Die Installation

Nachdem der Download abgeschlossen ist, finden Sie in ihrem Ordner, in dem Ihre Downloads abgelegt werden, einen ZIP-Ordner mit dem Namen **Setup.zip**. Dieser enthält die **SetupTestManagerCE.exe**, die ausgeführt werden muss.
Es wird nun das Installationsfenster geöffnet.
Wählen Sie die Sprache und bestätigen Sie diese mit dem Button ➡ **OK**.

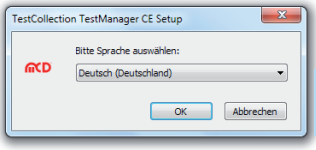


Bild 1-1
Sprachauswahl

Bestätigen Sie, dass Sie den TestManager CE installieren möchten mit ➡ **Weiter**.

Bild 1-2
Installation

Im folgendem Fenster werden wichtigen Informationen zu Systemvoraussetzungen und zum Umgang mit dem Programm angezeigt. Lesen Sie den Text aufmerksam durch und bestätigen Sie dann mit ➡ **Weiter**.

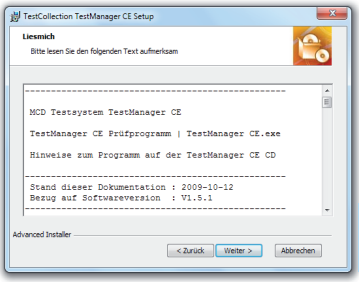


Bild 1-3
Hinweise zum Programm

Im nächsten Schritt werden Sie aufgefordert das Verzeichnis zu wählen unter dem das Programm installiert werden soll. Bestätigen Sie den voreingestellten Pfad oder ändern Sie ihn über ➡ **Wählen** und bestätigen Sie dann mit ➡ **Weiter**.

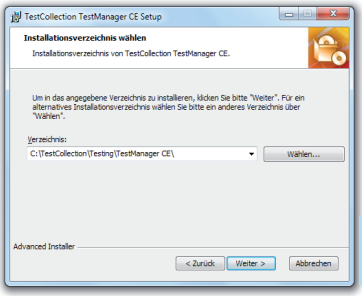


Bild 1-4
Verzeichnis wählen

Um den TestManager CE zu installieren wählen Sie ➡ **Installieren**. Um die Installation zu beenden wählen Sie ➡ **Fertigstellen**.

Bild 1-5
Fertigstellen

Der MCD TestManager CE ist nun auf ihrem System installiert und kann ausgeführt werden. Auf Ihrem Desktop befindet sich eine Verknüpfung (TestCollection.TestManager CE), über die Sie den TestManager starten können.

Erste Anwendung

Mit einem Doppelklick auf die Verknüpfung wird der TestManager CE gestartet. Je nach Betriebssystem kann es zu einer Meldung kommen, dass das Programm neu gestartet werden muss. Sollte dies der Fall sein, schließen Sie das Programm und starten es neu. Es handelt sich bei diesem Programm um eine Demoversion, daher werden Sie ab und an vom System über eine Infobox aufgefordert, diese mit ➡ **OK** zu bestätigen. Es kann ebenso eine Vollversion über ➡ **Order now** angefordert werden.

Die Entwicklungsumgebung

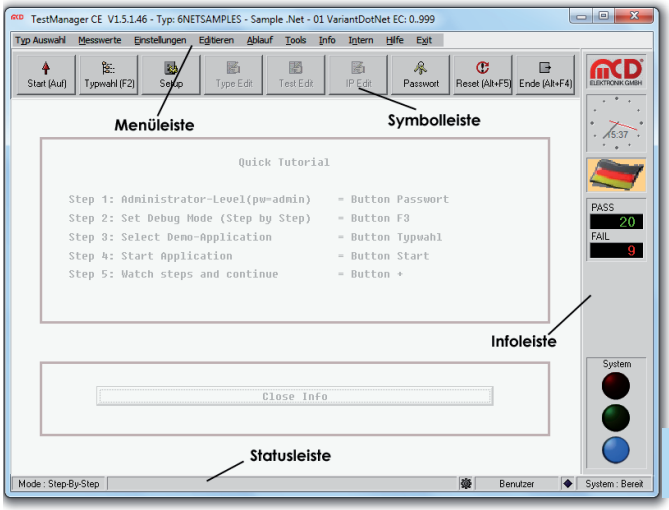


Bild 1-3
Programm-Fenster

Das **Bild 1-3** zeigt das Programm-Fenster der Entwicklungsumgebung. Hier können alle Einstellungen vorgenommen, Tools ausgewählt, Funktionen aktiviert und deaktiviert werden und vieles mehr. Sie erhalten zu gegebener Zeit zu diesen Einstellungsmöglichkeiten, Funktionen, Tools etc. nähere Informationen und Beispiele.

Der **Hauptbereich** wird für die Anzeige von Prüfergebnissen genutzt, wenn diese Option aktiviert ist. Alternativ kann dieser Bereich während des Prüfablaufes zur Ausgabe von Informationen genutzt werden.
In der Menüleiste ist das **Hauptmenü** verankert. Über dieses erlangt man den Zugang zu allen Funktionen, sofern diese im Setup aktiviert sind.
Die Menüleiste enthält folgende Register *(siehe Anhang - Die wichtigsten TMCE Menüs)*: Typ Auswahl, Messwerte, Einstellungen, Editieren, Ablauf, Tools, Info, Intern, Hilfe und Exit.

Die **Symbolleiste** enthält Buttons zum Aufruf wichtiger Funktionen ohne den Umweg über das Menü. Die Anzahl und Belegung der Buttons kann über die Grundeinstellungen variiert und auf die Wünsche des Anwenders angepasst werden.
Die Symbolleiste enthält standardmäßig folgende Buttons: Start, Typauswahl, Setup, Typ Edit, Test Edit, IP Edit, Passwort, Reset und Ende.

Informationen zum System sind in der **Infoleiste** enthalten. Auch in dieser Leiste lassen sich die einzelnen Elemente ein- und ausblenden. Sie enthält die Statusampel, die den aktuellen Systemzustand wiederspiegelt. Eine Uhr und Informationen über die zurzeit ausgewählte Sprache finden Sie ebenfalls in dieser Leiste.

Tipp:
Über die Grundeinstellungen lassen sich die Symbol- und die Infoleiste benutzerdefiniert anpassen.

Die **Statusleiste** enthält Informationen über den Modus der aktuell ausgewählt ist, die Lizenzierung der Software, den Benutzer und das System.

■ Übung 1-1
Starten eines vorhandenen Prüfablaufs

Lernziel:
- Kennenlernen von Prüfabläufen
- Aufbau von Prüfabläufen
- Umgang mit TestManager

Schritt für Schritt:

1. Als Administrator anmelden
Navigieren Sie über die ➡ **Symbolleiste** ➡ **Passwort ins Passworteingabefenster.** Drücken Sie die Funktionstaste ➡ **F4** oder wählen Sie ➡ **Administrator** aus und geben Sie im Eingabefenster das Passwort ➡ **admin** ein. Bestätigen Sie das Passwort mit der Eingabetaste oder mit dem ➡ **OK**-Button.

Hinweis:
Anwender können im TestManager mit Hilfe von 5 Passwortstufen unterteilt werden. Hinter den verschiedenen Passwortstufen sind unterschiedliche Berechtigungen hinterlegt, die über die ➡ **Menüleiste** ➡ **Intern** ➡ **Berechtigungen** editiert werden können.

2. Auswahl der Typfamilie 0MATH:
Wählen Sie über ➡ **Symbolleiste** ➡ **Typwahl** um ins Typwahlfenster zu gelangen. Öffnen Sie die Baumstruktur der **Typfamilie 0MATH** mit einem Doppelklick auf den Familiennamen oder über das Zeichen ➡ **+**. Die gleiche Vorge-

hensweise wenden Sie beim nun sichtbaren ➡ **Code: 01 VariantMath** der Typfamilie an und wählen Sie den Ablauf ➡ **1** aus. Bestätigen Sie die Wahl mit dem ➡ **OK**-Button.

SBS und Automatik Modus:
Im Normalbetrieb befindet sich das System im Automatik Modus. Der Test wird hierbei genau nach dem festgelegten Prüfablauf ausgeführt. Aus den einzelnen Prüfschrittergebnissen wird ein Prüflingsergebnis gebildet, um den Prüfling zu bewerten. Der SBS Modus (Step-by-Step) gibt dem Anwender Funktionen zur Hand, mit denen Ablaufschritte Schritt für Schritt abgehandelt und editiert werden können. Der Modus ist für den Entwickler von Prüfabläufen oder für das Debugging vorgesehen. Während eines Prüfschritts können unter anderem Informationen ausgegeben oder der Quellcode editiert werden.

3. Starten Sie den Prüfablauf manuell:
Der Prüfablauf wird über die ➡ **Symbolleiste** ➡ **Start (Auf)** oder alternativ über die Pfeiltaste nach oben der Tastatur gestartet. Im Einzelschritt Modus (Step-by-Step, SBS) hält das System nach jedem Prüfschritt an und der Bediener kann innerhalb des Prüfablaufes beliebig den nächsten Prüfschritt auswählen. Die Umschaltung kann sowohl vor dem Start der Prüfung als auch während der Prüfung geschehen.

4. Prüfschritt wiederholen:
Befindet sich das System im SBS (Step-by-Step) Modus, so lassen sich die einzelnen Prüfschritte wiederholen. Dies kann über den Button ➡ **Wiederholen**, der im Prüffenster unten angeordnet ist, ausgelöst werden.

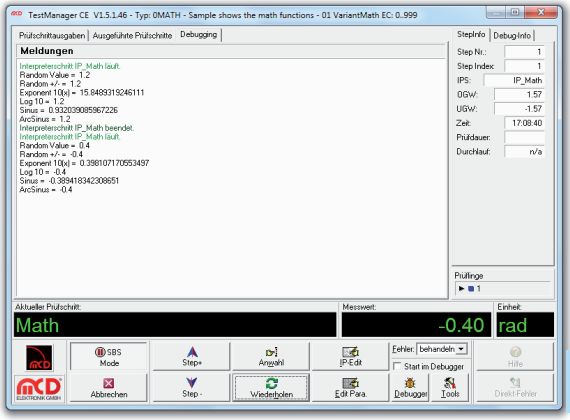


Bild 1-4
Prüfschritt wiederholen

Was macht der Ablauf?
In diesem Ablauf wird zunächst eine Zufallszahl generiert. Das **Bild 1-4** zeigt

den Schritt, der drei Mal wiederholt wurde. Beim ersten Mal wurde die Zufallszahl 0 dann die 1,2 und zuletzt die 0,4 erzeugt. Für diese Zufallszahlen wird dann ein ebenfalls zufälliges Vorzeichen gewählt. Die nun entstandene Zufallszahl wird als Exponent zur Basis 10 verwendet und das Ergebnis berechnet. Beispiel aus dem Bild: $10^{-0,4}$ = 0,398107170553497. Um nun wieder die generierte Zufallszahl zu erhalten, wird diese mathematische Operation umgekehrt. Es wird vom Ergebnis der Logarithmus zur Basis 10 errechnet. Aus dieser Rechnung muss somit erneut die zu Beginn erzeugte Zufallszahl (-0,4) resultieren. Nun wird der Sinus der Zufallszahl errechnet (-0,38941…) und auch diese Operation wird wiederum umgekehrt und der Arcussinus aus dem Ergebnis errechnet. Hieraus muss nun ebenfalls wieder die erzeugte Zufallszahl resultieren, die nun als Messwert im Messwertfenster ausgegeben wird.

5. Wählen des nächsten Steps:
Über den Button ➡ **Step+** kann zum nächsten Step weitergeschaltet werden.

Der Prüfablauf ist nach nur einem Step beendet, da der Prüfablauf nur einen Schritt beinhaltet. Somit wird nach der Wahl des nächsten Steps mit **Step+** der Prüfablauf beendet und das System kehrt zurück ins Hauptfenster.

Testfenster

Sobald ein Test startet, wird vom normalen Programmfenster auf das Testfenster umgeschaltet. Die Bereiche oben rechts teilen sich die Schritt-Info bzw. die Debug-Info, die Anzeige der aktuellen Kontaktierposition und die Prüflingsübersicht für die aktuelle Kontaktierposition. Falls der Nutzen nur eine Kontaktierposition besitzt, wird die Anzeige der Kontaktierposition ausgeblendet.

Bild 1-5
Testfenster

Der **Ausgabebereich** lässt sich zwischen Prüfschrittausgaben, der Anzeige der ausgeführten Prüfschritte und der Debuganzeige hin und her schalten. Über die **Knopfleiste** lassen sich im SBS Mode Prüfschritte steuern und je nach Passwortlevel können die Parameter der Schritte editiert werden. Prüfschritte können über die Knöpfe abgebrochen, weitergeschaltet (Step+), zurückgeschaltet (Step-) und wiederholt werden.
Der Ablaufmodus lässt sich über den Mode-Button (von SBS zu Automatik und umgekehrt) einstellen, dies kann auch während des Ablaufs geschehen.
Der Interpretereditor (Prüfschritt Quellcode) sowie der Parametereditor (Parameter-Daten für Prüfschritte) kann aufgerufen werden.
Zusätzliche Hilfsmittel und Informationen lassen sich mit dem Debugger und den Tools aufrufen. Ablaufschritte können wiederholt werden, um beispielsweise Messergebnisse zu überprüfen, indem die Messungen erneut durchgeführt werden. Über die Fehlereinstellung erhält der Anwender die Möglichkeit das Verhalten im Fehlerfall zu beeinflussen.
Das **Step- und Debuginfo-Fenster** bietet die Übersicht über Details zum Step, wie Ober- und Untergrenzen, Prüfdauer und die aktuelle Prüfdauer.

2. Schritt für Schritt zum eigenen Prüfablauf

Nach dem Sie nun erste Erfahrungen mit dem TestManager sammeln konnten, stellt sich die Frage: **Wie erstellt man einen Prüfablauf?**
Diese Frage wird nun Schritt für Schritt anhand von anschaulichen Beispielen beantwortet.
Der Prüfablauf aus **Übung 1-1** hat noch recht wenig mit Messtechnik und Prüfungen zu tun, ist aber recht leicht nachzuvollziehen, da es sich um mathematische Operationen handelt. Auf diesen Ablauf wird zunächst aufgebaut, um Ihnen den Aufbau, die Struktur und die einzelnen Elemente näher zu bringen, bevor wir in die Messtechnik einsteigen.

Typfamilie

Das Programm erlaubt den Test verschiedenster Prüflingsarten. Jede Prüflingsart benötigt zu ihrem Test einen eigenen Satz von Daten: Ablauf des Tests, Umfang des Tests, Grenzwerte, Prüfschritte und so weiter. Dieser Satz von Daten wird als Typfamilie bezeichnet. Jede Typfamilie besteht aus einer oder mehreren Typvarianten. Die Typfamilie stellt den Überbegriff der Datensätze dar. Unter ihr können ähnliche Typvarianten zusammengefasst werden. Ein denkbares Beispiel hierfür wäre eine Prüflingsart, die in verschiedenen Varianten (Typvarianten) vorliegt (z.B. ein Autoradio für PKW (12V) oder LKW (24V) und in der Gesamtheit somit unter einer Typfamilie verwaltet werden kann.

■ Übung 2-1
Anlegen einer neuen Typfamilie

Im ersten Schritt zum eigenen Prüfablauf muss eine eigene Typfamilie erstellt werden. In dieser Übung soll die Vorhandene 0MATH abgeleitet werden. Das Ableiten von Typfamilien stellt die Funktion dar, eine bestehende Typfamilie kopieren zu können. Das System legt eine neue Typfamilie an, deren Namen gewählt werden kann und deren Inhalt alle Mitglieder der Ursprungsfamilie umfasst. Die Familie 0MATH bleibt somit erhalten und unverändert und steht weiterhin zur Verfügung.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort **„admin“** eingeben.

2. Öffnen Sie das Menü **Typen und Varianten bearbeiten** über die ➡ **Symbolleiste** ➡ **Type Edit**.

3. Markieren Sie die Typfamilie **0MATH** im linken Sichtfenster des Typen und Varianten bearbeiten-Fensters und wählen Sie in der ➡ **Symbolleiste** den Button ➡ **Typfamilie von bestehender Familie ableiten** aus.

4. Geben Sie einen neuen Familiennamen (Beispiel: 1MATH) ein und bestätigen diesen mit dem Button ➡ **OK**.

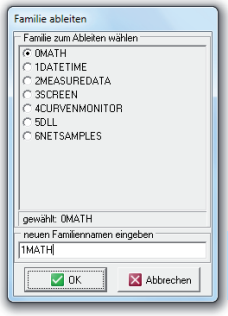


Bild 2-1
Typfamilie ableiten

> **Hinweis:**
> Der Name der Typfamilie wird für die Erstellung von Dateien und Verzeichnisse verwendet. Hieraus resultiert die strenge Prüfung der Namen.

5. Ändern Sie den **Typ – Code** (Beispiel: 01 Math), da zwei Varianten mit dem gleichen Namen nicht zulässig sind. Das Feld **Typ – Information** können Sie ebenfalls ändern (Beispiel: Zufallszahl), dies ist aber nicht zwingend notwendig.

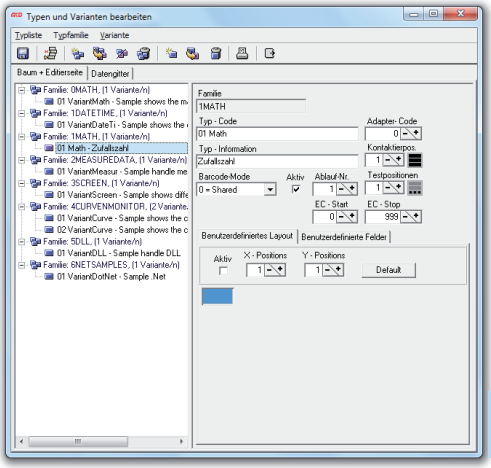


Bild 2-2
Typenliste

6. Speichern Sie die Typenliste ab.
   ➡ **Symbolleiste** ➡ **Typenliste speichern**

Typfamilien können über das gerade verwendete Fenster (**Bild 2-2**) auch neu angelegt, umbenannt und gelöscht werden.
Mit der Funktion neu anlegen, die über die ➡ **Symbolleiste** und den Eintrag ➡ **neue Typfamilie anlegen** gewählt werden kann, wird eine neue Typfamilie ohne Inhalt angelegt.
Wählen Sie in der ➡ **Symbolleiste** den Button ➡ **Typfamilie umbenennen,** um einer bereits vorhandenen Familie einen neuen Namen zu geben.
Die im linken Sichtfenster aufgelisteten Typfamilien lassen sich über die **Symbolleiste** ➡ **Typfamilie löschen** entfernen. Die links markierte Familie wird mit dieser Funktion gelöscht.
Änderungen übernehmen Sie über die ➡ **Symbolleiste** ➡ **Typenliste speichern**.

Typvariante

Eine Typvariante ist eine eindeutig bestimmbare Art von Prüflingen. Anhand der Daten einer Typvariante ist der genaue Ablauf der Prüfung durch das TestManager-System festgelegt. Sollten sich zwei Prüflinge hinsichtlich der Art der Prüfung unterscheiden, so bietet es sich an, sie in zwei verschiedenen Typvarianten zu definieren.

Hinweis:
Über die Variantenauswahl können Sie ähnliche Prüflinge mit einem Ablauf verwalten. Die geringen Unterschiede können dann über die Variantenfilter gewählt werden.

■ Übung 2-2
Verwalten von Varianten

Um einen eigenen Ablauf erstellen zu können, legen Sie nun eine neue Variante in der **Übung 2-1** abgeleiteten Typfamilie (1MATH) an.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Varianten anlegen:
   Varianten können im „**Typen und Varianten bearbeiten**“-Menü angelegt werden ➡ **Symbolleiste** ➡ **Type Edit.**

3. Im linken Fenster eine Typfamilie auswählen in der eine neue Variante angelegt werden soll (aus Übung 2-1: 1MATH) ➡ **Symbolleiste**
➡ **neue Variante anlegen**

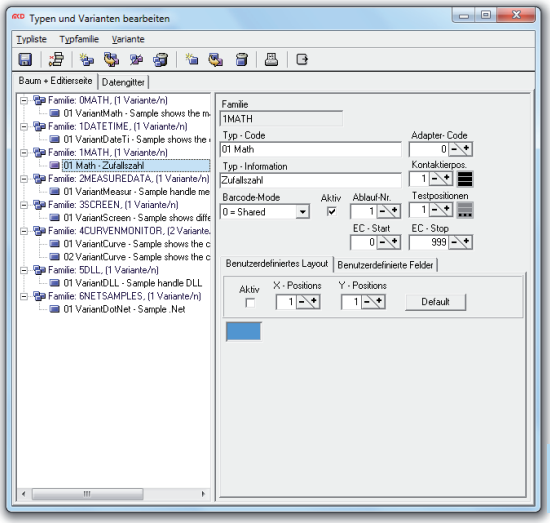


Bild 2-3
Variante anlegen

4. ➡ **Typ – Code** (Beispiel: 02 Math), ➡ **Typ – Information** (Beispiel: Zahlenvergleich) eingeben und die Variante aktiv mit einem Haken ➡ **aktiv** setzen.

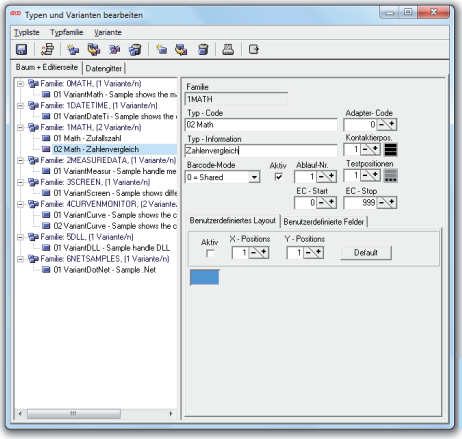


Bild 2-4
Variante aktivieren

5. Typenliste speichern um Änderungen zu übernehmen.
➡ **Symbolleiste** ➡ **Typenliste speichern**

Weitere Funktionen, die bei Varianten angewendet werden können, sind Kopieren und Umbenennen. Eine Variante die kopiert werden soll, muss im linken Fenster markiert und die Funktion ausgewählt werden. ➡ **Symbolleiste ➡ Variante kopieren.**
Das Löschen von Varianten erfolgt ebenfalls über die Auswahl der Variante im linken Fenster und der Wahl der Funktion ➡ **Symbolleiste ➡ Variante löschen.**
Nun besitzt der eigene Prüfablauf eine Typfamilie und zwei Varianten.

Prüfablauf

Tests laufen nach einem festgelegten Ablauf ab. Der Prüfablaufeditor kann im Hauptfenster über die ➡ **Symbolleiste ➡ Test Edit** oder über die ➡ **Menüleiste ➡ Editieren ➡ Prüfablauf editieren** aufgerufen werden.
Im Prüfablaufeditor werden beim Öffnen die Daten der jeweils gerade geladenen Typfamilie angezeigt und zum Bearbeiten bereitgestellt. Sollen die Daten einer anderen Familie bearbeitet werden, so muss entweder eine Variante der Familie in der Typauswahl geladen werden oder die Typfamilie wird im Prüfablaufeditor selektiert.

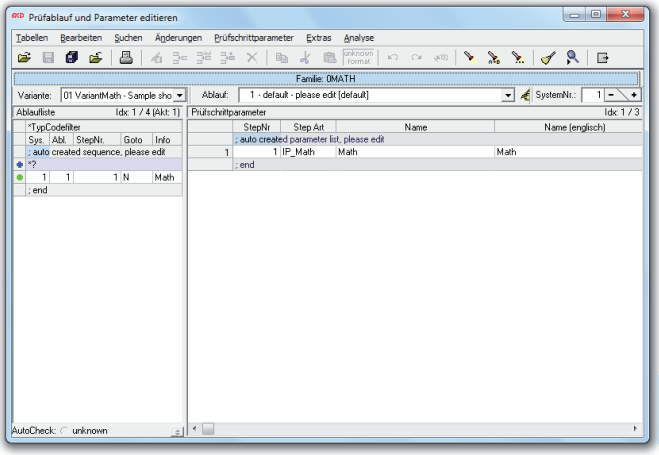


Bild 2-5
Prüfablaufeditor

Das Fenster des Prüfablaufeditors besteht von oben nach unten aus der Menüleiste, der Symbolleiste und dem eigentlichen Bearbeitungsfeld. Werden mehrere Abläufe geladen, so legt der Prüfablaufeditor für jeden Ablauf eine Registerkarte an, zwischen denen gewechselt werden kann.
Die Registerkarte für jede Familie besteht aus der Aktivierungsleiste mit den Auswahlfeldern für die Variante, den Ablauf und die Systemnummer, sowie darunter den drei Bereichen mit der Ablaufliste, dem Parameter-Inspektor und der Prüfschrittliste.

Wichtig:
Beim Bearbeiten der Ablaufliste oder der Prüfschritte wirken sich Änderungen immer auf alle Varianten einer Typfamilie aus, da sich alle Varianten diese Daten teilen. Dies ist im Normalfall durchaus erwünscht, da so eine Änderung, die alle Varianten betrifft, nur einmal gemacht werden muss. Jedoch kann bei Änderungen auch unabsichtlich eine andere Variante verändert werden.

Die Ablaufliste und die Prüfschrittliste haben eine unbegrenzte Undo-Funktion. Bearbeitungsschritte können jeweils bis zum letzten Speichern zurückgenommen werden.

Ablaufliste

Die Ablaufliste legt fest, welche Prüfschritte in einem Test ausgeführt werden und in welcher Reihenfolge. Dabei kann die Ablaufliste mehr als einen Prüfablauf und Prüfabläufe von mehr als einer Typvariante beinhalten. Welche der Schritte in der Ablaufliste in den eigentlichen Prüfablauf übernommen werden, ist von mehreren Filtern abhängig. Diese Filter sind: Typ-Code der Typ-Variante, Ablaufnummer und Systemnummer.
Eine Zeile der Ablaufliste kann entweder einen Kommentar, einen Codefilter, einen Ablaufschritt oder einen Fehler enthalten.
Kommentare werden vom Programm nicht weiter beachtet und können zur Beschreibung von Abläufen genutzt werden.
Typcodefilter aktivieren oder deaktivieren nachfolgende Ablaufschritte. Sie dienen als Kriterium zur Auswahl, welche Varianten welche Ablaufschritte beinhalten. Einschließende Typcodefilter beginnen mit einem (*) und werden mit einem Plus-Symbol in der ersten Spalte gekennzeichnet. Ausschließende Typcodefilter beginnen mit einem (^) und werden mit einem Minus-Symbol gekennzeichnet.

Ablaufschritte

Ein Prüfablauf setzt sich aus den Ablaufschritten zusammen.

	*TypCodefilter				
	Sys.	Abl.	StepNr.	Goto	Info
	; auto created sequence, please edit				
+	*?				
●	1	1	1	N	Math
	; end				

Bild 2-6
Ablaufschritte

Die ermittelte Aktivität des Ablaufschrittes wird in der ersten Spalte angegeben. Die **Systemnummer** dient der Filterung und legt fest, auf welchem System der Schritt ausgeführt werden soll. Sie muss mit der in der Grundeinstellung festgelegten Systemnummer übereinstimmen. Die Systemnummer „0“ wird für Schritte verwendet, die bei allen Systemen ausgeführt werden.

Die **Ablaufnummer** erlaubt es, in einer Ablaufliste mehrere Prüfabläufe zu speichern. Über die in der Typliste abgelegte Ablaufnummer wird festgelegt, welcher Ablauf geladen wird. Es werden nur solche Ablaufschritte in den Prüfablauf aufgenommen, deren Nummer mit der in der Typliste identisch ist oder deren Ablauf-Nummer 0 ist.
Die **Prüfschrittnummer** stellt die Verbindung von Ablaufliste zu den eigentlichen Prüfschritten her. Wird der Ablaufschritt in den Prüfablauf geladen, so werden die Daten des jeweiligen Prüfschrittes hinzugeladen und der Prüfschritt an der entsprechenden Stelle ausgeführt.
Die **Sprungart** legt fest, ob und wann nach der Ausführung eines Prüfschrittes die Test-Sequenz verlassen und ein Sprung ausgeführt werden darf.

Sprungarten:

N	Never	Niemals springen, Angabe für Ziel wird ignoriert
P	Pass	Sprung bei gut
F	Fail	Sprung bei schlecht oder ungültig
A	Always	Immer springen, Ergebnis ist egal
C	Calculated	Der Interpreterstep berechnet, ob gesprungen wird

Als Sprungziel wird eine Prüfschrittnummer angegeben. In der Ablaufliste werden Sprungart und Sprungziel gemeinsam in einer Spalte, getrennt durch einen Bindestrich angezeigt (Beispiel: 'P – 1000').

Prüfschritte

In den Prüfschritten wird definiert, wie eine bestimmte Aktion ausgeführt wird. Die Prüfschritte können in einem Ablauf mehrmals verwendet werden. Ein Prüfschritt besteht aus mehreren Feldern, von denen die Prüfschrittnummer die Verknüpfung zur Ablaufliste darstellt.

Prüfschrittparameter — Idx: 1 / 3

	StepNr	Step Art	Name	Name (englisch)
	; auto created parameter list, please edit			
1	1	IP_Math	Math	Math
	; end			

Bild 2-7
Prüfschritte

Felder der Prüfschrittparameter:

StepNr: Stellt die Verbindung zur Ablaufliste dar

StepArt: Name des aufzurufenden Interpreterschrittes

Name/Name: (deutsch/englisch) Beschreibung des Prüfschrittes

Einheit: Maßeinheit des Messwertes

NKS: Anzahl der Nachkommastellen

Obergrenze/ Untergrenze:	Grenzen in denen sich der Messwert bewegen darf
MDE:	Messdatenerfassung (ein/aus)
FRepeat:	Anzahl der Wiederholungen des Prüfschritts bei Fehlern
FRecover:	Fehlerbeseitigung, Fehlerbehandlung
FContinue:	Legt fest, ob im Fehlerfall die Prüfung fortgesetzt wird
Name Parameter x / Wert Parameter x:	Frei nutzbares Paar aus Parametername und Wert, das an den Interpreterschritt übergeben wird.

■ Übung 2-3
Prüfablauf erstellen

Inhalt dieser Übung ist es, einen Prüfablauf für die Typfamilie 1MATH (aus Übung 1-1 bis 3.) zu erstellen. Dieser soll es ermöglichen, für die beiden Varianten 01 Math und 02 Math verschiedene Abläufe zu erhalten.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Navigieren Sie über die ➡ **Symbolleiste** ➡ **Typwahl** ➡ **Typ wählen** ins Typauswahlfenster und wählen Sie dort den Ablauf der Typfamilie 1MATH aus.

3. Um nun den Ablauf anzupassen navigieren Sie über die ➡ **Symbolleiste** ➡ **Test Edit** in den Prüfablaufeditor.
   Fügen Sie einen neuen Prüfschritt hinzu, indem Sie die Ablaufliste markieren und über die ➡ **Symbolleiste** ➡ **Neuer Prüfschritt** wählen.

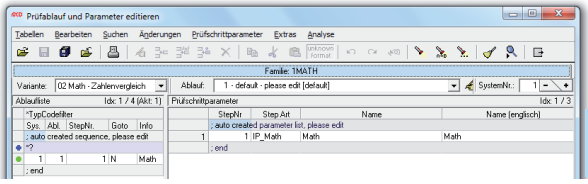


Bild 2-8
Neuer Prüfschritt

4. Geben Sie dem Schritt die ➡ **StepNr.** ➡ **2.** Ziehen Sie den Prüfschritt in der Ablaufliste mit gedrückter linker Maustaste an die gewünschte Position.

*TypCodefilter				
Sys.	Abl.	StepNr.	Goto	Info
; auto created sequence, please edit				
*?				
1	1	1	N	Math
1	1	2	N	undefined StepNo
; end				

Bild 2-9
Schritt 2

Das + über dem ersten Prüfschritt symbolisiert einen einschließenden Varianten-filter, das * steht für alle Typen. Diese Zeile sollte immer am Anfang der Ablauf-liste stehen.

5. Nun muss mit Hilfe von Variantenfiltern die Ablaufliste so angepasst werden, dass Step Nummer 1 nur bei der Variante 01 Math und Step 2 nur bei Variante 02 Math aufgerufen wird. Um neue Variantenfilter einzufügen markieren Sie die Ablaufliste und wählen Sie in der ➡ **Symbolleiste** ➡ **Neuer Variantenfilter**. Dieser Filter soll ein ausschließender (excludierender) Filter werden und muss im Feld hinter dem + Symbol ➡ **^02 Math** eingefügt werden. Platzieren Sie den Filter nun direkt hinter den ersten einschließenden Filter.

*TypCodefilter				
Sys.	Abl.	StepNr.	Goto	Info
; auto created sequence, please edit				
*?				
^02 Math				
1	1	1	N	Math
1	1	2	N	undefined StepNo
; end				

Bild 2-10
Neuer Variantenfilter

Es sind nun alle unter diesem Filter aufgelisteten Schritte nur für die Variante 01 Math aktiv.

6. Step Nummer 2 soll jedoch nur für die Variante 02 Math aktiv sein. Es könnte nun wieder ein ausschließender Filter verwendet werden.

> **Tipp:**
> Es bietet sich bei Typfamilien mit mehreren Varianten aufgrund der besseren Lesbarkeit an, einschließende (includierende) Filter zu verwenden und die Varianten im Filter aufzuzählen, für die die folgenden Schritte aktiv sein sollen.

Einfügen und anschließendes platzieren vom Variantenfilter über ➡ **Symbolliste** ➡ **Neuer Variantenfilter**. Editieren Sie den Filter mit ➡ ***02 Math**.

*TypCodefilter				
Sys.	Abl.	StepNr.	Goto	Info
; auto created sequence, please edit				
*?				
^02 Math				
1	1	1	N	Math
*02 Math				
1	1	2	N	Vergleich
; end				

Bild 2-11
Variantenfilter platzieren

7. Fügen Sie nun einen neuen Prüfschritt zur Prüfschrittliste hinzu. Markieren Sie hierzu die Prüfschrittliste und wählen Sie dann ➡ **Symbolleiste** ➡ **Neuer Prüfschritt**.

Editieren Sie die Felder des neuen Prüfschritts:

StepNr	➡ **2**
StepArt	➡ **IP_Vergleich**
Name	➡ **Vergleich**
Name	➡ **comparison**
Akt.	➡ **Ja**
Obergrenze	➡ **4**
Untergrenze	➡ **0**

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
	; auto created parameter list, please edit								
1	1	IP_Math	Math	Math	Ja	rad	2	1,57	-1,57
2	2	IP_Vergleich	Vergleich	comparison	Ja	rad	0	4	0
	; end								

Bild 2-12 Prüfpara.

8. Übernehmen Sie nun die Änderungen. ➡ **Symbolleiste** ➡ **Alles speichern.** Schließen Sie den Prüfablaufeditor.

Interpreter

Der Interpreter führt Interpreterschritte (Text-Dateien) aus, die vom Anwender angelegt und editiert werden können. Wie schon zu Beginn des Buches erwähnt, ist die Interpretersprache ähnlich PASCAL und BASIC aufgebaut. Enthalten sind Prozeduren, Funktionen und Bibliotheken. Die verwendeten Datentypen sind im wesentlichen Real und String-Typen.

Kurz erklärt:

Prozedur:
Als Prozedur wird ein Stück zusammengehörigen Quellcodes bezeichnet. Dieser ermöglicht es, wiederkehrende Aufgaben in einer wieder verwendbaren Art zu editieren. Auf diese Prozedur kann dann wiederholt zugegriffen werden. Prozeduren liefern kein Resultat oder Rückgabewert.

Funktionen:
Eine Funktion stellt ein Programmierkonzept ähnlich der Prozedur dar, das ein Resultat zurückliefert. Auf bereits vorhandene Funktionen kann zurückgegriffen werden.

Bibliotheken:
Eine Bibliothek stellt im Grunde eine Sammlung von Funktionen und Prozeduren dar. Wird eine Bibliothek in einen Quellcode eingebunden, stehen dem Programmierer

die in ihr enthaltenen Funktionen und Prozeduren zur Verfügung. Andernfalls ist der Programmierer gezwungen, benötigte Funktionen/Prozeduren selbst zu editieren.

Datentypen Real:
Der Datentyp Real wird für alle numerischen Werte verwendet (Gleitkomma-Zahlen, Ganz-Zahlen, logische und binäre Werte).

String:
Der String wird für Zeichenketten verwendet.

Vector:
Im Vector ist ein Datentyp realisiert, der ein eindimensionales Array für Real-Zahlen darstellt. Ein Array ist ein aus mehreren Objekten zusammengesetzter Datentyp, auf dessen verschiedene Elemente zugegriffen werden kann.

StringVector:
Stellt ein eindimensionales Array von Strings dar.

ComObject:
Das ComObject ist ein „Pseudo-Datentyp“ um ein ComObject (ActiveX-Object) zu definieren.

NetObject:
NetObject ist ein Pseudo-Datentyp zur Einbindung von .NET-Klassen in C#, VBB oder Java.

Interpretereditor

Der Interpretereditor ist über die ➡ **Symbolleiste** ➡ **IP Edit** oder über die ➡ **Menüleiste** ➡ **Editieren** ➡ **Interpreterschritte editieren** zu erreichen. Der Interpretereditor wird zur Eingabe des Quellcodes verwendet.

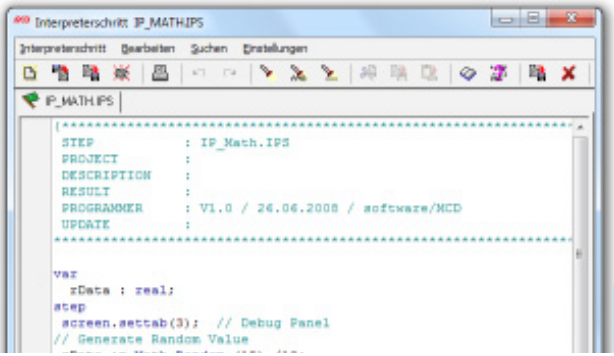


Bild 2-13
Interpretereditor

Interpreterhilfe

Die Interpreterhilfe kann über die Funktionstaste ➡ **F1** aufgerufen werden. Suchen Sie zu einem bestimmten Befehl, einer Funktion oder Prozedur Hilfe, können Sie diesen Begriff markieren und ➡ **STRG + F1** drücken. Mit dieser Tastenkombination springen Sie direkt zu diesem Befehl, falls dieser in der Hilfe enthalten ist.

Debugger

Der TestManager verfügt über einen Debugger für den Interpreter, der es erlaubt den Quellcode befehls- oder blockweise abzuarbeiten. Der Debugger gewährt ebenfalls Einsicht in Daten (Variablen und Parameter) und ermöglicht deren Änderung.

■ Übung 2-4
Interpreterschritt editieren

Diese Übung dient dem Kennenlernen der Programmiersprache und ist daher sehr einfach gehalten. Zum ersten eigenen Prüfablauf sind Sie nun nur noch eine Übung weit entfernt. Es fehlt lediglich das Editieren eines Interpreterschritts (IP-Step). Sollten Sie während dieser Übung Fragen zum Quellcode beziehungsweise zur Syntax haben, können Sie jederzeit die Interpreterhilfe zurate ziehen.

Der erste eigene Prüfablauf soll eine Zufallszahl zwischen 0 und 9 generieren. Weiter soll geprüft werden, ob diese Zufallszahl größer oder kleiner als 4 ist und entsprechend soll eine Ausgabe erfolgen.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Navigieren Sie über die ➡ **Symbolleiste** ➡ **Typwahl** ins Typwahlfenster und wählen Sie den Ablauf der Variante ➡ **02 Math** aus.

3. Öffnen Sie den Interpretereditor über die ➡ **Symbolleiste** ➡ **IP Edit** und wählen Sie im Interpretereditorfenster über die ➡ **Symbolleiste** ➡ **Leere Seite** ein leeres Dokument.

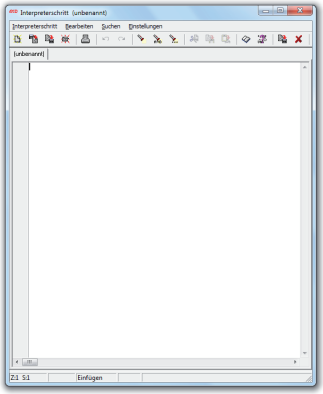


Bild 2-14
IP Edit

Speichern Sie den Step über die ➡ **Menüleiste** ➡ **Interpreterschritt** ➡ **speichern als**. Der Name des Interpreterschrittes muss dem entsprechen, der im Ablaufplan angegeben wurde: ➡ **IP_Vergleich**. Geben Sie den Namen ein und bestätigen Sie mit ➡ **Speichern**.

4. Definieren einer Variablen:
Variablen werden vor dem eigentlichen Step definiert. Es ergibt sich also grundsätzlich folgender Aufbau:

Syntax: *IP_VERGLEICH.IPS*

```
var
     //hier werden Variablen definiert

step
     //hier wird der eigentliche Quellcode editiert
end. //hier endet der Quellcode
```

Realisieren Sie nun den Aufbau in ihrem leeren Dokument und definieren sich nach folgender Syntax eine Variable:

Eine Variable setzt sich aus einem Bezeichner und einem Typen zusammen. Der Bezeichner definiert den Namen der Variablen.

Syntax:

```
Bezeichner : Typ;
```

In Ihrem Fall müsste der **Quellcode** wie folgt eingegeben sein:

```
var
  Zufallszahl : Real;

step

end.
```

Sie haben sich nun eine Variable mit dem Namen Zufallszahl definiert. Über diesen Namen können Sie im Quellcode jederzeit auf den Inhalt vom Typ 'real' zugreifen.

5. Generieren einer Zufallszahl:
Um eine Zufallszahl zu generieren kann folgende Funktion genutzt werden:

```
Math.Random (rRange : real) : real;
```

Der Name der Funktion ist also '**Math.Random**' und in Klammern muss ein Wert vom Typ 'real' eingegeben werden, der den Bereich der Zufallszahl definiert. Der Bereich soll von 0 bis 9 definiert werden. Die Angabe des zweiten 'real' gibt den Typ des Rückgabewertes an, den die Funktion liefert. Diese Zufallszahl soll der Variablen zugewiesen werden. Zuweisungen werden wie folgt realisiert:

Zum Beispiel:

```
Variable := Wert;
```

oder in unserem Fall:

```
Variable := Funktion();
```

Quellcode: *IP_VERGLEICH.IPS*

```
var
  Zufallszahl : Real;

step
  Zufallszahl := Math.Random (9);
end.
```

Möchten Sie aufgrund der Nachvollziehbarkeit zusätzliche Informationen in den Quellcode schreiben, können Sie das über Kommentare realisieren. Kommentaren werden // vorangestellt und haben die Besonderheit, dass sie vom Compiler nicht übersetzt werden.

Quellcode:

```
//Dies ist ein Kommentar!
```

6. Zufallszahl vergleichen:
Ein Vergleich kann über eine so genannte If-Then-Anweisung (If-Then-Else) realisiert werden:
Auf das 'If' folgt eine Logikanweisung, auf die reagiert wird. Nach 'Then' folgt der Code, der ausgeführt werden soll, wenn die Logikanweisung 'true' (wahr) liefert, nach Else der Code für die Rückgabe von 'fail' (falsch).
Es soll geprüft werden, ob die Zufallszahl größer oder kleiner gleich 4 ist und das Ergebnis soll im Debugfenster ausgegeben werden.

Verbundanweisungen setzen sich aus einer Folge von Anweisungen zusammen, die in der genannten Reihenfolge ausgeführt werden. Die in einer Verbundanweisung enthaltenen Teilanweisungen werden von den Wörtern 'begin' und 'end' eingeschlossen.

Quellcode: *IP_VERGLEICH.IPS*

```
var
  Zufallszahl : Real;

step
  Zufallszahl := Math.Random (9);

  If Zufallszahl > 4 Then begin
     Debug.Show(0,'Zufallszahl = ',Zufallszahl);
     Debug.Show(0,'Zufallszahl ist größer als 4');
  end
  Else begin
     Debug.Show(0,'Zufallszahl = ',Zufallszahl);
     Debug.Show(0,'Zufallszahl ist kleiner gleich 4');
  end;
end.
```

Tipp:
Wenn Sie wissen möchten, was die **'Debug.show'** Funktion bewirkt und wie sie eingegeben wird, dann markieren Sie die Funktion und drücken die Tasten STRG + F1.

7. In diesem Schritt soll die Zufallszahl im Messwertfenster des Prüfablaufs angezeigt werden. Zudem soll eine kurze Verzögerungszeit eingefügt werden.

Quellcode: *IP_VERGLEICH.IPS*

```
var
  Zufallszahl : Real;

step
//***************Erzeugen einer Zufallszahl********************
  Zufallszahl := Math.Random (9);
//*************************************************************

//***************Abfrage der Größe mit Ausgabe:****************
  If Zufallszahl > 4 Then begin
     Debug.Show(0,'Zufallszahl = ',Zufallszahl);
     Debug.Show(0,'Zufallszahl ist größer als 4');
  end
  else begin
     Debug.Show(0,'Zufallszahl = ',Zufallszahl);
     Debug.Show(0,'Zufallszahl ist kleiner gleich 4');
  end;
//*************************************************************

  SetValue(Zufallszahl);        //Zufallszahl wird als Messwert
  DateTime.Delay(2000);         //zurückgegeben
                                //Zeitverzögerung von 2 sec.
  Repeat                        //Diese Schleife wird solange
  until StepContinue;           //wiederholt, bis das
                                //StepContinue-Ereignis erfolgt
end.
```

8. Speichern und schließen Sie den Interpreterschritt und starten Sie den Ablauf.

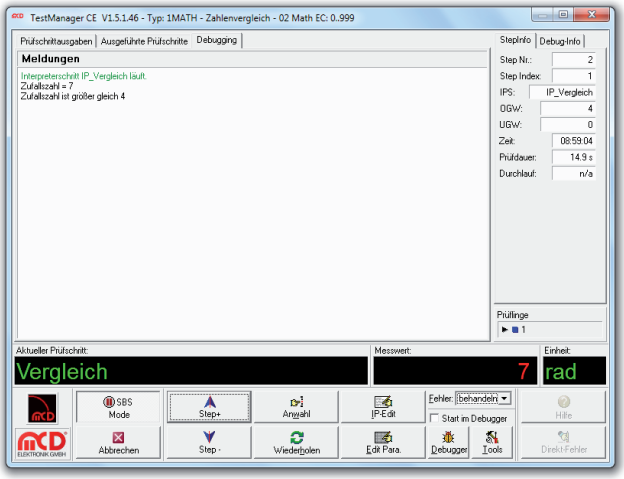


Bild 2-15
Ablauf starten

Aufgrund der Grenzwerte wird der Prüfablauf 'Fail' (durchgefallen), weil eine Zufallszahl generiert wurde, die größer ist als 4 ist.

3. Screen-Objekte

Bildschirmausgaben lassen sich über die so genannten Screen-Objekte initialisieren. Unter diese Kategorie fallen Buttons, Bitmaps, Curve (Kurven), Dialoge, Edit (Eingabefelder), Frames (Rahmen), Labels, Meter, uvm..
Solche Objekte lassen sich über folgende Syntax initialisieren:

Screen.Objektname, Beispiel:

```
Screen.Dialog
```

Diese Objekte besitzen Prozeduren um sie beispielsweise zu aktivieren, verbergen, anzuzeigen uvm.

Beispiel:

```
Screen.Dialog.Show
```

Weitere Informationen zur Syntax können über die Interpreterhilfe nachgelesen werden.

Vor der Verwendung von Screen-Objekten, müssen alle Objekte verborgen und auf Vorgabewerte zurückgesetzt werden. Wollen Sie Objekte darstellen, müssen Sie zunächst ein benutzerdefinierbares Fenster öffnen und dieses anzeigen. Es ergibt sich folgende Syntax, die vor der Benutzung von Screen-Objekten editiert werden muss:

```
Screen.ClrScr;  //Objekte verbergen und auf Vorgabewerte zurücksetzen
Screen.Show;    //Benutzerdefinierbares Fenster öffnen und anzeigen
```

■ Übung 3-1
Screen-Objekte

Diese Übung zielt auf den Umgang mit Screen-Objekten.
Das Erstellen eines Ablaufs, bei dem zwei Buttons auf dem Testfenster ausgegeben werden, ist die Aufgabe dieser Übung. Diese sollen die Beschriftung 'Pass' und 'Fail' tragen. Mit Hilfe dieser Buttons soll das Prüfergebnis wie folgt beeinflusst werden können: Beim Betätigen des Buttons Pass soll das Prüfergebnis auf bestanden (Pass) gesetzt werden. Beim Aktivieren von 'Fail' soll das Prüfergebnis so angepasst werden, dass es als durchgefallen (Fail) gilt.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Erstellen Sie eine neue Typfamilie mit dem Namen ➡ **8MONITOR** und einer Variante ➡ **01 Button**. Setzen Sie diese aktiv und geben Sie einen Namen für die Typ-Information ein (Beispiel: 2 Button).

3. Wählen Sie die in Schritt 2. angelegte Variante über die Typwahl aus.

4. Editieren Sie einen Schritt in der Ablauf- und Prüfschrittparameterliste. Um einen neuen Interpreterschritt verwenden zu können, geben Sie einen neuen Namen im Step Art Feld ein (Beispiel: IP_Button). Legen Sie Ober- und Untergrenze bei einem bestimmten Wert fest, um das Prüfergebnis beeinflussen zu können.

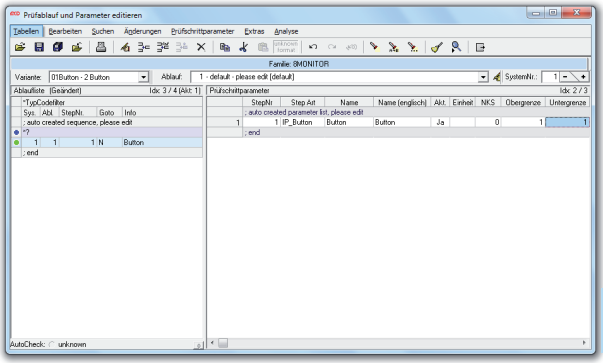


Bild 3-1
Screen-Objekte

5. Interpreterschritt editieren:
Editieren Sie ihren Quellcode, indem Sie sich eine Variable definieren, der Sie den Wert des zuletzt gedrückten Buttons übergeben können und initialisieren Sie den Monitor. Definieren Sie sich in einer 'repeat until' Schleife die beiden Buttons und fragen Sie ab, welcher Button gedrückt wurde. Die Schleife soll erst dann verlassen werden können, wenn einer der beiden Buttons aktiviert wurde. Passen Sie je nach Knopf den Messwert an, um das Ergebnis des Ablaufs ('Pass' oder 'Fail') beeinflussen zu können.

Quellcode: *IP_BUTTON.IPS*

```
var
  Knopf : Real;
step
//****************Initialisierung Screen:********************
  Screen.SetTab(1);
  Screen.ClrScr;
  Screen.Show;
//*************************************************************
```

➡ *weiter geht´s auf der nächsten Seite*

```
  //*********Schleife zum Wiederholen mit Wertanpassung:*******
 repeat
  Screen.Button.Show( 1, 100, 100, 'Pass', 100, 1, 24);
  Screen.Button.Show( 2, 300, 100, 'Fail', 100, 1, 24);

  Knopf := Screen.LastButton;

  If Knopf = 1 Then
  begin
     SetValue(1);
  end;

  If Knopf = 2 Then
  begin
     SetValue(100);
  end;

  GlobalVar.Set ('gKnopf', Knopf);

 until Knopf > 0;
//*************************************************************

 repeat

  DateTime.Delay(300);

 until StepContinue;

 Screen.ClrScr;
end.
```

Die Schleife repeat … until StepContinue ermöglicht im Step by Step – Modus, dass der Quellcode der Schleife solange wiederholt wird, bis der Sprungbefehl zum nächsten Step (über den Step+ Button) erfolgt.

4. Globale Variable

Variablen die in einem Step definiert und initialisiert werden, gehen nach dem Ende des Steps verloren. Es ist jedoch in vielen Fällen notwendig, dass Variablen und deren Werte über einen oder mehrere Steps, möglicherweise bis zum Ablaufende erhalten bleiben. Globale Variablen verfügen über diese Eigenschaft. Ihre Lebensdauer geht über das Stepende hinaus und kann bis hin zum Systemende definiert werden.
Diese Globalen Variablen können über die Prozedur ➡ **GlobalVar.Set** definiert und mit der Funktion ➡ **GlobalVar.Get** abgerufen werden. Sie müssen, wie normale Variablen, über einen eindeutigen Bezeichner verfügen.
Variablen, Werte und Kommentare lassen sich über die ➡ **Menüleiste** ➡ **Tools** ➡ **Globale Daten** einsehen und verändern.

Datentypen wandeln

Der TestManager stellt Funktionen zum umwandeln von Datentypen zur Verfügung. Diese sind notwendig, damit beispielsweise Rückgabewerte von Funktionen, die in einem festen Datentyp vorliegen, umgewandelt werden können.
Mit der Funktion **Val** lassen sich Strings in einen Real Datentyp umwandeln. Umgekehrt ist es über die Funktion **Str** möglich, einen Real-Wert in einen String zu wandeln.

Programmieren leicht gemacht

Für das Editieren von Interpreterschritten empfiehlt es sich, die Variante, für die der Interpreterschritt editiert werden soll, auszuwählen und den Ablauf im SBS-Modus zu starten. Ist noch kein IP-Step mit dem entsprechenden Namen angelegt, wird der Interpretereditor mit dem Pfad, unter dem der IP-Step erwartet wird, geöffnet. Der Quellcode kann direkt in diesem Fenster editiert werden. Beim Schließen des Editors wird der Code direkt ausgeführt. Im Fall eines Fehlers kann sich der Programmierer diesen direkt im Editor anzeigen lassen. Die Fehlerzeile wird im Interpretereditor rot angezeigt. In der Statusleiste wird der Fehlercode mit Beschreibung angezeigt.
Es kann vorkommen, dass der Fehler bereits weiter oben im Quellcode steckt, dies ist zum Beispiel bei der Verwendung einer nicht definierten Variable der Fall. Der Fehler kann somit direkt behoben und der verbesserte Quellcode erneut ausgeführt werden. So kann die Richtigkeit des Codes und das Verhalten des Programms überprüft werden. Der Quellcode kann dadurch programmiererfreundlich schrittweise editiert und überprüft werden.

Hinweis:
Eine weitere Eigenart des Interpreters ist, dass manche Fehler erst zur Laufzeit erkannt werden können. Die Syntax im Editor wird als 'OK' geprüft. Im echten Start des Prüfschritts werden die Funktionen dann tatsächlich ausgeführt und auf Typengleichheit kontrolliert.

Während den Übungen werden Ihnen Schritt für Schritt Funktionen und Prozeduren zur Hand gegeben, mit denen Sie sich womöglich zum ersten Mal konfrontiert sehen. Sollte Ihnen die Syntax oder die Wirkung dieser Befehle unklar sein, können Sie die Interpreterhilfe zurate ziehen.
Dies sollte Ihre Standardvorgehensweise sein, denn hierdurch lernen Sie selbstständig mit dem TestManager umzugehen und Interpreterschritte selbst zu erstellen.

■ Übung 4-1
Glücksspiel

In dieser Übung soll ein neuer Ablauf (Glücksspiel) angelegt werden. Dieser soll aus 3 Prüfschritten bestehen und 4 vom Anwender eingegebene Zahlen (0-9) mit 4 vom Programm generierten Zufallszahlen vergleichen. Das Ergebnis soll in Prozent angegeben werden und die Übereinstimmungen der Zahlen wiederspiegeln.

Die Namen der Prüfschritte lauten:
- Eingabe von Zahlen
- Zufall
- Vergleichen

Mit den zugehörigen Interpreterschritten:
- IP_Eingabe
- IP_Zufall
- IP_Vergleichen

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Legen Sie eine neue Typfamilie mit dem Namen ➡ **7GLUECKSSPIEL** mit der Variante ➡ **01 Zufallszahl** an und setzen Sie diese ➡ **aktiv**.

3. Editieren Sie den Prüfablauf mit den drei oben genannten Schritten und definieren Sie sinnvolle Messwertgrenzen. Beachten Sie, dass das Ergebnis in Prozent ausgegeben werden soll.

Ablaufliste Idx: 1 / 6 (Akt: 3)

	Sys.	Abl.	StepNr.	Goto	Info
	*TypCodefilter				
	; auto created sequence, please edit				
	*?				
	1	1	1	N	Eingabe von Zahlen
	1	1	2	N	Zufall
	1	1	3	N	Vergleich
	; end				

Bild 4-1
Ablaufliste

Prüfschrittparameter Idx: 1 / 5

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
	; auto created parameter list, please edit								
1	1	IP_Eingabe	Eingabe von Zahlen		Ja		0	1	1
2	2	IP_Zufall	Zufall		Ja		0	1	1
3	3	IP_Vergleichen	Vergleich		Ja	%	0	100	25
	; end								

Bild 4-2
Prüfschrittliste

4. Wählen Sie über die Typwahl die in Schritt 2. angelegte Variante aus.

5. Editieren des ersten Interpreterschritts:
   Definieren Sie zunächst die Variablen vom Typ Real, denen Sie die vom Anwen der eingegebenen Zahlen zuweisen können. Eine weitere Variable vom Typ Real soll als Zähler fungieren und muss daher ebenfalls definiert werden.

6. Weisen Sie dem Zähler einen Anfangswert zu (es bietet sich 0 an) und initialisieren Sie den Bildschirm.

7. Erstellen Sie ein Eingabefenster auf dem Prüfschrittausgabefenster und aktivieren Sie es.

8. Editieren Sie eine repeat until Schleife, die mit Hilfe des Zählers so lange wiederholt wird, bis dieser einen Wert von Anfangswert + 4 erreicht. Der Sinn dieser Schleife besteht darin, dass ihr Inhalt so lange wiederholt wird, bis der Zähler seinen Endwert erreicht. Innerhalb der Schleife lassen sich somit die vier Benutzerzahlen einlesen und an die Variablen übergeben.

9. Fragen Sie innerhalb der repeat until Schleife mit Hilfe einer IF-Then Anweisung ab, ob ein gültiges Zeichen in das Eingabefeld eingegeben wurde. Falls dies der Fall sein sollte, kann der Button zum Bestätigen angezeigt und aktiviert werden.

10. Innerhalb der Schleife definieren Sie sich eine weitere If-Then Anweisung, mit der Sie abfragen, ob ein Button aktiviert wurde. Im Innern der Anweisung können Sie mit Hilfe einer Case Anweisung, bei der Sie den Zähler als Selektor verwenden, die einzelnen Zahlen den Variablen zuweisen. Aktivieren Sie hiernach erneut das Eingabefeld und leeren Sie dieses. Der Button soll wieder ausgeblendet werden. Wichtig ist, dass Sie den Zähler inkrementieren.

11. Definieren Sie die eingegebenen Zahlen als Globale Variablen und setzen Sie den Messwert.

Quellcode: *IP_EINGABE.IPS*

```
var
  Zahl1       : Real;
  Zahl2       : Real;
  Zahl3       : Real;
  Zahl4       : Real;
  Zaehler     : Real;

step
//*****************Screen initialisieren:**********************
  Screen.SetTab(1);       //Fokus auf Prüfschrittausgabe
  Screen.ClrScr;
  Screen.Show;
//*************************************************************

//*********Eingabefenster erstellen und aktivieren:************
  Screen.Edit.Setup (1, 1, '0..9', 0);    //Eingabefenster
  Screen.Edit.Show ( 1, 100, 100, 50);
  Screen.Edit.Activate (1);
//*************************************************************

//*******************Zähler auf 0 stellen:*********************
  Zaehler := 0;
//*************************************************************

 repeat      //Wiederholen bis Zähler 4 ist!

  Screen.Label.Show (1, 100, 30, 'Bitte geben Sie eine Zahl
  ein:');

  If Screen.Edit.GetText( 1 ) <> '' Then //Abfrage ob Zeichen
                                          eingegeben wurde
  begin
     Screen.Button.Show( 1, 200, 100, 'Ok', 100, 1, 24);
     Screen.Button.Activate (1);
  end;

  If Screen.LastButton =1 Then  //Wenn Button gedrückt wurde,
                                  dann:
  begin

//******************Auswahl nach Zählerstand:****************
     Case Zaehler Of
       0 : Zahl1 := Val (Screen.Edit.GetText( 1 ),100);
       1 : Zahl2 := Val (Screen.Edit.GetText( 1 ),100);
       2 : Zahl3 := Val (Screen.Edit.GetText( 1 ),100);
       3 : Zahl4 := Val (Screen.Edit.GetText( 1 ),100);
     end;
//*************************************************************

//*************Zähler hochzählen und Button verbergen:*********
     Screen.Edit.Activate (1);
     Screen.Edit.SetText (1, '');
     Screen.Button.Hide (1);
     Zaehler := Zaehler + 1;
```

➡ *weiter geht´s auf der nächsten Seite*

```
//*********************************************************************
  end;

 until (Zaehler = 4);

//**************Lable und Eingabefeld verbergen:***************
 Screen.Label.Hide (1);
 Screen.Edit.Hide(1);
//*************************************************************

//*****************Globale Variablen definieren:****************
 GlobalVar.Set ('gZahl1', Zahl1);//Zahlen in Globalen Variablen
                                  speichern
 GlobalVar.Set ('gZahl2', Zahl2);
 GlobalVar.Set ('gZahl3', Zahl3);
 GlobalVar.Set ('gZahl4', Zahl4);
//*************************************************************
 SetValue(1);        //Messwert setzen

 repeat
 until StepContinue;

 Screen.ClrScr;      //Bildschirm löschen
end.
```

12. IP_Zufall editieren:
Lassen Sie sich 4 Zufallszahlen generieren und speichern Sie diese in Globalen Variablen ab.

Quellcode: *IP_ZUFALL.IPS*

```
var
  Zufallszahl1 : Real;
  Zufallszahl2 : Real;
  Zufallszahl3 : Real;
  Zufallszahl4 : Real;
step
//****************Zufallszahlen generieren:********************
  Zufallszahl1 := Math.Random (10);
  Zufallszahl2 := Math.Random (10);
  Zufallszahl3 := Math.Random (10);
  Zufallszahl4 := Math.Random (10);
//*************************************************************

//***************Globale Variablen setzen:*********************
  GlobalVar.Set ('gZufallszahl1', Zufallszahl1);
  GlobalVar.Set ('gZufallszahl2', Zufallszahl2);
  GlobalVar.Set ('gZufallszahl3', Zufallszahl3);
  GlobalVar.Set ('gZufallszahl4', Zufallszahl4);
//*************************************************************

  SetValue (1);
```

➡ *weiter geht´s auf der nächsten Seite*

```
  repeat
  until StepContinue;
end.
```

Tipp:
Achten Sie bei der Namensgebung der globalen und normalen Variablen darauf, dass keine Missverständnisse entstehen können und die Syntax nachvollziehbar ist.

13. IP_Vergleichen editieren:
Prüfen Sie mit Hilfe von If-Then Anweisungen, ob es Übereinstimmungen zwischen den eingegebenen Zahlen und den Zufallszahlen gibt. Passen Sie gegebenenfalls das Ergebnis innerhalb der Anweisung entsprechend an. Geben Sie die Zahlen für den Anwender auf dem Prüfschrittausgabefenster aus.

Quellcode: *IP_VERGLEICHEN.IPS*

```
var
  Ergebnis : Real;

step
  Ergebnis :=  0;

//**********Zahlen vergleichen und Ergebnis anpassen:**********
  If GlobalVar.Get ('gZahl1') = GlobalVar.Get ('gZufallszahl1')
  Then
  begin
     Ergebnis := Ergebnis + 25;
  end;

  If GlobalVar.Get ('gZahl2') = GlobalVar.Get ('gZufallszahl2')
  Then
  begin
     Ergebnis := Ergebnis + 25;
  end;

  If GlobalVar.Get ('gZahl3') = GlobalVar.Get ('gZufallszahl3')
  Then
  begin
     Ergebnis := Ergebnis + 25;
  end;

  If GlobalVar.Get ('gZahl4') = GlobalVar.Get ('gZufallszahl4')
  Then
  begin
     Ergebnis := Ergebnis + 25;
  end;
//*************************************************************

     Screen.SetTab (1);
     Screen.ClrScr;
     Screen.Show;
```

➡ *weiter geht´s auf der nächsten Seite*

```
//*****************Ausgabe der Zahlen:*************************
     Screen.Label.Show (1, 100,  30, 'Ihre Zahlen:');
     Screen.Label.Show (2, 100,  60, str (GlobalVar.Get
     ('gZahl1')));
     Screen.Label.Show (3, 100,  90, str (GlobalVar.Get
     ('gZahl2')));
     Screen.Label.Show (4, 100, 120, str (GlobalVar.Get
     ('gZahl3')));
     Screen.Label.Show (5, 100, 150, str (GlobalVar.Get
     ('gZahl4')));

     Screen.Label.Show (6, 250, 30, 'Zufallszahlen:');
     Screen.Label.Show (7, 250, 60, str (GlobalVar.Get
     ('gZufallszahl1')));
     Screen.Label.Show (8, 250, 90, str (GlobalVar.Get
     ('gZufallszahl2')));
     Screen.Label.Show (9, 250, 120, str (GlobalVar.Get
     ('gZufallszahl3')));
     Screen.Label.Show (10, 250, 150, str (GlobalVar.Get
     ('gZufallszahl4')));
//*************************************************************

     Screen.SetTab (1);
     SetValue (Ergebnis);

     repeat
       DateTime.Delay (2000);
     until StepContinue;


     Screen.ClrScr;
end.
```

14. Speichern Sie Ihre editierten Daten ab und starten Sie den Ablauf.

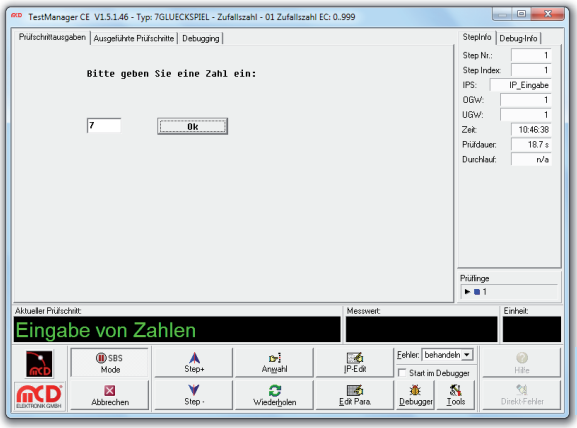


Bild 4-3
Eingabe von Zahlen

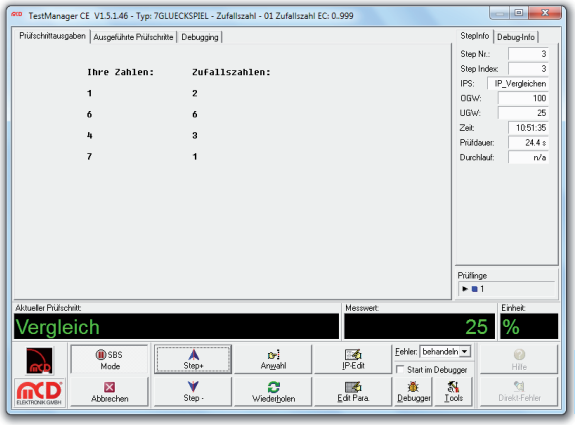


Bild 4-4
Vergleich

5. Tools

Wie schon zu Beginn des Buchs erwähnt, enthält der TestManager eine Vielzahl von Funktionen und Tools. Über die ➡ **Menüleiste** ➡ **Tools** können die in den Grundeinstellungen aktivierten Tools aufgerufen werden. Diese Tools ermöglichen dem Anwender Informationen zu erhalten und Einstellungen vorzunehmen und können auch außerhalb von Abläufen verwendet werden.
Unter dem Menüpunkt Tools finden sich verschiedene Monitore, wie der CAN-, LIN- und IC-Monitor, sowie diverse ME- (Meilhaus) Monitore. Mit diesen Monitoren können beispielsweise Signale gesendet und empfangen werden. Einstellungsmöglichkeiten aller Art sind ebenfalls möglich.
Diese Tools lassen sich auch in Abläufen über die Funktion ➡ **Action.Trigger** aufrufen.

■ Übung 5-1
Verwendung von Tools

Diese Übung befasst sich mit der Verwendung von Tools. Als Beispiel für die Verwendung von Tools, wird für diese Übung der Kurvenmonitor genutzt. Es sollen Daten für drei Kurven an den Monitor übergeben werden. Die mathematischen Funktionen Sinus, Kosinus und Sinus*Kosinus sollen als Ergebnis auf dem Kurvenmonitor ausgegeben werden.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**Admin**“ eingeben.

2. Legen Sie eine neue Typfamilie mit dem Namen ➡ **9KURVENMONITOR** mit der Variante ➡ **01SinusKosinus** an und setzen Sie die Variante auf ➡ **Aktiv**. Speichern Sie ihre neue Typfamilie und Variante ab.

3. Wählen Sie über die ➡ **Typwahl** die gerade angelegte Typfamilie aus.

4. Prüfablauf:
   Editieren Sie einen Prüfablauf mit einem Prüfschritt ➡ **IP_SINUS**, definieren Sie Ober- und Untergrenzen und setzen Sie den Step aktiv.

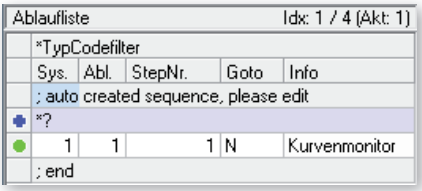


Bild 5-1
Ablaufliste

Prüfschrittparameter — Idx: 1 / 5

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
	; auto created parameter list, please edit								
1	1	IP_Eingabe	Eingabe von Zahlen		Ja		0	1	1
2	2	IP_Zufall	Zufall		Ja		0	1	1
3	3	IP_Vergleichen	Vergleich		Ja	%	0	100	25
	; end								

Bild 5-2
Prüfschrittparameter

5. Interpreterschritt:
   Legen Sie sich 3 Variablen vom Typ **Vector** an. Diese Vektoren dienen der Ablage der Funktionswerte von Sinus, Kosinus und Sinus*Kosinus. Des Weiteren wird eine Variable zum Zählen benötigt, die vom Typ real sein sollte. Diese soll es ermöglichen, die Funktionswerte in einzelnen Zellen des Vectors zu schreiben (vergleichbar mit dem bekannten Datentyp Array).

6. Bevor die Zellen der Vektoren beschrieben werden, sollte sichergestellt sein, dass alle Zellen ohne Inhalt sind. Somit haben diese Vektoren eine 'Länge' von 0. Dies geschieht über die Syntax:
   Beispiel: `Kurve1 := [];`

7. Als nächstes sollten Sie eine Schleife generieren, mit der Sie die Zellen des Vektors einzeln beschreiben können. Hierfür bietet sich eine For-To-Do Schleife an, mit der die Vektoren mit beliebig vielen Werten beschrieben werden können. Sie sollten darauf achten, dass nach jedem Durchlauf eine Zelle des Vektors 'weitergeschaltet' wird.

   Quellcode: *IP_SINUS.IPS*

```
For Zaehler := 1 To 1000 Step 1 Do begin
     Kurve1 := Kurve1 + [(Math.Sin( Zaehler / 100 )) ];
end;
```

   Mit [(`Math.Sin( Zaehler / 100 )`)] wird die aktuelle Zelle mit dem Sinuswert vom Zählerstand / 100 beschrieben. Wobei die Klammer [den Anfang und das Ende] der Zelle symbolisiert. Über die Syntax Kurve1 := Kurve1 + []; wird der Vektor mit sich selbst und einer weiteren (neuen) Zelle [] überschrieben. Die Länge des Vektors wird somit um ein Element vergrößert.

   Dieses Vorgehen von 6. und 7. wiederholen Sie für den Kosinus und Sinus*Kosinus.

8. Einstellungen des Kurvenmonitors:
   Die Achsen des Kurvenmonitors können Sie mit der Prozedur ➡ **Curve.Scale.SetText** benennen. Es bieten sich die Achsennamen Volt für die Y-Achse und Messpunkte für die X-Achse an. Die Kurvennamen können Sie mit der Prozedur ➡ **Curve.Name** in den Kurvenmonitor einfügen und über ➡ **Curve.Setup** den Monitor konfigurieren.

9. Anzeigen der Kurven:
Die Kurven können mit Hilfe der Funktion ➡ **Action.Trigger** im Kurvenmonitor ausgegeben werden. Hierzu wird der ➡ **Action-Code** des Kurvenmonitors (2020) benötigt. Setzen Sie noch den Messwert auf den von Ihnen definierten Wert.

Quellcode: *IP_SINUS.IPS*

```
var
  Kurve1  : Vector;
  Kurve2  : Vector;
  Kurve3  : Vector;
  Zaehler : Real;

step

//***************Vectoren mit Werten füllen:******************
  Kurve1 :=[];
  For Zaehler := 1 To 1000 Step 1 Do begin
     Kurve1 := Kurve1 + [(Math.Sin( Zaehler / 100 )) ];
  end;

  Kurve2 :=[];
  For Zaehler := 1 To 1000 Step 1 Do begin
     Kurve2 := Kurve2 + [Math.Cos ( Zaehler / 100)];
  end;

  Kurve3 :=[];
  For Zaehler := 1 To 1000 Step 1 Do begin
     Kurve3 := Kurve3 + ([Math.Sin ( Zaehler / 100)*Math.Cos
               (Zaehler / 100)]);
  end;
//*************************************************************

//***********Achsenbezeichnungen und Kurvennamen definieren:****
  Curve.Scale.SetText ( 1, 1, 'Volt' );
  Curve.Scale.SetText ( 1, 0, 'Messpunkte' );

  Curve.Name( 1, 'Kurve 1', 'Kurve 2','Kurve 3');
//*************************************************************

//****************Ausgabe der Kurven:***************************
  Curve.Setup ( 1, 1000, Kurve1, Kurve2, Kurve3);
  Action.Trigger ( 2020);
//*************************************************************

  SetValue (1);

  repeat
  until StepContinue;

end.
```

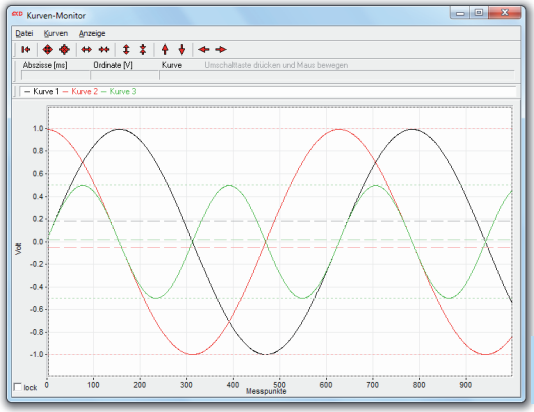


Bild 5-3
Ausgabe im Kurvenmonitor

6. Files

Der TestManager bietet die Möglichkeit Files anzulegen, auf sie zuzugreifen und diese zu bearbeiten. Es können verschiedene Funktionen und Prozeduren aufgerufen werden, die beispielsweise das Lesen und Schreiben in Files ermöglichen. Zum Aufruf dieser Funktionen wird folgende Syntax verwendet:

```
➡ File.(Funktion oder Prozedur)
```

Bei den Funktionen und Prozeduren für Files ist zu beachten, ob das File zur Ausführung geöffnet oder geschlossen sein muss. So muss beispielsweise ein File geöffnet sein, bevor aus ihm gelesen oder in es geschrieben werden soll.

■ Übung 6-1
Schreiben und Lesen von Files

Diese Übung befasst sich mit dem Lesen und Schreiben von Files. Dieser Ablauf soll einen vom Anwender eingegebenen Text in ein Textdokument (Endung.txt) schreiben und anschließend wieder auslesen. Der ausgelesene Text soll im Prüfschrittausgabefenster angezeigt werden.

Schritt für Schritt:

1. Im ersten Schritt melden Sie sich als Administrator an (falls Sie das nicht bereits getan haben). Unter ➡ **Symbolleiste** ➡ **Passwort** und im Eingabefenster das Passwort „**admin**“ eingeben.

2. Neue Typfamilie mit einer Variante anlegen und aktiv setzen.

   Beispiel: Typfamilienname: 11UEBUNG9
   Typ-Code: 01Txt-File
   Typ-Information: Schreiben und lesen

   Speichern Sie die editierten Daten ab.

3. Wählen Sie über die ➡ **Typwahl** die in Schritt 2. angelegte Variante der neuen Typfamilie aus.

4. Editieren Sie unter ➡ **Test Edit** die Ablaufliste und die Prüfschrittparameter. Es wird lediglich ein Schritt für den Ablauf benötigt. Wählen Sie einen Namen für die StepArt (Beispiel: IP_TXTFILE) und definieren Sie eine Ober- und Untergrenze. Speichern Sie anschließend ihre Daten ab.

Ablaufliste Idx: 1 / 4 (Akt: 1)

	Sys.	Abl.	StepNr.	Goto	Info
*TypCodefilter					
; auto created sequence, please edit					
*?					
	1	1	1	N	Lesen + Schreiben
; end					

Bild 6-1
Ablaufliste

Prüfschrittparameter Idx: 1 / 3

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
; auto created parameter list, please edit									
	1	1	IP_TXTFILE	Lesen + Schreiben	read + write		Ja	0	1	1
; end									

Bild 6-2
Prüfschrittparameter

5. Starten Sie den Ablauf und editieren Sie den Interpreterschritt. Verwenden Sie ein Eingabefeld und zum Bestätigen der Eingabe einen Button, mit dessen Hilfe der eingegebene Text einer Vaiablen übergeben werden kann. Speichern Sie den Text in einer Variablen ab.

6. Mit der Prozedur ➡ **File.SetFileName ()** kann ein File erstellt werden. Benennen Sie das File mit ➡ **'Übung9.txt'** und öffnen Sie anschließend das File mit Hilfe der Prozedur ➡ **File.Open()**.

Syntax:

```
File.SetFileName (1, 'Übung9.txt');
File.Open (1);
```

7. Schreiben Sie mit Hilfe Ihrer angelegten Variablen den Text in das File.

Syntax:

```
File.Write (1, Variable);
```

8. Um den Text auslesen zu können, müssen Sie zunächst den Filepointer wieder auf den Anfang des Files positionieren. Der Filepointer kann mit der Prozedur ➡ **File.Seek()** an beliebige Stellen des Files gesetzt werden. Der Inhalt des Files lässt sich mit der Funktion ➡ **File.Read()** auslesen und kann direkt an eine Variable übergeben werden. Nachdem Sie das File ausgelesen haben, sollten Sie es wieder mit dem Befehl ➡ **File.Close()** schließen.

Syntax:

```
File.Seek (1,0);      //Filepointer an den Anfang setzen
     Inhalt := File.Read (1, 'EOF',100);
     File.Close (1);
```

9. Geben Sie den Inhalt des Files im Prüfschrittausgabefenster aus und definieren Sie eine Wartezeit von ca. drei Sekunden, um dem Anwender Zeit zu geben die Ausgabe erfassen zu können.

10. Löschen Sie das File mit ➡ **File.Delete** und setzen Sie ihren Messwert innerhalb der von Ihnen definierten Grenzen.

Syntax:

```
File.Delete (1);      //Löschen der Datei!!!
```

Die Datei soll gelöscht werden, damit bei weiteren Anwendungen und der Eingabe von kürzeren Texten der im vorherigen Ablauf eingegebene Text nicht zum Teil mit ausgegeben wird.

> **Tipp:**
> Probieren Sie es einfach aus, mit und ohne Löschen der Datei.

Quellcode: *IP_TXTFILE.IPS*

```
var
  Text   : String;
  Inhalt : String;

step
//******************Screen initialisieren:**********************
  Screen.SetTab (1);
  Screen.ClrScr;
  Screen.Show;
//*************************************************************

//******************Ausgabe und Eingabefeld generieren:*********
  Screen.Label.Show (1, 30, 30, 'Bitte geben Sie Ihren Text
  ein:');
  Screen.Edit.Setup (1, 100,'' ,0);
  Screen.Edit.Show (1, 30, 60, 400);
  Screen.Edit.Activate (1);
//*************************************************************

//***********Wiederholen bis Button gedrückt wurde:*************
  repeat
     If Screen.Edit.GetText( 1 ) <> ,' Then
     begin
       Screen.Button.Show (1, 480, 60, 'OK',70 ,1 , 24);
     end;
  until Screen.LastButton =1;
//*************************************************************

 //**************Text in File schreiben:************************
  Text := Screen.Edit.GetText (1);

  File.SetFileName (1, 'Übung9.txt');
  File.Open (1);
  File.Write (1, Text);
```

➡ *weiter geht's auf der nächsten Seite*

```
  File.Seek (1,0);      //Filepointer an den Anfang setzen
  Inhalt := File.Read (1, 'EOF',100);
  File.Close (1);
//*************************************************************

//*****************Ausgabe:***********************************
  Screen.Label.Show (2, 30, 200, 'Text aus File:');
  Screen.Label.Show (3, 30, 230, Inhalt);
//*************************************************************

  DateTime.Delay (3000);

  repeat
  until StepContinue;

  File.Delete (1);      //Löschen der Datei!!!
  SetValue (1);

end.
```

11. Speichern Sie den Interpreterschritt ab und starten Sie den Ablauf im SBS-Modus. Sollten Sie das Löschen der Datei nach der 'repeat until StepContinue' Schleife editiert haben, können Sie nach der Eingabe Ihres Textes und der Bestätigung, das Textdokument einsehen, bevor es gelöscht wird. Hierzu müssen Sie in den Windows Explorer in das ➡ **Verzeichnis** navigieren indem Sie den TestManager installiert haben. Das File befindet sich dort im Ordner ➡ **TypData** ➡ **TYPE_**(der von Ihnen eingegebene **Typfamilienname**). Sie können somit nachvollziehen, ob das Schreiben in das File erfolgreich war.

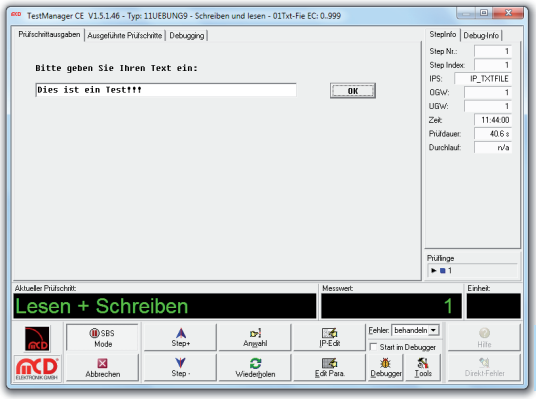


Bild 6-3
Lesen + Schreiben

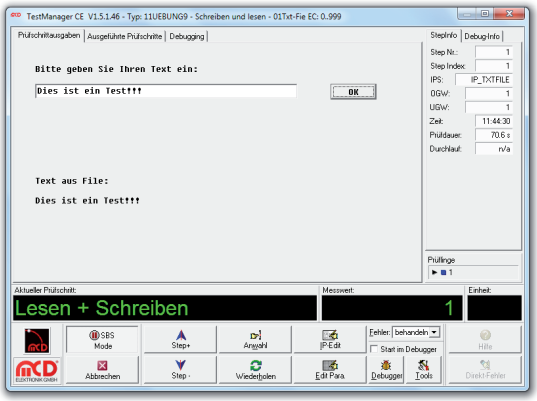


Bild 6-4

Lesen + Schreiben

■ Übung 6-2

Schreiben, Lesen, Umbenennen von Files und Kennenlernen der Date-Time Funktionen

Diese Übung umfasst ein weiteres Beispiel im Umgang mit Files. Die Aufgabe ist es einen Ablauf zu erstellen in dem der Anwender aufgefordert wird, einen Benutzernamen einzugeben. Dieser soll in einem Textdokument mit dem aktuellen Datum und Zeit hinterlegt werden.
Im zweiten Schritt soll der aktuelle Anwender und der vorherige Anwender ausgegeben werden, der in einem zweiten Textdokument ebenfalls mit Datum und Zeit hinterlegt sein soll.
Im dritten Schritt soll das Textdokument mit dem vorherigen Anwender durch das Texdokument mit dem aktuellen Anwender überschrieben werden. Dies kann über das Umbenennen der Files realisiert werden.

Schritt für Schritt:

1. Stellen Sie sicher, dass Sie beim System als Administrator angemeldet sind. Sollte dies nicht der Fall sein, melden Sie sich bitte an.

2. Legen Sie eine neue Typfamilie mit einer Variante an, setzen Sie diese auf aktiv und speichern Sie die Daten ab.

   Beispiel:

Typfamilienname:	12UEBUNG10
Typ-Code:	01File
Typ-Information:	txt Datei

3. Wählen Sie die angelegte Variante über die Typwahl aus.

4. Erstellen Sie drei Schritte in der Ablaufliste und Prüfschrittparameterliste. Verwenden Sie drei neue Step Arten, definieren Sie Namen und Grenzen für die Schritte und speichern Sie ihre Daten ab.

Ablaufliste Idx: 1 / 6 (Akt: 3)

*TypCodefilter

Sys.	Abl.	StepNr.	Goto	Info
; auto created sequence, please edit				
*?				
1	1	1	N	Textfie schreiben
1	1	2	N	Textfile lesen
1	1	3	N	Textfile umbenennen
; end				

Bild 6-5
Ablaufliste

Prüfschrittparameter Idx: 1 / 5

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
; auto created parameter list, please edit									
1	1	IP_FILE1	Textfie schreiben	write file	Ja		0	1	1
2	2	IP_GETUSER	Textfile lesen	read file	Ja		0	1	1
3	3	IP_FILE2	Textfile umbenennen	rename file	Ja		0	1	1
; end									

Bild 6-6
Prüfschrittparam.

5. Wählen Sie über die ➡ **Typwahl** die in Schritt 3. angelegte Variante und starten Sie den Ablauf.

6. Editieren des ersten Interpreterschritts:
Legen Sie Variablen vom Typ Real zum Abspeichern des Datums (Jahr, Monat und Tag) und der Zeit (Stunden, Minuten und Sekunden) an. Eine Variable vom Typ String wird ebenfalls benötigt um den Namen des Anwenders hinterlegen zu können.

7. Erstellen Sie ein Eingabefeld auf dem Prüfschrittausgabefenster und aktivieren Sie dieses. Geben Sie in diesem Fenster eine Aufforderung vom Typ Label zum Eingeben des Benutzernamens aus.

8. Setzen Sie ein Button, mit dem eine Eingabe bestätigt werden kann, sobald ein Zeichen eingegeben wurde.

9. Speichern Sie den Benutzernamen in der dafür angelegten Variablen ab.

10. Lesen Sie die aktuelle Systemzeit mit der Funktion ➡ **DateTime.Time ()** aus. Mit dieser Funktion können die Rückgabewerte direkt den angegebenen Variablen zugewiesen werden.

Syntax:

```
DateTime.Time (Stunden, Minuten , Sekunden);
```

Stunden, Minuten und Sekunden sind in diesem Fall die Variablen, die die aktuelle Zeit zugewiesen bekommen.

11. Die Funktion ➡ **DateTime.Date ()** liefert das aktuelle Systemdatum und weist sie den Variablen (wie in Schritt 9.) zu.

Syntax:

```
DateTime.Date ( Jahr, Monat, Tag);
```

12. Öffnen Sie nun das Textdokument in dem der aktuelle Anwender hinterlegt werden soll.

13. Mit Hilfe der Prozedur ➡ **File.WriteLn()** kann zeilenweise in das Textdokument geschrieben werden. Schreiben Sie mit Hilfe dieser Prozedur den Anwender, das Datum und die Zeit in je eine Zeile.

Syntax:

```
File.WriteLn (1, 'User:  ' + Anwender);
File.WriteLn (1, 'Date:  ' + Str (Jahr) + '-' + Str (Monat) +
'-'+Str (Tag));
File.WriteLn (1, 'Time:  '+Str(Stunden) + ':' + Str (Minuten) +
':' + Str (Sekunden));
```

14. Schließen Sie das Textdokument und setzen Sie den Messwert.

Quellcode des ersten Interpreterschritts: *IP_FILES.IPS*

```
var
  Anwender    : String;
  Stunden     : Real;
  Minuten     : Real;
  Sekunden    : Real;
  Jahr        : Real;
  Monat       : Real;
  Tag         : Real;

step
//***************Screen initialisieren:************************
  Screen.SetTab (1);
  Screen.ClrScr;
  Screen.Show;
//*************************************************************

//************Eingabefeld erzeugen und aktivieren:*************
  Screen.Edit.Setup (1, 20,'' ,0);
  Screen.Edit.Show (1, 200, 100, 150);
  Screen.Edit.Activate (1);
//*************************************************************

//*************Benutzeraufforderung ausgeben:******************
  Screen.Label.Show (1, 20, 105, 'Benutzername eingeben:');
```

➡ *weiter geht´s auf der nächsten Seite*

```
//*************************************************************

//************Wiederholen bis Button gedrückt wurde:***********
  repeat
     If Screen.Edit.GetText( 1 ) <> ,“ Then
     begin
       Screen.Button.Show (1, 380, 100, 'OK',70 ,1 , 24);
     end;
  until Screen.LastButton  = 1;
//*************************************************************

//**************Text, Datum und Zeit holen:********************
  Anwender := Screen.Edit.GetText (1);

  DateTime.Time ( Stunden, Minuten , Sekunden);
  DateTime.Date ( Jahr, Monat, Tag);
//*************************************************************

//***************Zeilenweise in File schreiben*****************
  File.SetFileName (1, 'Anwender_aktuell.txt');
  File.Open (1);

  File.WriteLn (1, 'User:  ' + Anwender);
  File.WriteLn (1, 'Date:  ' + Str (Jahr) + '-' + Str (Monat) +
  '-' + Str (Tag));
  File.WriteLn (1, 'Time:  ' + Str (Stunden) + ':' + Str
  (Minuten) + ':' + Str (Sekunden));

  File.Close (1);

//*************************************************************

  SetValue (1);

  repeat
  until StepContinue;

end.
```

15. Editieren des zweiten Interpreterschritts:
    Legen Sie drei Variablen vom Typ String an, um die ausgelesenen Anwenderdaten zuweisen zu können.

16. Öffnen Sie das Textdokument des aktuellen Anwenders und legen Sie die zeilenweise ausgelesenen Daten in den Variablen ab.

Syntax:

```
Userdaten1 := File.ReadLn (1);
Userdaten2 := File.ReadLn (1);
Userdaten3 := File.ReadLn (1);
```

17. Schließen Sie das Dokument und geben Sie die Anwenderdaten in dem Prüfschrittausgabefenster aus.

18. Öffnen Sie das Dokument des vorherigen Anwenders, lesen Sie ebenfalls die Daten zeilenweise aus und hinterlegen Sie diese in den vorgesehenen Variablen.

19. Schließen Sie das Dokument und geben Sie die Daten aus. Setzen Sie den Messwert und eine Verzögerungszeit von ca. 2-4 Sekunden um dem Anwender das Lesen der Ausgabe zu ermöglichen.

20. Löschen Sie den Bildschirm.

Quellcode des zweiten Interpreterschritts: *IP_GETUSER.IPS*

```
var
  Userdaten1 : String;
  Userdaten2 : String;
  Userdaten3 : String;

step
//******************File öffnen:*******************************
  File.SetFileName (1, 'Anwender_aktuell.txt');
  File.Open (1);
//*************************************************************

//****************Zeilenweise auslesen:************************
  Userdaten1 := File.ReadLn (1);
  Userdaten2 := File.ReadLn (1);
  Userdaten3 := File.ReadLn (1);
//*************************************************************

//******************File schließen:****************************
  File.Close (1);
//*************************************************************

//******************Screen initialisieren:*********************
  Screen.SetTab (1);
  Screen.ClrScr;
  Screen.Show;
//*************************************************************

//******************Ausgabe der Daten:*************************
  Screen.Label.Show (4, 100, 40,  'aktueller Anwender:');
  Screen.Label.Show (1, 100, 70,   Userdaten1 );
  Screen.Label.Show (2, 100, 100,  Userdaten2 );
  Screen.Label.Show (3, 100, 130,  Userdaten3 );
//*************************************************************

//******************File öffnen:*******************************
  File.SetFileName (1, 'Anwender_vorher.txt');
  File.Open (1);
//*************************************************************
```

➡ *weiter geht's auf der nächsten Seite*

```
//***************Zeilenweise auslesen:*************************
  Userdaten1 := File.ReadLn (1);
  Userdaten2 := File.ReadLn (1);
  Userdaten3 := File.ReadLn (1);
//*************************************************************

//******************File schließen:****************************
  File.Close (1);
//*************************************************************

//******************Ausgabe der Daten:*************************
  Screen.Label.Show (5, 100, 180,  'vorheriger Anwender:');
  Screen.Label.Show (6, 100, 210,  Userdaten1 );
  Screen.Label.Show (7, 100, 240,  Userdaten2 );
  Screen.Label.Show (8, 100, 270,  Userdaten3 );
//*************************************************************

  SetValue (1);

  DateTime.Delay (2000);

  repeat
  until StepContinue;
  Screen.ClrScr;

end.
```

21. Editieren des dritten Interpreterschritts:
Mit der Prozedur ➡ **File.Move ()** lassen sich Files umbenennen. Nutzen Sie die Prozedur und ändern Sie den Namen des Dokuments des aktuellen Anwenders in den des vorherigen Anwenders. Somit wird zum Schluss des Ablaufs der aktuelle Anwender zum vorherigen Anwender. Beim erneuten Ausführen des Ablaufs wird die Datei des aktuellen Anwenders gelöscht und erneut erzeugt. Somit bleiben der aktuelle und der vorherige Anwender hinterlegt.

Syntax:

```
File.Move ('C:\...\TypeData\TYPE_12UEBUNG10\Anwender_aktuell.
txt', 'C:\...\Anwender_vorher.txt', True);
```

Die Prozedur erfordert die Angabe von Pfaden um die Dateien umzubenennen.

22. Setzen Sie den Messwert und eine Verzögerungszeit von ca. einer Sekunde.

Quellcode für den dritten Interpreterschritt: *IP_FILE2.IPS*

```
var

step

//******************File umbenennen:****************************
  File.Move ('C:\TestCollection\Testing\TestManager CE\TypeDa-
ta\TYPE_12UEBUNG10\Anwender_aktuell.txt', 'C:\TestCollection\
Testing\TestManager CE\TypeData\TYPE_12UEBUNG10\Anwender_
vorher.txt', True);
//*************************************************************
  SetValue (1);

  DateTime.Delay (1000);

  repeat
  until StepContinue;

end.
```

23. Speichern Sie Ihre Interpreterschritte ab und starten Sie den Ablauf

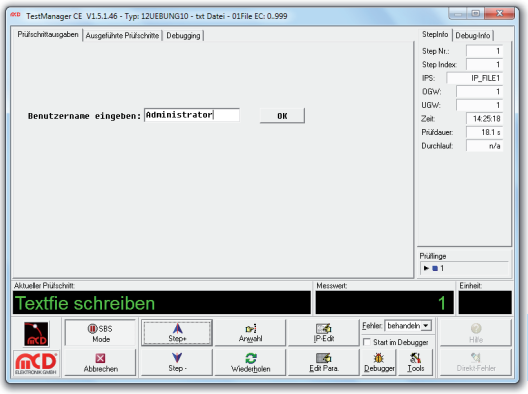


Bild 6-7
Ablauf starten - 1. Schritt

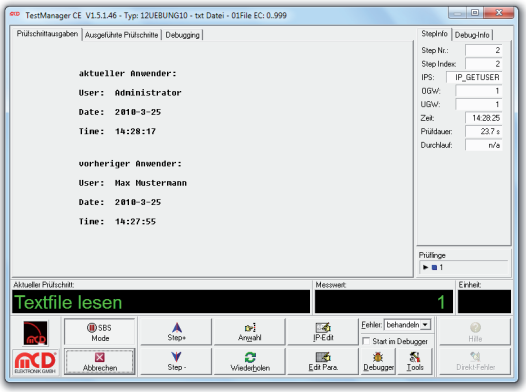


Bild 6-8
Ablauf starten - 2. Schritt

INI-File

Der TestManager unterstützt ebenfalls den Zugriff und das Editieren von INI-Dateien. Diese Dateien sind Initialisierungsdateien, die beispielsweise zum Abspeichern von Programmeinstellungen verwendet werden. INI-Files können über den TestManager eingelesen, sektionsweise eingelesen, beschrieben, gelöscht, sektionsweise gelöscht werden. Hierfür stehen Funktionen und Prozeduren zur Verfügung.

Ein INI-File besteht aus den Komponenten Sektion (eng.: Section), Elementen (eng.: Items) und Werten (eng.: Values). Die so genannten Sektionen stellen Überbegriffe dar, unter denen Elemente mit ihren Werten abgelegt werden können.

Beispielaufbau INI-File:
[Sektion1]
Item1 = Wert1
Item2 = Wert2

[Sektion2]
Item1 = Wert1
Item2 = Wert2

Item1 der Sektion1 und Item2 der Sektion2, sowie deren Werte können zufällig identisch sein, sind aber grundsätzlich verschieden zu behandeln.

■ Übung 6-3
Erstellen, Schreiben und Lesen von INI-Files

Diese Übung befasst sich mit INI-Files und soll Ihnen den Umgang mit diesen Dateien näher bringen. Die Aufgabe ist es einen Ablauf zu editieren, der sich aus zwei Schritten zusammensetzt.
Im ersten Schritt soll ein INI-File geöffnet bzw. erstellt werden. In dieses File sollen drei generierte Zufallszahlen (Länge, Breite und Höhe) in die Sektion Dimensionen geschrieben werden. Aus den generierten Zahlen soll das Volumen errechnet und in der Sektion Rechenwerte hinterlegt werden.
Das Auslesen und die Ausgabe der Daten des INI-Files soll im zweiten Schritt realisiert werden.

Schritt für Schritt:

1. Überprüfen Sie Ihren Passwortlevel, sollten Sie nicht als Administrator beim System angemeldet sein, ändern Sie dies.

2. Legen Sie eine neue Typfamilie mit einer Variante an, setzen Sie diese aktiv und speichern Sie die Daten ab.

   Beispiel: Familie: 13UEBUNG11
   Typ-Code: INIFile
   Typ-Information: Schreiben und Lesen

3. Wählen Sie die angelegte Variante über die Typwahl aus.

4. Fügen Sie in der Ablaufliste und in den Prüfschrittparametern einen weiteren Prüfschritt ein und legen Sie zwei neue Step Arten fest. Definieren Sie Grenzen, setzen Sie die Steps aktiv und speichern Sie die editieren Daten ab.

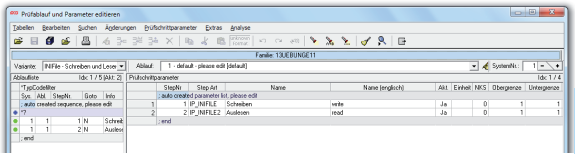


Bild 6-9
Prüfablauf editieren

5. Aktivieren Sie die erstellte Variante über die Typwahl und starten Sie den Ablauf.

6. Editieren des ersten Interpreterschritts:
   Legen Sie zwei Variablen vom Typ String an, um die Sektionsnamen zu hinterlegen und vier vom Typ Real, um die Werte abspeichern zu können.

7. Öffnen bzw. erstellen Sie ein INI-File mit Hilfe der Prozedur ➡ **INIFile.SetFileName(Pfad+Name.INI)**. Es bietet sich an einen Pfad zu wählen, der bei dem von Ihnen verwendeten Computer auf den Desktop, oder in das TestManager Installationsverzeichnis führt.

Syntax:

```
INIFile.SetFileName ('C:\...\TypeData\TYPE_13UEBUNG11\Übung11.
INI');
```

Hinweis:
Bitte beachten Sie, dass der von Ihnen verwendete Pfad von diesem unterscheiden kann.

8. Generieren Sie sich drei Zufallszahlen mit der ➡ **Math.Random()** Funktion und weisen diese den vorgesehenen Variablen zu.

9. Sektion ins INI-File schreiben:
Weisen Sie der Variablen für die Sektion den Namen ➡ **Dimensionen** zu. Mit Hilfe der Prozedur ➡ **INIFile.WriteReal**(Sektion, Item, Wert) können Sie einen Real Wert für ein Item in die Sektion schreiben.

Syntax:

```
Section1 := 'Dimensionen';
INIFile.WriteReal ( Section1,  '1: Länge', Wert1);
```

10. Berechnen Sie das Volumen und übergeben Sie den Wert an die von Ihnen hier für vorgesehene Variable.

11. Weisen Sie den Namen der zweiten Sektion der vorgesehenen Variablen zu und schreiben Sie das Volumen in dieser Sektion ins INI-File.

12. Setzen Sie den Messwert innerhalb der von Ihnen definierten Grenzen.

Quellcode: *IP_INIFILE.IPS*

```
var
  Section1    : String;
  Section2    : String;
  Wert1       : Real;
  Wert2       : Real;
  Wert3       : Real;
  Rauminhalt  : Real;

step
```

➡ *weiter geht´s auf der nächsten Seite*

```
//*************INIFile öffnen oder erstellen:******************
  INIFile.SetFileName ('C:\...\TypeData\TYPE_13UEBUNG11\
  Übung11.INI');
//************************************************************

//*****************Zufallswerte generieren:********************
  Wert1 := Math.Random (100);
  Wert2 := Math.Random (100);
  Wert3 := Math.Random (100);
//************************************************************

//*************Section1 in INIFile schreiben:******************
  Section1 := 'Dimensionen';

  INIFile.WriteReal ( Section1, '1: Länge', Wert1);
  INIFile.WriteReal ( Section1, '2: Breite', Wert2);
  INIFile.WriteReal ( Section1, '3: Höhe', Wert3);
//************************************************************

//***************Volumen errechnen:****************************
  Rauminhalt := Wert1 * Wert2 * Wert3;
//************************************************************

//*************Volumen in Section2 schreiben:******************
  Section2 := 'Rechenwerte';

  INIFile.WriteReal ( Section2,  '1: Volumen', Rauminhalt);
//************************************************************
  repeat
  until StepContinue;

  SetValue (1);
end.
```

13. Editieren des zweiten Interpreterschritts:
    Definieren Sie sich sechs Variablen vom Typ String. Zwei für die Zuweisung der Sektionen und vier für die Itemnamen.

14. Initialisieren Sie den Bildschirm für die Ausgabe von Screenobjekten.

15. Öffnen Sie das INI-File mit der Prozedur ➡ **INIFile.SetFileName**.

16. Sektionen auslesen:
    Lesen Sie die Sektionen über die Prozedur ➡ **INIFile.Load()** ein und setzen Sie den internen Zeiger mit dem Befehl ➡ **INIFile.First** auf die erste eingelesene Sektion. Nun kann der Sektionsname über die Funktion ➡ **INIFile.Name** direkt an die vorgesehene Variable übergeben werden.

Syntax:

```
INIFile.Load ('');
INIFile.First;          //Zeiger auf die erste Sektion stellen
Section1 := INIFile.Name;
```

Um die zweite Sektion auslesen zu können, muss der interne Zeiger auf das nächste Element gesetzt werden. Dies wird durch den Befehl ➡ **INIFile.Next** ausgelöst. Wiederum mit dem ➡ **INIFile.Name** Befehl kann der Sektionsname an die Variable übergeben werden.

Syntax:

```
INIFile.Next;           //Zeiger auf die nächste Section stellen
Section2 := INIFile.Name;
```

17. Items aus dem INI-File lesen:
Mit Hilfe des schon eben verwendeten Befehls ➡ **INIFile.Load**(Sektion) kann der Inhalt einer Sektion eingelesen werden. Sollen nun Elemente (Items) aus dieser Sektion ausgelesen werden, so wird ähnlich wie beim Auslesen der Sektionen vorgegangen. Der interne Zeiger muss nun auf das erste Element der Sektion gestellt werden. Hierfür wird wieder die Prozedur ➡ **INIFile.First** verwendet. Nun können Sie wieder mit der Funktion ➡ **INIFile.Name** den Itemname an eine Variable übergeben. Der Wert des Items lässt sich mit ➡ **INIFile.Value** einlesen. Soll dieser direkt an eine Real Variable übergeben werden, so müssen Sie den Typen mit ➡ **Val()** vom String in einen Real wandeln. Den internen Zeiger können Sie mit ➡ **INIFile.Next** auf das nächste Element stellen, um dieses auszulesen.

Syntax:

```
INIFile.Load (Section1);
INIFile.First;
Variable1 := INIFile.Name;
Wert1     := Val (INIFile.Value, 0);
INIFile.Next;
```

18. Gehen Sie wie in Schritt 16. vor um das Item aus der zweiten Sektion auszulesen.

Syntax:

```
INIFile.Load (Section2);
INIFile.First;
Rechenname := INIFile.Name;
Rechenwert := Val (INIFile.Value, 0);
```

19. Geben Sie die Variablen mit Hilfe der Screen-Funktionen im Prüfschrittausgabefenster aus und setzen Sie den Messwert innerhalb der von Ihnen definierten Grenzen.

Quellcode: *IP_INIFILE2.IPS*

```
var
  Section1    : String;
  Section2    : String;
  Variable1   : String;
  Variable2   : String;
  Variable3   : String;
  Wert1       : Real;
  Wert2       : Real;
  Wert3       : Real;
  Rechenwert  : Real;
  Rechenname  : String;

step
//*****************Screen initialisieren:***********************
  Screen.SetTab (1);       //Fokus auf Prüfschrittausgabe
  Screen.ClrScr;
  Screen.Show;
//************************************************************

//***************Öffnen oder Erzeugen des INIFiles:************
  INIFile.SetFileName ('C:\TestCollection\Testing\TestManager
  CE\TypeData\TYPE_13UEBUNG11\Übung11.INI');
//************************************************************

//****************Sectionsnamen werden ausgelesen:*************
  INIFile.Load ('');
  INIFile.First;       //Zeiger auf die erste Section stellen
  Section1 := INIFile.Name;    //Name des Eintrags auslesen,
                               auf die der Zeiger zeigt
  INIFile.Next;        //Zeiger auf die nächste Section stellen
  Section2 := INIFile.Name;
//************************************************************

//**********Items und Werte aus Section1 auslesen:*************
  INIFile.Load (Section1);
  INIFile.First;
  Variable1 := INIFile.Name;
  Wert1     := Val (INIFile.Value, 0);
  INIFile.Next;
  Variable2 := INIFile.Name;
  Wert2     := Val (INIFile.Value, 0);
  INIFile.Next;
  Variable3 := INIFile.Name;
  Wert3     := Val (INIFile.Value, 0);
//************************************************************

//************Items und Werte aus Section2 auslesen:***********
  INIFile.Load (Section2);
```

➡ *weiter geht's auf der nächsten Seite*

```
  INIFile.First;
  Rechenname := INIFile.Name;
  Rechenwert := Val (INIFile.Value, 0);
//*************************************************************

//****************Ausgabe:**********************************
  Screen.Label.Show ( 1, 100, 30, Section1);
  Screen.Label.Show ( 2, 250, 30, 'Wert');
  Screen.Label.Show ( 3, 100, 80, Variable1);
  Screen.Label.Show ( 4, 250, 80, Str (Wert1));
  Screen.Label.Show ( 5, 100, 110, Variable2);
  Screen.Label.Show ( 6, 250, 110, Str (Wert2));
  Screen.Label.Show ( 7, 100, 140, Variable3);
  Screen.Label.Show ( 8, 250, 140, Str (Wert3));
  Screen.Label.Show ( 9, 100, 200, Section2);
  Screen.Label.Show (10, 100, 230, Rechenname);
  Screen.Label.Show (11, 250, 230, Str (Rechenwert));
//*************************************************************

  repeat
  until StepContinue;

  SetValue (1);
end.
```

20. Speichern Sie Ihre Interpreterschritte und starten Sie den Ablauf.

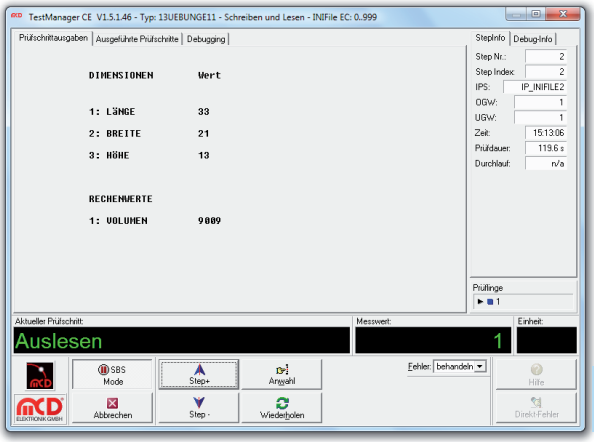


Bild 6-10
Ablauf starten

Inhalt des INI-Files:

```
[DIMENSIONEN]
1: LÄNGE=33
2: BREITE=21
3: HÖHE=13
[RECHENWERTE]
1: VOLUMEN=9009
```

7. Hardware Zugriff

Für den Betrieb des Programms mit Zugriff auf alle Schnittstellen wird eine Programmlizenz benötigt. Um Hardwarezugriff mit der Demoversion zu erlangen, kann eine Kurzzeitlizenz angefordert werden.
Eine Vollversion des TestManagers kann über das Anfordern einer Lizenz und den Erhalt eines Keys erlangt werden.

Lizenz anfordern

Über die ➡ **Menüleiste** ➡ **Intern** ➡ **Lizenz** und das Register ➡ **Lizenz anfordern**, gelangt man zum Fenster, über das Lizenzen angefordert werden können.

Serielle Schnittstelle

Das Modul für die Seriellen Ports dient dem Zugriff auf alle Arten von RS232 Ports, egal durch welche Karten oder Hardware sie im PC realisiert werden. Wird der COM Port in der Systemverwaltung von Windows angezeigt, kann er vom Programm genutzt werden. Das Modul stellt zusätzlich die Ports für weitere Module (z.B. LIN) zur Verfügung und übernimmt für diese die Verwaltung der Portparameter.

Über die **Grundeinstellungen** (➡ **Symbolleiste** ➡ **Setup** ➡ **Serielle Ports**) kann ein serieller Port erstellt und konfiguriert werden.

■ Übung 7-1
Verwendung serieller Schnittstellen

Bitte beachten Sie, dass Sie diese Übung nur durchführen können, wenn ihr Computer über eine erforderliche COM Schnittstelle verfügt. Für die Durchführung der Übung ist ein Stecker notwendig, den Sie sich leicht selbst anfertigen können. Es handelt sich hierbei um eine 9-polige Sub-D Buchse, die nach folgendem Schaltplan gebrückt werden muss:

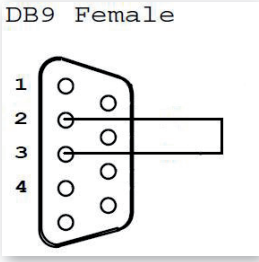


Bild 7-1
Schaltplan 9-polige Sub-D Buchse

Verwendung Serielle Schnittstelle

Diese Übung zielt auf die Verwendung der Seriellen Schnittstelle. Aufgabe ist es eine Kurzzeitlizenz anzufordern, in den Grundeinstellungen einen neuen Seriellen Port anzulegen und zu konfigurieren. Im Ablauf soll ein String über die Serielle Schnittstelle gesendet und empfangen werden. Die Überprüfung des empfangenen Strings sollte ergeben, dass der gesendete String dem empfangenen entspricht. Ist dies nicht der Fall, soll das Ergebnis des Ablaufs entsprechend angepasst werden.

Schritt für Schritt:

1. Stellen Sie sicher, dass Sie als Administrator angemeldet sind.

2. Seriellen Port anlegen:
   Navigieren Sie über die ➡ **Symbolleiste** ➡ **Setup** in die Grundeinstellungen. Wählen Sie auf der linken Seite des Fensters den Punkt ➡ **Serielle Ports**. Legen Sie mit Hilfe des Buttons ➡ **neuen Port** erzeugen einen neuen Port an.

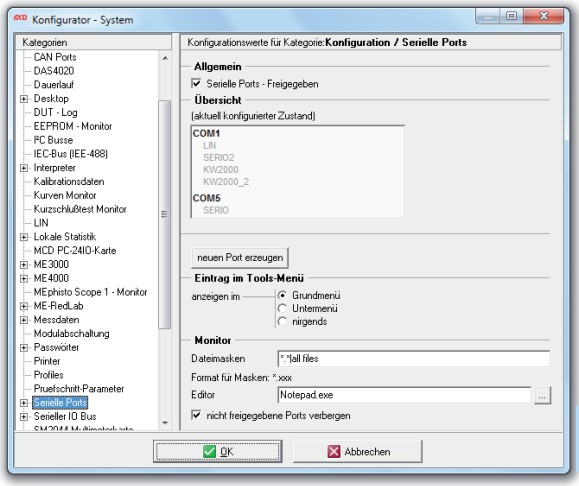


Bild 7-2
Seriel. Port anlegen

3. Geben Sie einen ➡ **Namen** (Beispiel: SERIELL1) für den COM-Port ein und fügen Sie optional einen Kommentar ein. Setzen Sie den Port ➡ **Aktiv** und stellen Sie die ➡ **Baudrate** auf ➡ **9600**. Bestätigen Sie Ihre Änderungen mit dem ➡ **OK** Button.

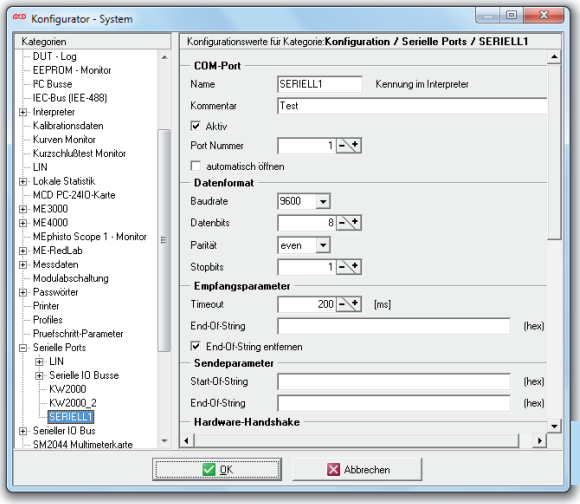


Bild 7-3
Seriel. Port anlegen 2

4. Legen Sie eine neue Typfamilie mit einer Variante an, setzen Sie diese aktiv und speichern Sie die Daten ab.

   Beispiel: Familienname: 14SERIELL
   Typ-Code: Seriell
   Typ-Information: Test

5. Wählen Sie über die Typwahl die angelegte Variante aus.

6. Legen Sie in den Prüfschrittparametern einen neuen Namen für die Step Art fest und definieren Sie Grenzen. Setzen Sie den Step aktiv und speichern Sie ab.

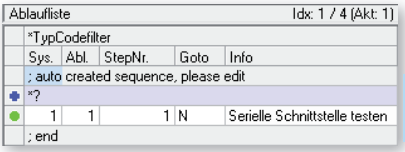


Bild 7-4
Ablaufliste

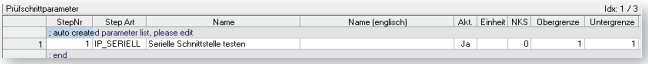


Bild 7-5
Prüfschrittp.

7. Fordern Sie eine Kurzzeitlizenz über die ➡ **Menüleiste** ➡ **Intern** ➡ **Lizenz** an.

8. Starten Sie den Ablauf im SBS-Modus und editieren Sie den Interpreterschritt.

9. Interpreterschritt editieren:
   Legen Sie zwei Variablen vom Typ String an, um die gesendeten und die empfangenen Daten abzuspeichern.

10. Initialisieren Sie, zur Ausgabe von Daten im Prüfschrittausgabefenster, mit Hilfe der Screen-Befehle den Monitor.

11. Weisen Sie der Variablen zum Senden einen String ihrer Wahl zu.

12. Fragen Sie mit Hilfe der Funktion ➡ **RS232.Exists** ('Name der Schnittstelle') ab, ob die angelegte Schnittstelle existiert und geben Sie das Ergebnis auf dem Prüfschrittausgabefenster aus.

    Syntax:

```
If RS232.Exists ('SERIELL1') = 1 Then
begin
  Screen.Label.Show (3, 100, 30, 'Schnittstelle existiert!');
end;
```

13. Öffnen Sie mit dem Befehl ➡ **RS232.Open ()** die angelegte Schnittstelle. Fragen Sie mit ➡ **RS232.IsOpen ()** ab, ob die Schnittstelle geöffnet wurde und geben Sie das Ergebnis ebenfalls aus.

    Syntax:

```
RS232.Open ('SERIELL1');

If RS232.IsOpen ('SERIELL1') = 1 Then
begin
  Screen.Label.Show (4, 100, 60, 'Port ist offen!');
end;
```

14. Senden Sie den in der Variablen abgespeicherten String mit dem Befehl ➡ **RS232.Send ()** über die angelegte Serielle Schnittstelle.

    Syntax:

```
RS232.Send ('SERIELL1', Senden);
```

15. Weisen Sie mit Hilfe der Funktion ➡ **RS232.Read ()**, die über die Schnittstelle empfangenen Daten, der zweiten Variablen zu.

    Syntax:

```
Empfang := RS232.Read ('SERIELL1');
```

16. Geben Sie den gesendeten und den empfangenen String aus und setzen Sie eine Verzögerungszeit von ca. einer Sekunde.

17. Vergleichen Sie, ob der empfangene String dem gesendeten entspricht und passen Sie dementsprechend Ihren Messwert an. Der gesendete String muss aufgrund der Verdrahtung des Steckers dem Empfangenen entsprechen.

Quellcode: *IP_SERIELL.IPS*

```
var
  Empfang     : String;
  Senden      : String;
Step

  Screen.SetTab (1);        //Fokus auf Prüfschrittausgabe
  Screen.ClrScr;
  Screen.Show;

//*****************String zuweisen:****************************
  Senden := 'Hallo';
//*************************************************************

//*********Abfragen ob Schnittstelle konfiguriert ist:**********
  If RS232.Exists ('SERIELL1') = 1 Then
  begin
    Screen.Label.Show (3, 100, 30, 'Schnittstelle ist
  end;                 konfiguriert!');
//*************************************************************

//*****************Schnittstelle öffnen:************************
  RS232.Open ('SERIELL1');

  If RS232.IsOpen ('SERIELL1') = 1 Then
  begin
     Screen.Label.Show (4, 100, 60, 'Port ist offen!');
  end;
//*************************************************************

//*****************Senden:**************************************
  RS232.Send ('SERIELL1', Senden);
//*************************************************************

//*****************Empfangen:***********************************
  Empfang := RS232.Read ('SERIELL1');
//*************************************************************
//*****************Ausgabe:*************************************
  Screen.Label.Show (1, 100,  90, 'Gesendet:  ' +  Senden);
  Screen.Label.Show (2, 100, 120, 'Empfangen: ' + Empfang);
  DateTime.Delay (1000);
//*************************************************************
  repeat
  until StepContinue;
```

➡ *weiter geht´s auf der nächsten Seite*

```
//*****************Messwert setzen:****************************
  If Empfang = Senden Then
  begin
      SetValue (1);
  end
  Else begin
      SetValue (0);
  end;
//*************************************************************

end.
```

18. Speichern Sie Ihren Interpreterschritt ab und starten Sie den Ablauf.

Bild 7-6
Seriel. Schnittstelle testen

> **Hinweis:**
> Sollte es wiederholt zu einem Fail-Ergebnis des Steps kommen, kann dies an einem nichtlizenzierten System liegen. Überprüfen Sie, ob Sie über eine Lizenz verfügen und fordern Sie gegebenenfalls eine Kurzzeitlizenz an.

8. Messdaten

Jeder Prüfschritt erzeugt bei der Ausführung eine Reihe von Daten. Zum einen den echten Messwert und das Ergebnis des Schrittes, aber auch eine Reihe vom Metadaten. So z.B. den Zeitpunkt der Prüfung, die Dauer des Schrittes und Daten, die durch Verweis auf die Art des Prüfschrittes vorhanden sind, z.B. die Nummer des Prüfschrittes, die Einheit etc.
Der Standardweg zur Speicherung dieser Daten erfolgt im DBF Format (dBase). Dies erfordert eine installierte BDE (Borland Database Engine).
Das Abspeichern der Messdaten kann im Standardformat oder in einer frei definierbaren Tabellenstruktur erfolgen. Zur Vermeidung zu großer Datenmengen steht eine eingebaute Datenpflege zur Verfügung.
Der Pfad, unter dem die Messdaten abgelegt werden, ist einstellbar. Bei typbezogener Speicherung wird für jede Typfamilie ein Unterverzeichnis angelegt. Standardmäßig werden die Daten aller Typen gemeinsam gespeichert.
Es besteht zusätzlich die Möglichkeit Messdaten an eine DLL zu übergeben.

■ Übung 8-1
Umgang mit Messdaten

Inhalt dieser Übung ist es, den Umgang mit Messdaten aufzuzeigen. Es soll die Erfassung der Messdaten im DBF-Format und die lokale Speicherung realisiert werden.

Schritt für Schritt:

1. Überprüfen Sie Ihren Passwortlevel, sollten Sie nicht als Administrator beim System angemeldet sein, ändern Sie dies.

2. Aktivieren Sie die Messwerterfassung im DBF-Format indem Sie über die ➡ **Symbolleiste** ➡ **Setup** in die Grundeinstellungen navigieren. Wählen Sie im linken Feld der Grundeinstellungen ➡ **Messdaten** aus und lassen Sie sich mit ➡ + die Unterpunkte anzeigen.

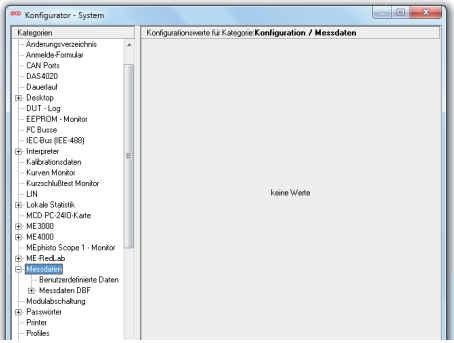


Bild 8-1
Messdaten Konfiguration

3. Wählen Sie den Unterpunkt ➡ **Messdaten DBF** aus und geben Sie im rechten Feld im Unterpunkt ➡ **Allgemein** die Messdaten DBF frei.

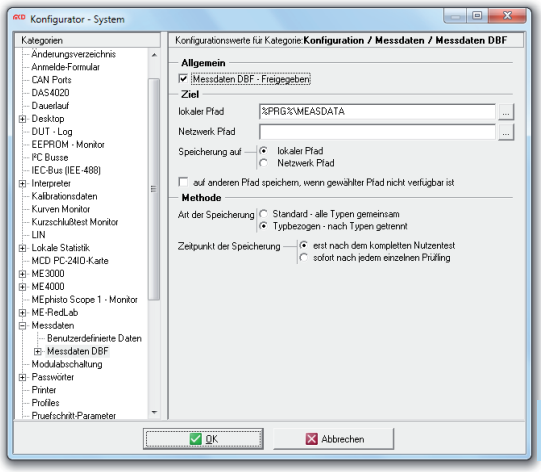


Bild 8-2
Messdaten DFB freigeben

Die Messwerte werden somit in einem DBF-File unter dem Installationsverzeichnis im Ordner MEASDATA hinterlegt. Alternativ kann auch ein Netzwerkpfad angegeben werden. Die Daten werden nach Typen getrennt und erst nach dem kompletten Nutzentest lokal abgespeichert.

4. Öffnen Sie das Menü zur Einstellung der Datenpflege (Unterpunkt von Messdaten DBF im linken Feld des Fensters). Stellen Sie ein, dass die Datenpflege bei Programmstart gestartet wird.

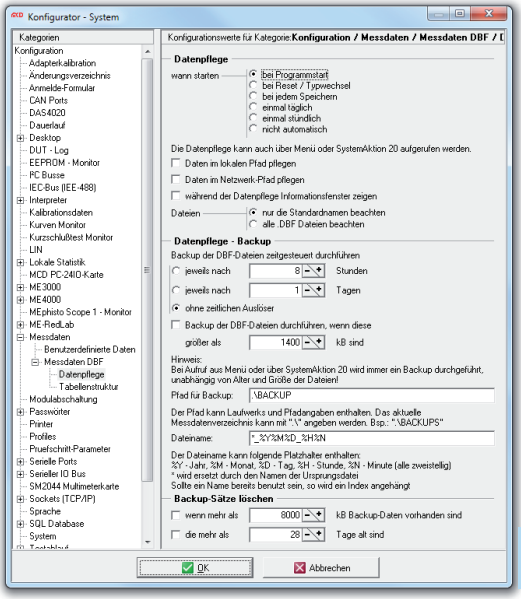


Bild 8-3
Datenpflege einstellen

5. Speichern Sie die Einstellungen ab, indem Sie mit dem Button ➡ **OK** die Änderungen bestätigen.

6. Wählen Sie über die ➡ **Typwahl** den Ablauf der Typfamilie ➡ **2MEASUREDATA** aus und starten Sie den Ablauf im **Automatic-Modus**.

7. Navigieren Sie über den Windows Explorer über das ➡ **Installationsverzeichnis** des TestManagers in den Ordner ➡ **MEASDATA**. Dort sollte sich nun ein Ordner ➡ **TYPE_2MEASUREDATA** befinden, der die DBF-Files ➡ **FAILDATA**, ➡ **MEASDATA** und ➡ **PASSFAIL** enthalten.

8. Sie können sich den Inhalt der DBF-Files mit Hilfe von Microsoft Excel anzeigen lassen.

9. Reports

Messwerte und Ergebnisse können eingesehen, abgespeichert und ausgedruckt werden. Um die Ausdrucke an (fast) beliebige Benutzerwünsche anpassen zu können, wurde der TestManager mit einem leistungsfähigen Reportgenerator ausgestattet. Mit dessen Hilfe kann der Ausdruck frei gestaltet werden. Es kann auf Vorlagen zurückgegriffen und diese verändert werden. Messwerte lassen sich über die ➡ **Menüleiste** ➡ **Messwerte** ➡ **Anzeigen** abrufen, Reports verwalten und ausdrucken.
Durch die Verwendung von Reports lassen sich allgemeine 'Schablonen' anlegen, so dass nicht für jeden Ablauf ein eigener Report erstellt werden muss.

■ Übung 9-1
Erstellen eines eigenen Reports

Das Ziel dieser Übung ist es einen eigenen Report zu erstellen, in dem die Messwerte des bereits vorhandenen 2MEASUREDATA Ablaufes abgelegt und ausgedruckt werden können. Hierbei soll das Design individuell angepasst werden.

Schritt für Schritt:

1. Überprüfen Sie Ihren Passwortlevel, sollten Sie nicht als Administrator beim System angemeldet sein, ändern Sie dies.

2. Fordern Sie eine Kurzzeitlizenz an, sollte Ihre Software nicht lizenziert sein.

3. Wählen Sie über die ➡ **Typwahl** den Ablauf der Typfamilie ➡ **2MEASUREDATA** aus und starten Sie den Ablauf im Automatik-Modus.

4. Nach Beendigung des Ablaufs navigieren Sie über die ➡ **Menüleiste** ➡ **Messwerte** ➡ **Anzeigen** zur Anzeige der aktuellen Messwerte des letzen Prüflaufes.

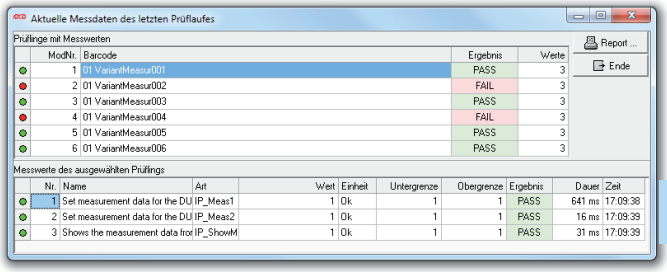


Bild 9-1 Messwerte

5. Aktivieren Sie den Button ➡ **Report** um das Verwaltungsfenster zu öffnen.

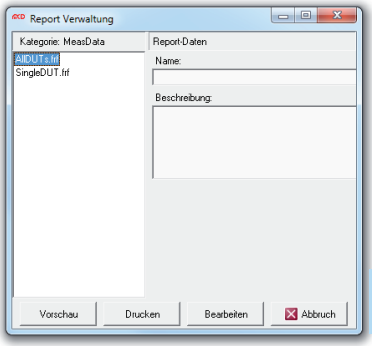


Bild 9-2
Report Verwaltung

6. Wählen Sie den bereits vorhandenen Report AllDUTs.frf aus und bestätigen Sie Ihre Wahl mit dem Button ➡ **Bearbeiten**, um den Report an die eigenen Wünsche anzupassen.

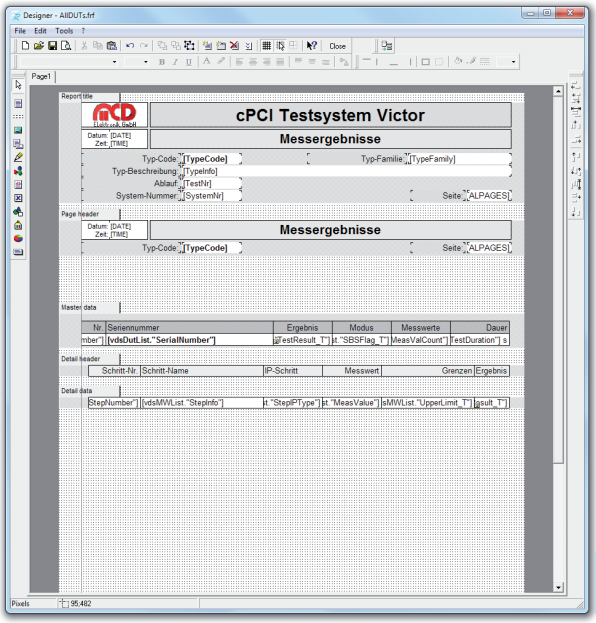


Bild 9-3
Report Designer

7. Um einen leeren Report zu bearbeiten, wählen Sie über die ➡ **Symbolleiste** ➡ **New report** aus.

8. Legen Sie nun ihren eigenen Report title an, dies könne Sie über die ➡ **Symbolleiste** ➡ **Insert band** tun.

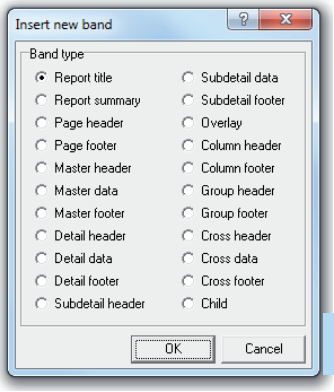


Bild 9-4
Report Titel anlegen

9. Wählen Sie im Insert new band Fenster den ➡ **Report title** aus und bestätigen Sie mit dem Button ➡ **OK**.

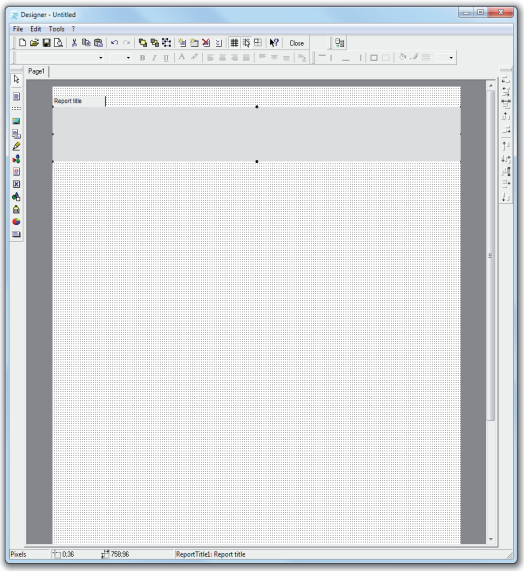


Bild 9-5
Report title auswählen

10. Wählen Sie einen Platz im Report aus, an dem Sie ihren Report title einfügen möchten.

11. Über die ➡ **Symbolleiste** ➡ **Insert rectangle object** lassen sich rechteckige Felder auf dem Report einfügen. Platzieren Sie ein solches Feld in ihrem Report title.

12. Es wird ein Text Editor Fenster geöffnet, das es Ihnen ermöglicht Text in das Feld einzugeben oder sich Variablen vom System übergeben zu lassen.

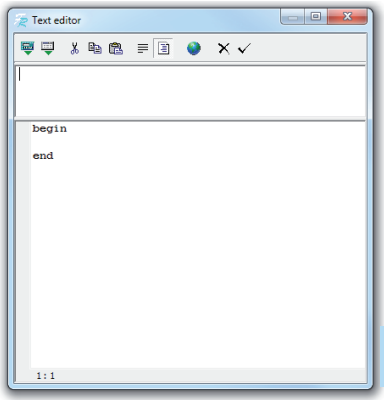


Bild 9-6
Text editor

13. Geben Sie im oberen Feld einen geeigneten Namen für Ihren Report ein und bestätigen Sie Ihre Eingabe. Ziehen Sie ihn an die gewünschte Stelle und stellen Sie die gewünschte Schriftgröße ein.

14. Sie können je nach Belieben über die ➡ **Symbolleiste** eine Füllfarbe, einen Rahmen oder ähnliches für das Feld einstellen.

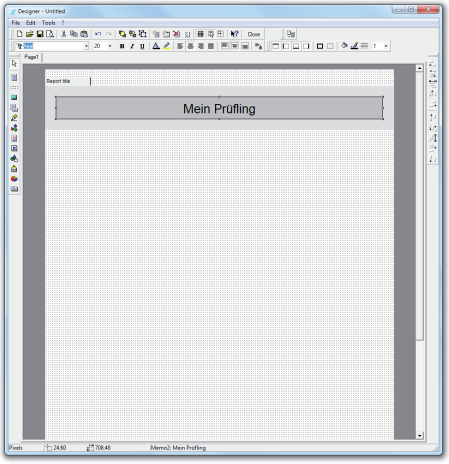


Bild 9-7
Beispiel Designer

15. Um Variablen wie zum Beispiel das aktuelle Datum abzurufen und auf dem Report auszugeben, platzieren Sie ein weiteres Textfeld. Wählen Sie in der ➡ **Symbolleiste** des Text Editor Fensters ➡ **Insert data field**, im linken Feld

➡ **vdsDutList** und im rechten ➡ **TestTime_T**.

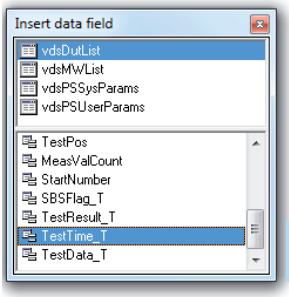


Bild 9-8
Beispiel Designer

Mit dieser Vorgehensweise lassen sich beliebige Variablen des Systems abrufen und müssen nicht für jeden Ablauf von Hand editiert werden.

16. Fügen Sie Ihrem Report title auf diese Weise weitere Variablen wie Typfamilie, Typ Code, Testzeit usw. hinzu.

    Es können in einem Feld auch mehrere Variablen ausgegeben werden, dies ist beispielsweise bei den Seitenzahlen von Vorteil:

    [PAGE#] von [TOTALPAGES]

    Angezeigt wird somit die aktuelle Seitenzahl von der Gesamtanzahl der Seiten.

17. Fügen Sie ein neues Band ein und wählen Sie für die Art ➡ **Page header** aus. Mit diesem Befehl fügen Sie eine Kopfzeile (Page header) in Ihrem Report hinzu. Über einen Rechtsklick auf die Kopfzeile kann eingestellt werden, ob diese auch auf der ersten Seite verwendet werden soll (On first page). Dies ist in diesem Fall nicht notwendig. Erstellen Sie eine Kopfzeile nach ihren Vorstellungen.

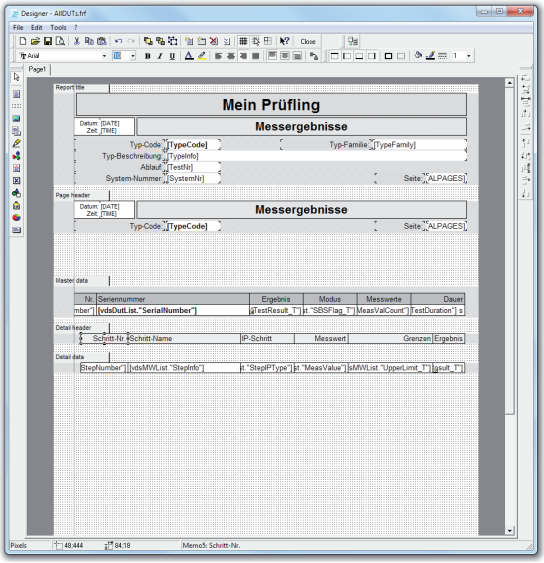


Bild 9-9
Beispiel

18. Erstellen Sie ein weiteres Band vom Typ ➡ **Master data**. Stellen Sie für dieses Band die Quelle für die zu beziehenden Daten ein. Führen Sie dazu einen Doppelklick auf dem Band aus:

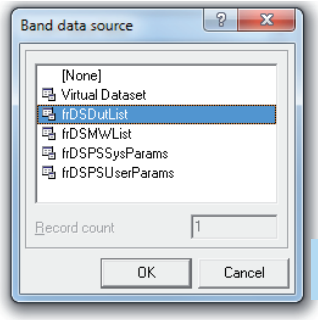


Bild 9-10
Neues Band

19. Wählen Sie ➡ **frDSDutList** aus und bestätigen Sie mit ➡ **OK**.

20. Realisieren Sie anhand einer Tabelle die Ausgabe der Informationen Modulnummer, Seriennummer, Ergebnis, Modus, Messwerte und Dauer.
Die zu verwendeten Daten können Sie der Sparte ➡ **vdsDutList** entnehmen.

Modulnummer	➡ **[vdsDutList.“ModuleNumber“]**
Seriennummer	➡ **[vdsDutList.“SerialNumber“]**
Ergebnis	➡ **[vdsDutList.“TestResult_T“]**
Modus	➡ **[vdsDutList.“SBSFlag_T“]**
Messwerte	➡ **[vdsDutList.“MeasValCount“]**
Dauer	➡ **[vdsDutList.“TestDuration“] s**

➡ *Beispiel Master data siehe Bild 9-9 Beispiel*

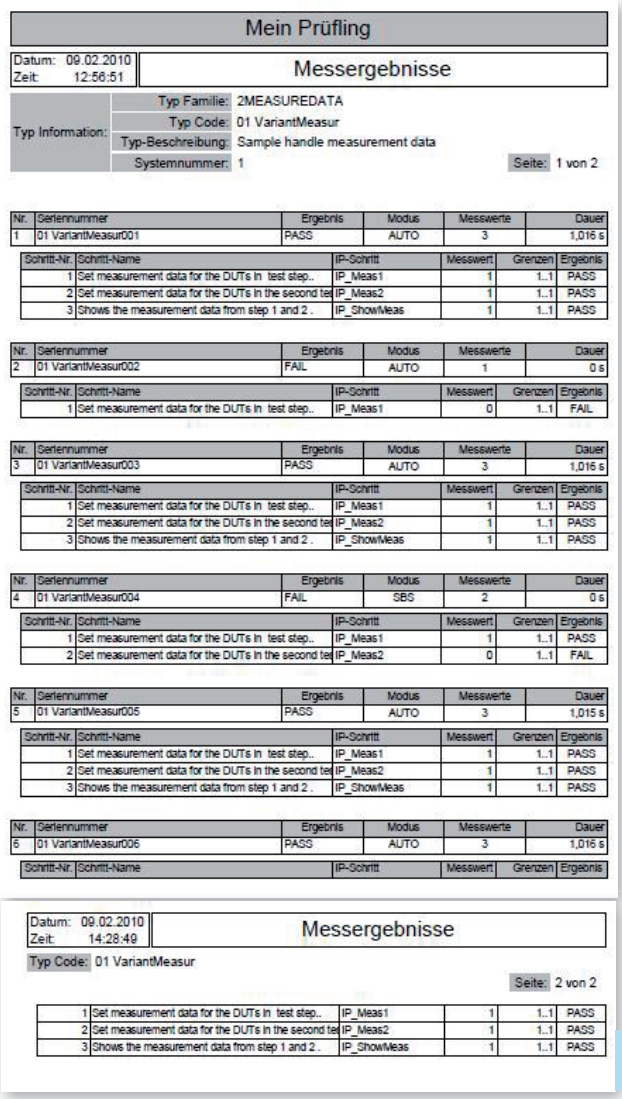
21. Erstellen Sie ein neues Band vom Typ ➡ **Detail header** und eines vom Typ ➡ **Detail data**. Diese beiden Bänder sind eng miteinander verbunden und dienen der Ausgabe der Messwerte der einzelnen Interpreterschritte. Im Detail header werden die Namen und im Detail data die Variablen (Werte) hinterlegt.

    Detail header:
    ➡ **Schritt-Nr.**
    ➡ **Schritt-Name**
    ➡ **IP-Schritt**
    ➡ **Messwert**
    ➡ **Grenzen**
    ➡ **Ergebnis**

    Der Detail data Abschnitt soll auf die ➡ **frDSMWList** zurückgreifen und die Variablen (Werte) zum Detail header ausgeben.

    ➡ *Beispiel Detail header + data siehe Bild 9-9 Beispiel*

22. Mit Hilfe der grauen Zonen der Bänder können Sie die Abstände variieren. Lassen Sie sich Ihren Report über die ➡ **Symbolleiste** ➡ **Preview report** anzeigen und passen Sie das Layout nach Ihren Vorstellungen an und überprüfen Sie ob die Variablen korrekt angezeigt werden.



Mein Prüfling

Datum: 09.02.2010
Zeit: 12:56:51

Messergebnisse

Typ Information:	
Typ Familie:	2MEASUREDATA
Typ Code:	01 VariantMeasur
Typ-Beschreibung:	Sample handle measurement data
Systemnummer:	1

Seite: 1 von 2

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
1	01 VariantMeasur001	PASS	AUTO	3	1,016 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis
1	Set measurement data for the DUTs in test step..	IP_Meas1	1	1..1	PASS
2	Set measurement data for the DUTs in the second tes	IP_Meas2	1	1..1	PASS
3	Shows the measurement data from step 1 and 2 .	IP_ShowMeas	1	1..1	PASS

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
2	01 VariantMeasur002	FAIL	AUTO	1	0 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis
1	Set measurement data for the DUTs in test step..	IP_Meas1	0	1..1	FAIL

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
3	01 VariantMeasur003	PASS	AUTO	3	1,016 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis
1	Set measurement data for the DUTs in test step..	IP_Meas1	1	1..1	PASS
2	Set measurement data for the DUTs in the second tes	IP_Meas2	1	1..1	PASS
3	Shows the measurement data from step 1 and 2 .	IP_ShowMeas	1	1..1	PASS

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
4	01 VariantMeasur004	FAIL	SBS	2	0 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis
1	Set measurement data for the DUTs in test step..	IP_Meas1	1	1..1	PASS
2	Set measurement data for the DUTs in the second tes	IP_Meas2	0	1..1	FAIL

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
5	01 VariantMeasur005	PASS	AUTO	3	1,015 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis
1	Set measurement data for the DUTs in test step..	IP_Meas1	1	1..1	PASS
2	Set measurement data for the DUTs in the second tes	IP_Meas2	1	1..1	PASS
3	Shows the measurement data from step 1 and 2 .	IP_ShowMeas	1	1..1	PASS

Nr.	Seriennummer	Ergebnis	Modus	Messwerte	Dauer
6	01 VariantMeasur006	PASS	AUTO	3	1,016 s

Schritt-Nr.	Schritt-Name	IP-Schritt	Messwert	Grenzen	Ergebnis

Datum: 09.02.2010
Zeit: 14:28:49

Messergebnisse

Typ Code: 01 VariantMeasur

Seite: 2 von 2

1	Set measurement data for the DUTs in test step..	IP_Meas1	1	1..1	PASS
2	Set measurement data for the DUTs in the second tes	IP_Meas2	1	1..1	PASS
3	Shows the measurement data from step 1 and 2 .	IP_ShowMeas	1	1..1	PASS

Bild 9-11 Messergebnisse

23. Speichern Sie Ihren Report über die ➡ **Menüleiste** ➡ **File** ➡ **Save as** unter einem neuen Namen ab.

10. DLL

Die Abkürzung DLL steht für **Dynamic Link Library**, was soviel bedeutet wie dynamisch vernetzte Bibliothek. Eine DLL kann Programmcode, Daten und Ressourcen in beliebigen Kombinationen enthalten. Verwendet wird hierfür meist das Portable-Executable-Dateiformat.
Eine DLL dient im eigentlichen Sinne der Reduzierung des benötigten Speicherplatzes auf der Festplatte oder dem Hauptspeicher. Aber auch andere Anwendungsgebiete sind möglich. Ein Beispiel für die Verwendung einer DLL:

Programmcode, der von mehreren Anwendungen genutzt wird, kann in einer DLL zusammengefasst und auf der Festplatte hinterlegt werden. Der Vorteil liegt darin, dass der Programmcode nur einmal in den Speicher gelesen werden muss und nicht von jeder Anwendung, die auf ihn zugreifen möchte.

■ Übung 10-1
Umgang mit DLLs

Diese Übung soll Ihnen den Umgang mit DLLs näher bringen. Verwendet werden soll die Kernel32.dll, die Sie in Ihren ➡ **Windowsverzeichnis** im Unterverzeichnis ➡ **System32** finden. Diese DLL enthält eine Funktion zum Erzeugen eines Beep über den internen Lautsprecher ihres Computers. Die Kernel32.dll soll in ihren Ablauf eingebunden und die Beep-Funktion dem System bekannt gemacht werden. Realisieren Sie mit Hilfe dieser Funktion ein beliebiges Klangmuster.

Schritt für Schritt:

1. Stellen Sie sicher, dass Sie beim System als Administrator angemeldet sind. Sollte dies nicht der Fall sein, melden Sie sich bitte an.

2. Erstellen Sie eine neue Typfamilie, die eine Variante enthält und setzen Sie diese aktiv.

   Beispiel: Typfamilienname: 15DLL
   Typ-Code: 01DLL
   Typ-Information: Kernel32

3. Editieren Sie die Ablaufliste und die Prüfschrittparameter. Benötigt wird lediglich ein Prüfschritt. Geben Sie einen neuen Namen für die Step Art ein, um einen neuen Interpreterschritt zu erzeugen.

Ablaufliste				Idx: 1 / 4 (Akt: 1)
*TypCodefilter				
Sys.	Abl.	StepNr.	Goto	Info
; auto created sequence, please edit				
*?				
1	1	1	N	DLL
; end				

Bild 10-1
Ablaufliste

Prüfschrittparameter

	StepNr	Step Art	Name	Name (englisch)	Akt.	Einheit	NKS	Obergrenze	Untergrenze
; auto created parameter list, please edit									
1	1	IP_DLL2	DLL	DLL	Ja		0	0	0
; end									

Bild 10-2
Prüfschrittpara.

4. Wählen Sie über die ➡ **Typwahl** die in Schritt 3. angelegte Variante und starten Sie den Ablauf.

5. Editieren des Interpreterschritts:
Die Beep-Funktion, die Sie aus der **Kernel32.dll** verwenden möchten, liefert bei der Ausführung einen Real Wert zurück. Um diesen zuweisen zu können sollten Sie zunächst eine Variable vom Typ Real anlegen.

6. Soll eine DLL in einen Interpreterschritt eingebunden werden, muss die DLL dem System bekannt gemacht und einen so genannten Alias zugewiesen bekommen. Über diesen Alias kann die DLL im Interpreter aufgerufen werden.
Mit der Prozedur ➡ **DLL.OPEN** (Alias, Pfad unter der die DLL abgespeichert ist) wird die DLL geöffnet und mit dem internen Alias verwaltet.

Syntax:

```
DLL.Open ('Kernel32', 'C:\Windows\System32\Kernel32.dll');
```

Die DLL kann nun über den Namen Kernel32 angesprochen bzw. aufgerufen werden.

7. Mit der Prozedur ➡ **DLL.REGISTER()** lassen sich Funktionen und Prozeduren in einer DLL suchen und dem Ablaufschritt wiederum über einen Alias mit einem Namen bekannt machen.

Syntax:

```
DLL.REGISTER ('Kernel32', 'Beep', 'S:I,I:L');
```

Die in der DLL Kernel32 enthaltene Funktion Beep wird dem System bekannt gemacht. Um Funktionen oder Prozeduren aus einer DLL aufrufen zu können, müssen in den meisten Fällen Variablen übergeben werden. Diese müssen ebenfalls beim Registrieren des Beep-Befehls angegeben werden, dies erfolgt abhängig von den zu übergebenden Datentypen. In diesem Fall mit der Angabe

von ➡ **'S:I,I:L'**. Durch diese Angabe wird das Aufrufformat der Funktion/Prozedur festgelegt:

Der erste Buchstabe legt den Aufruftyp fest, hierbei werden drei Typen unterschieden. ➡ **P** für einen Pascal-Aufruf, ➡ **C** für einen C-Aufruf und ➡ **S** für einen Standard-Aufruf. Der Aufruftyp muss mit dem Aufruftyp der Routine in der DLL übereinstimmen.

Der Definition des Aufruftyps folgt ein Doppelpunkt (:), dem der Typ des ersten zu übergebenden Parameter folgt. Die Codes für die einzelnen Parameter werden durch Kommas getrennt und können einer Liste in der Interpreterhilfe entnommen werden. Der Rückgabewert wird mit einem weiteren Doppelpunkt (:) von den Parametern getrennt, dem ebenfalls die Angabe des Typs folgt.

Somit ergibt sich für die Syntax:

```
DLL.REGISTER ('Kernel32', 'Beep', 'S:I,I:L');
```

folgende Erläuterung:

Dem System wird die Funktion Beep aus der DLL Kernel32 bekannt gemacht. Der Aufruf der Funktion erfolgt über einen Standard-Aufruf (Standardcall). Übergeben werden zwei Parameter vom Typ Integer, die Beep-Funktion liefert einen Rückgabewert vom Typ Boolean (dies entspricht einem Logikwert). Übergeben werden der Funktion die Integer Frequenz (Tonhöhe) in Hz und Tondauer in ms.

8. Dem System sind nun die DLL unter dem Namen Kernel32 und die Funktion Beep unter Ton bekannt. Der Beep-Befehl kann nun aufgerufen und der Rückgabewert der Variablen zugewiesen werden. Der Aufruf eines Befehls (Funktion oder Prozedur) erfolgt über die Funktion ➡ **DLL.Call ()**.
Über diese Funktion lässt sich die zuvor registrierte Funktion aus der ebenfalls zuvor geöffneten DLL aufrufen. In der Funktion werden die Parameter, die in der Registrierung definiert wurden, übergeben.

Syntax:

```
Wert := DLL.Call ('Kernel32', 'Beep', 3500, 50);
```

Aus der DLL Kernel32 wird die Funktion Beep aufgerufen, die mit der Frequenz 3500Hz und einer Dauer von 50ms ausgeführt werden soll. Der Rückgabewert wird der Variablen Wert zugewiesen.

Mit wiederholtem Aufruf des Beep-Befehls, mit veränderten Frequenzen und Dauern können Sie beliebige Klangmuster erzeugen.

Syntax:

```
Wert := DLL.Call ('Kernel32', 'Beep', 3500, 50);
Wert := DLL.Call ('Kernel32', 'Beep', 4500, 50);
Wert := DLL.Call ('Kernel32', 'Beep', 5500, 50);
…
```

9. Am Ende Ihres Interpreterschrittes muss die DLL mit Hilfe der Prozedur ➡ **DLL.CLOSE()** geschlossen werden. Bei der Ausführung des Befehls werden alle Registrierungen gelöscht. Dies bedeutet, dass für einen erneuten Aufruf nach dem DLL.CLOSE-Befehl das erneute Öffnen und Registrieren der DLL und des Befehls erfolgen muss.

Syntax:

```
DLL.CLOSE ('Kernel32');
```

Die DLL mit dem internen Alias Kernel32 wird geschlossen und die Registrierung des Beep-Befehls wird gelöscht.

Quellcode: *IP_DLL.IPS*

```
var
  Wert : Real;

step
//**************Öffnen der DLL:*******************************
  DLL.Open ('Kernel32', 'C:\Windows\System32\Kernel32.dll');
//************************************************************

//**************Registrieren des Beep-Befehls:*****************
  DLL.REGISTER ('Kernel32', 'Beep', 'S:I,I:L');
//************************************************************

//**************Aufrufen des Beep-Befehls:**********************
  Wert := DLL.Call ('Kernel32', 'Beep', 3500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 4500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 5500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 6500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 5500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 4500, 50);
  Wert := DLL.Call ('Kernel32', 'Beep', 3500, 50);
//************************************************************

//*********Schließen der DLL und Setzen des Messerts:**********
  DLL.CLOSE ('Kernel32');
  SetValue (0);
//************************************************************

end.
```

10. Speichern Sie Ihren Interpreterschritt ab und starten Sie den Ablauf. Es sollten nun die von Ihnen erzeugte Tonfolge über den internen Lautsprecher ausgegeben werden.
    Bitte beachten Sie, dass Töne ab einer bestimmten Frequenz nicht mehr vom Lautsprecher übertragen werden. Der Frequenzbereich des menschlichen Gehörs liegt bei ca. 20 bis 20000Hz.

11. TestManager ME Messkarten Unterstützung

Der TestManager CE enthält die perfekte Integration des ME-iDS Treiber Konzeptes der Fa. Meilhaus Electronic. Mit dieser Software wird dem Anwender der einfache und schnelle Einstieg in das Meilhaus ME-iDS Treibersystem ermöglicht.

Um auch diesen Teil des Buches effektiv nutzen zu können, laden Sie sich bitte den MCD TestManager ME unter **www.mcd-elektronik.de/deutsch/meil.html** herunter. Auf dieser Seite steht die kostenlose Trialversion zur Verfügung.

Installieren Sie den TestManager ME auf Ihrem PC. Folgen Sie hierzu der Beschreibung zur **Installation** des TestManagers CE am Anfang des Buches *(siehe Kapitel 1. Einführung).*

Starten Sie das System, wie von der CE Version gewöhnt, mit einem Doppelklick auf die Verknüpfung auf ihrem Desktop.

Die Entwicklungsumgebung TestManager ME

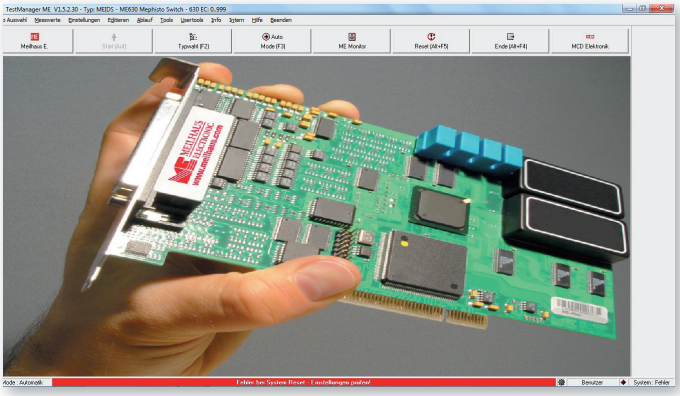


Bild 11-1 Programm-fenster

Das Programm-Fenster des TestManager ME ähnelt dem Aufbau des TestManager CE-Fensters. Wesentliche Unterschiede liegen im Aufbau der Menü- und der Symbolleiste, hier sind neue Menüpunkte beziehungsweise Buttons angelegt. In der Menüleiste ist der Punkt Usertools neu verankert, während die Symbolleiste mit einem Meilhaus E.-, einem ME Monitor- und einem MCD Elektronik-Button ausgestattet wurde. Bedienung und Menüführung sind in beiden Versionen identisch, daher wird an dieser Stelle nicht näher auf diese eingegangen *(siehe Anhang - Unterschiede im Menü des TestManager ME*).

Nähere Informationen zu den Produkten der Firma Meilhaus Electronic erhalten Sie über den Button ➡ **Meilhaus E.**, der Sie im Falle einer bestehenden Internetverbindung zur Homepage leitet. Hier erhalten Sie Informationen über Produkte, Treiber und Software, Verweise auf Fachbücher und Literatur usw.

Der Button ➡ **MCD Elektronik** leitet Sie auf die MCD Homepage, über die Sie bereits die Trialversion heruntergeladen haben.

Der ME Tool Monitor ist über den Button ➡ **ME Monitor** auf der Symbolleiste zu erreichen.

Die spezielle TestManager ME-Version wurde nur zur einfacheren Unterweisung erstellt und wird nicht weiterentwickelt. Die komplette Unterstützung der MEilhaus Karten ist im TestManager CE enthalten und kann direkt benutzt werden.

Anhang

System-Voraussetzungen

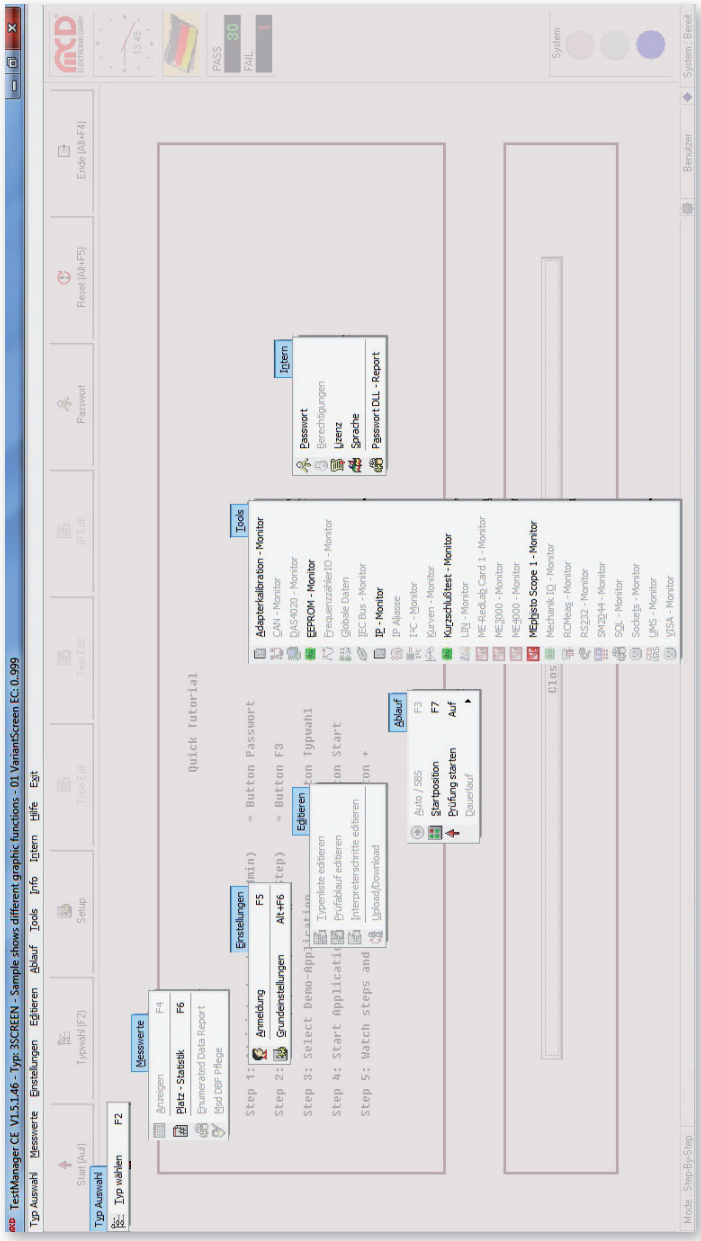
- Windows 7®, Vista®, XP®, 2000® oder NT®
- Programmverzeichnis auf lokalem Laufwerk
- Schreibberechtigung im Programmverzeichnis
- Ca. 25 MB Speicherplatz auf der Festplatte
- 32 MB RAM
- VGA Bildschirm (640x480 Pixel)
- Pentium® PC oder kompatibel
- Installierte Hardware je nach Ausstattung
- Optional: installiertes Netzwerk
- Optional: installierte Borland Database Engine (BDE)

Diese Werte sind Minimalangaben, der Rechner darf selbstverständlich besser ausgestattet sein. Das Programm selbst legt keinerlei Dateien außerhalb seines Programmverzeichnisses an. Eine Ausnahme sind nur Dateien in Pfaden, die ganz speziell angegeben werden (z.B. zur Messwertspeicherung auf dem Netz).
Das Programm kann auch auf normalen (Büro-) PCs gestartet werden, wenn der Zugriff auf die nicht vorhandene Hardware in der Grundeinstellung des Programms abgeschaltet wird.
Das Programm unterstützt Windows XP Stile, wenn diese im Betriebssystem aktiviert sind und auf der Desktop-Seite der Grundeinstellung die Unterstützung aktiviert wurde.

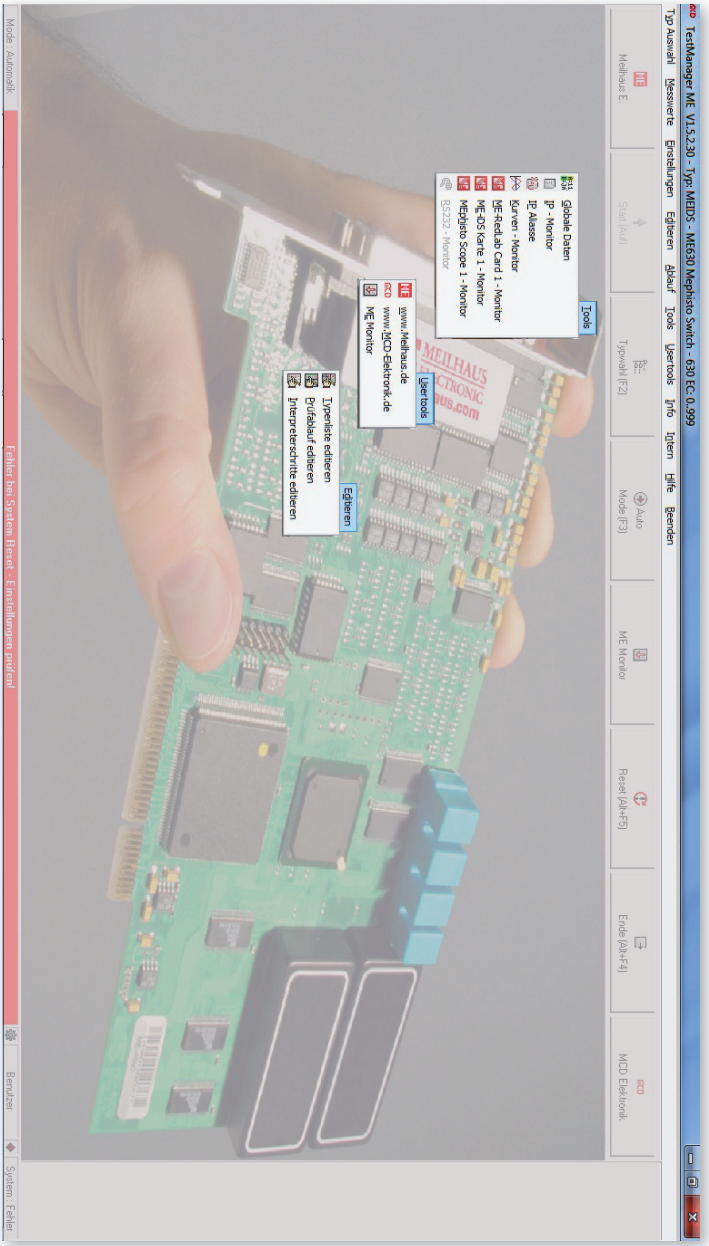
Interessante Web-Links für TestManager-User

www.mcd-elektronik.de	Webseite des Herstellers MCD Elektronik.
www.meilhaus.com	Distributor der zugehörigen Messkarten und Messtechnik-Produkten.
www.mcd-elektronik.de/tmce.html	Informationen zum TestManager CE und Download der Demo-Version, sowie des Tutorials.
www.mcd-elektronik.de/deutsch/meil.html	Informationen zum TestManager ME und Download der Trialversion ME und Toolmonitore MEphisto Scope und ME-iDS
www.mcd-elektronik.de/downloads.html	Die Download-Seite der MCD Elektronik. Hier finden Sie unter anderem die Übungsbeispiele aus diesem Buch zum Download.

Die wichtigsten TestManager CE Menüs



Unterschiede im Menü des TestManager ME



Index

A

Ablaufliste 22
Ablaufnummer 23
Ablaufschritte 22

B

Bibliotheken 26

C

ComObject 27

D

Datentypen Real 27
Datentypen wandeln 37
Debugger 28
DLL 85

E

Entwicklungsumgebung 11
Erstellen eines eigenen Reports 77

F

Felder der Prüfschrittparameter 23
Files 49
Funktionen 26

G

Globale Variable 37
Glücksspiel 38

H

Hardware Zugriff 67

I

INI-File 60
Installation 9
Interpreter 26
Interpretereditor 27
Interpreterhilfe 28
Interpreterschritt editieren 28

L

Lizenz anfordern 67

M

ME-iDS Treibersystem 91
Meilhaus Electronic 91
Messdaten 73

N

NetObject 27

P

Passwortstufen 12
Prozedur 26
Prüfablauf 21
Prüfablauf erstellen 24
Prüfschritte 23

R

Reports 77

S

SBS und Automatik Modus 13
Screen-Objekte 33
Serielle Schnittstelle 67
Sprungarten 23
String 27
StringVector 27

Systemnummer 22

T

Testfenster 14
TestManager ME 91
Tools 45
Typfamilie 17
Typvariante 19

U

Umgang mit DLLs 85
Umgang mit Messdaten 73

V

Vector 27
Verwalten von Varianten 19
Verwendung serieller Schnittstellen 67
Verwendung von Tools 45

TestManager CE

...ist die tausendfach bewährte Software-Entwicklungs-Umgebung für Prüfabläufe und Test. Die Umgebung ermöglicht das Anlegen und Verwalten von Typfamilien und deren Varianten, für die Prüfabläufe erstellt werden können. Der TestManager bietet ein großes Spektrum an Möglichkeiten, wie zum Beispiel die Anbindung an unterschiedliche Hardwarekomponenten. Es werden gängige Bussysteme, wie z.B. LIN, TCP/IP, CAN und IIC-Bus unterstützt. Dem Anwender werden eine Vielzahl von Tools und Einstellungsmöglichkeiten geboten.

Das vorliegende Buch soll kein Software-Handbuch sein. Vielmehr werden dem Einsteiger wichtige Programmier-Techniken kurz erklärt und anhand von ausführlich beschriebenen Übungen vertieft.

Aus dem Inhalt:

- Vorhandene Prüfabläufe starten
- Prüfabläufe erstellen
- Interpreterschritte editieren
- Screen-Objekte anwenden
- Verwendung von Tools
- Erstellen, Schreiben und Lesen von Files/INI-Files
- Verwendung serieller Schnittstellen
- Umgang mit Messdaten
- Anlegen neuer Typfamilien
- Verwalten von Varianten

MCD Elektronik - Messtechnologie bis ins Detail.

MCD Elektronik ist eines der führenden Unternehmen für die Entwicklung und Herstellung kundenspezifischer Messgeräte und Prüfsysteme von der Konzeption bis zu Inbetriebnahme am Einsatzort. Ihre innovativen Lösungen kommen in nahezu allen Bereichen der Qualitätssicherung zum Einsatz, in denen Praxistauglichkeit und Rentabilität entscheidend sind.

1. Auflage, für TestManager CE ab Version 1.5.1.46
MCD TestManager Lehrbuch - Bestell.-Nr. 118006