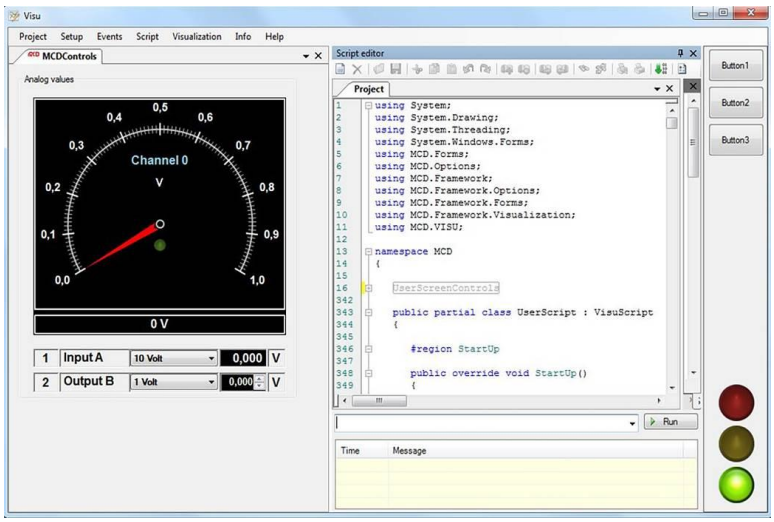
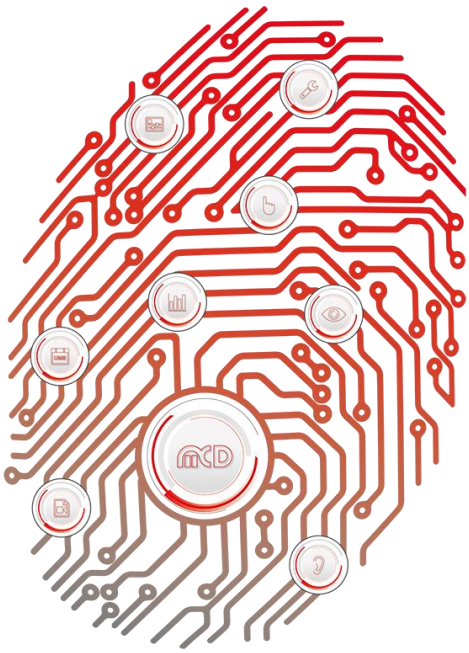
Bedienungsanleitung

Toolmonitor Visu





GET IN touch WITH SENSITIVE TESTING

Softline

Modline

Conline

Boardline

Avidline

Pixline

Applikation

MCD Elektronik GmbH
Hoheneichstr. 52
75217 Birkenfeld
Telefon 0 72 31/78 405-0
Telefax 0 72 31/78 405-10
info@mcd-elektronik.de
www.mcd-elektronik.de
Sitz: Birkenfeld
Geschäftsführer: Bruno Hörter
Registergericht Mannheim
HRB 505692

Vorlagenversion: 5.4 / 2018-04-09
V1.4 2018-09-03 (TS)

Inhalt

1. ALLGEMEIN 4

2. INSTALLATION VON SOFTWARE UND TREIBERN 4

2.1. VORAUSSETZUNGEN 4

2.2. STARTEN DER INSTALLIERTEN SOFTWARE 4

3. EINFÜHRUNG IN DIE BEDIENUNG 5

3.1. KURZEINFÜHRUNG 5

4. HANDBUCH SOFTWARE 7

4.1. DESIGNER 7

4.1.1. Designer öffnen 7

4.1.2. Neues Formular anlegen 7

4.1.3. Einstellungen und Parameter 8

4.1.4. Menüstruktur anpassen 9

4.1.5. Properties und Events 10

4.1.6. Controls 11

4.2. SCRIPT 13

4.2.1. Script Editor 13

4.2.1.1. Base Script 14

4.2.1.2. Explorer 15

4.2.1.3. Kontextmenü 16

4.2.1.4. Statusanzeige 18

4.2.1.5. Texteditor 19

4.2.2. Run - Befehl 23

4.2.3. Stop - Befehl 23

4.2.4. DLLs einbinden 24

4.2.5. Global Variables 25

4.2.6. Virtual Interface 25

4.3. REPORT 29

5. EVENTS 34

5.1. LOGGING 34

5.2. TRACE 34

6. DEBUGGER 35

6.1. INSTALLATION DES DEBUGGERS 35
6.2. STARTEN DES DEBUGGERS 36
6.3. SCRIPT DEBUGGEN 39
6.4. BREAKPOINTS NUTZEN 39
6.5. VARIABLEN ÜBERWACHEN 40

1. Allgemein

Mit dem „Visualisierungs - Toolmonitor“, oder auch „Toolmonitor Visu“, können einfach und schnell Bedien - und / oder Benutzeroberflächen in Verbindung mit dem MCD TestManager erstellt werden. Hierzu wird keine externe Software benötigt. Die über den *Designer* erstellten Oberflächen, können über den Skript Editor mit eigenen Prozeduren und Funktionen verknüpft werden.

Bestellnummer: # 122427

2. Installation von Software und Treibern

2.1. Voraussetzungen

- Windows (Windows XP® – Windows 8.1®, 32 oder 64 Bit)
- .Net – Framework 3.0

Zur Installation des Toolmonitors Visu ist es ausreichend, die *VisuMonitor.exe* in ein beliebiges Verzeichnis auf dem Zielsystem zu kopieren.

2.2. Starten der installierten Software

Die Software kann durch Ausführen der *VisuMonitor.exe* gestartet werden.

3. Einführung in die Bedienung

3.1. Kurzeinführung

Wird der Toolmonitor Visu das erste Mal gestartet, erscheint die noch leere Oberfläche des Monitors.

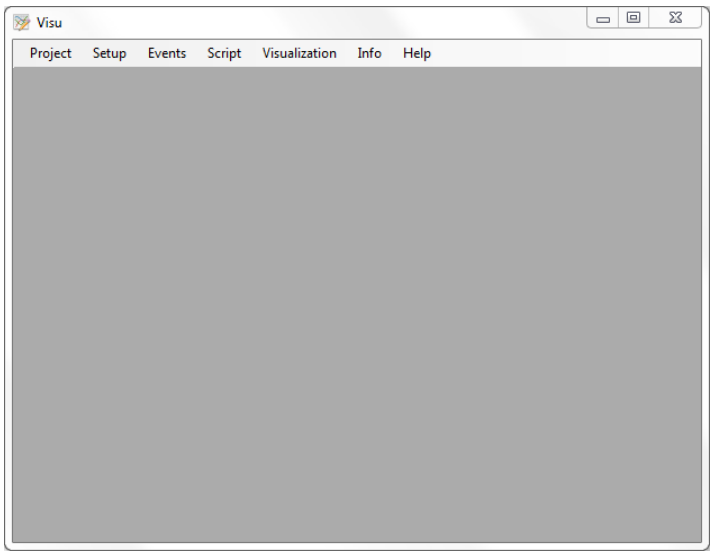


Abbildung 1: Default Oberfläche

Um eine Übersicht über die Funktionen und Möglichkeiten des Toolmonitors Visu zu erhalten, kann über **Project → Presets → Sample** ein Beispiel - Preset geladen werden, welches eine Übersicht über sämtliche Funktionen des Monitors bietet.

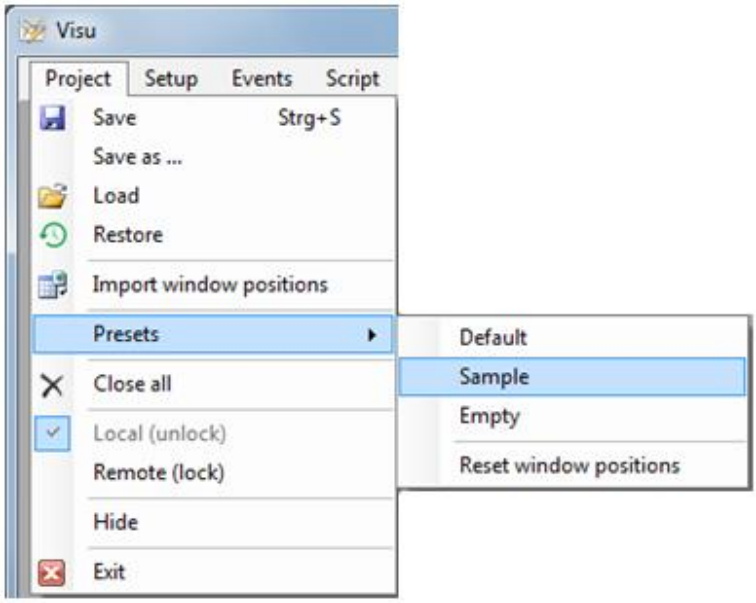


Abbildung 2: Sample Preset laden

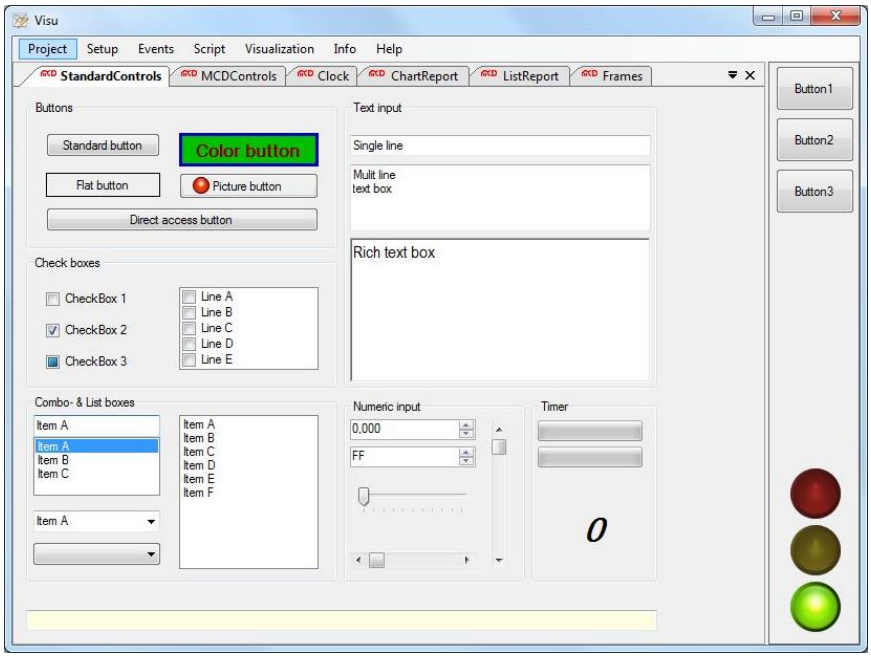


Abbildung 3: Oberfläche Sample Preset

Um eigene Funktionen und Prozeduren zu erstellen erreicht man den **Script Editor** über den Menüpunkt **Script → Script Editor**.

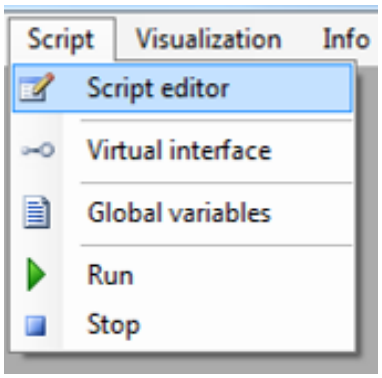


Abbildung 4: Script Editor öffnen

4. Handbuch Software

4.1. Designer

Im **Designer** des Toolmonitor Visu können beliebige Bedien - und Benutzeroberflächen erstellt werden. Hierzu sind Standard - Komponenten und zusätzlich die MCD Controls vorhanden.

4.1.1. Designer öffnen

Um den *Designer* des zuvor erstellten Projekts zu öffnen, wählt man unter dem Menüpunkt **Visualization → Designer** das gewünschte Projekt aus (hier „Sample“).

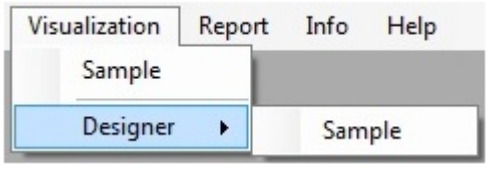


Abbildung 5: Designer öffnen

4.1.2. Neues Formular anlegen

Um im *Designer* ein neues Formular anzulegen, betätigt man die Schaltfläche **New** im Menü **Form**. In diesem Menü kann man auch eine bereits bestehende Oberfläche importieren, oder die neu erstellte exportieren. Die Schaltfläche **Clear** löscht die aktuelle Oberfläche.

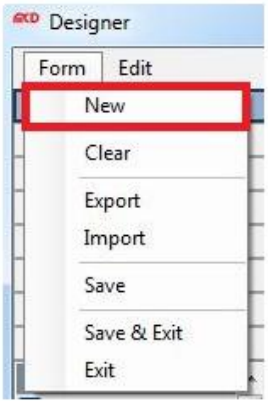


Abbildung 6: Neues Formular erstellen

Nach Erstellen eines neuen Formulars kann man diesem einen Namen geben. Dieser **Form Title** wird automatisch abgefragt und ist notwendig, damit das neue Formular erstellt werden kann.

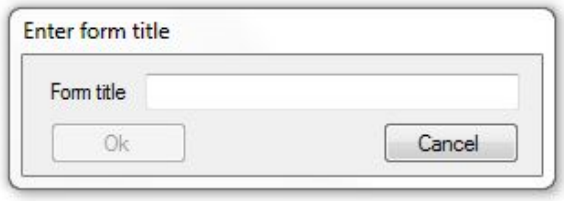


Abbildung 7: Formular benennen

4.1.3. Einstellungen und Parameter

Auf der rechten Seite im *Designer* befindet sich eine Liste mit den Einstellungen und Parametern des ausgewählten Objektes. Diese konfigurierbaren Parameter werden auch **Properties** genannt und sind bei jedem Formular und Control unterschiedlich. Bei den meisten Objekten lässt sich dort die exakte Größe und Position festlegen. Bei analogen Anzeigen lässt sich beispielsweise auch der Minimal -, Maximalwert und die Skalierung der Anzeige konfigurieren.

Um die *Properties* eines Controls / eines Formulars verändern zu können, muss dieses im Designfenster (mittleres Fenster) markiert werden.

Im unten stehenden Beispiel wurde der Button1 (linker roter Rahmen) selektiert. Somit können in der rechten Leiste beispielsweise die Farbe (oberer roter Rahmen) oder der Text des Buttons (unterer roter Rahmen) verändert werden.

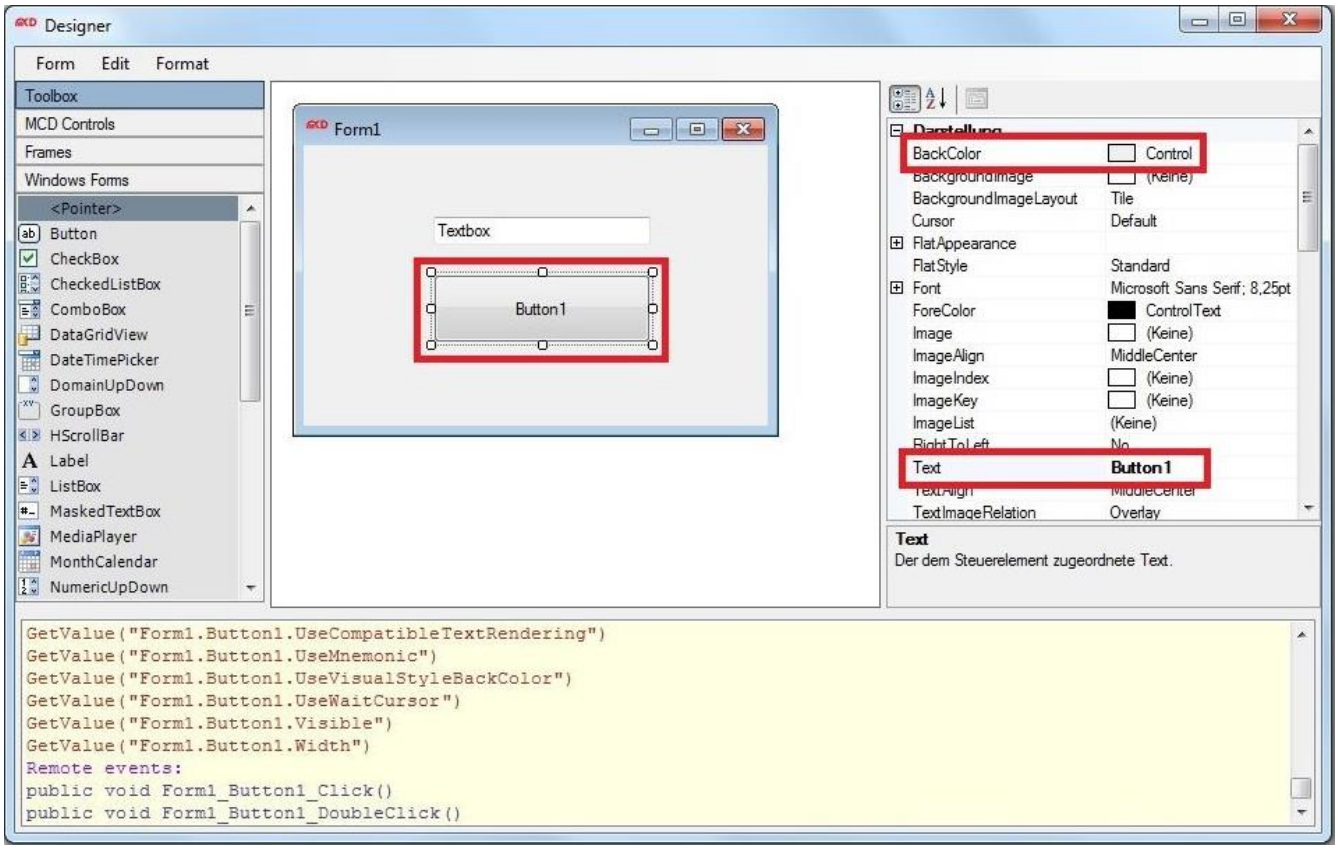


Abbildung 8: Eigenschaften eines Buttons

4.1.4. Menüstruktur anpassen

Über die Eigenschaften des Formulars kann auch die Struktur des **Menu Trees** angepasst werden. Nachdem das gewünschte Formular im *Designer* selektiert wurde, steht unter den *Properties* am rechten Bildrand die Eigenschaft *Menu Tree* zur Verfügung. Hier kann man festlegen unter welcher / welchen Untergruppe / n die Oberfläche im Toolmonitor angezeigt werden soll.

Abbildung 9: Menu Tree

In diesem Beispiel heißt das Formular „Test“ und beinhaltet eine analoge Spannungsanzeige. Die Struktur der Untermenüs, in der die Oberfläche „Test“ angezeigt wird, ist in Abbildung 10 zu sehen. „Visualization“ steht an erster Stelle und ist somit der Menüpunkt, der in der oberen Leiste des Toolmonitors zu sehen ist. „Voltage“ ist ein Unterpunkt im Menü „Visualization“. Der nächste Punkt wäre dann ein Unterpunkt von „Voltage“.

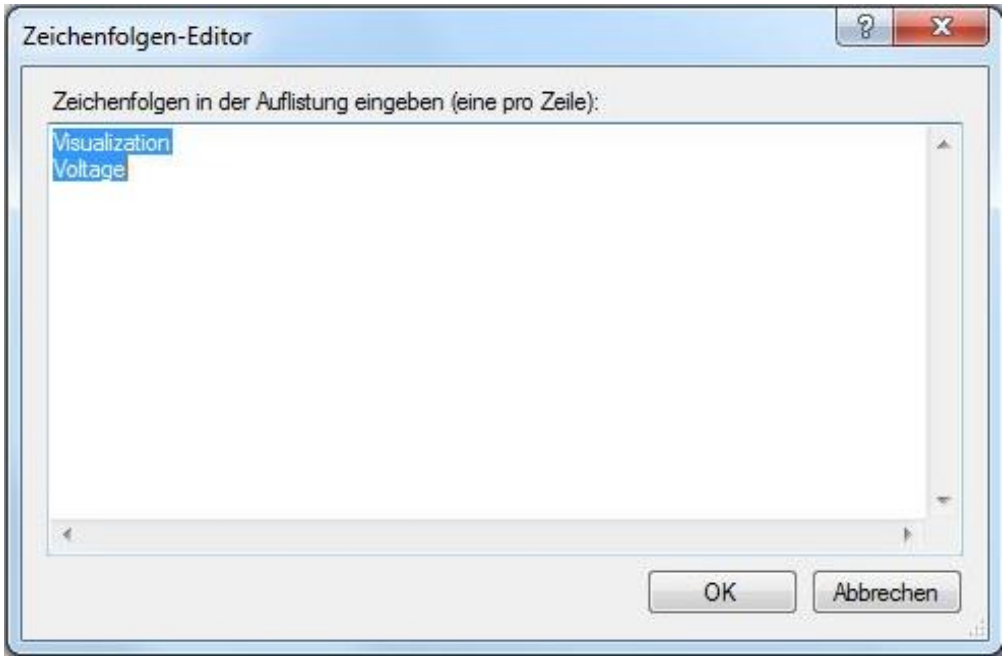


Abbildung 10: Struktur des Untermenüs

Nach diesem Schema lassen sich beliebig viele Unterpunkte erstellen, was allerdings nicht immer zur Übersichtlichkeit beiträgt. In unserem Beispiel wird die „Oberfläche Test“, wie in Abbildung 11 zu sehen, unter „Voltage“ angezeigt.

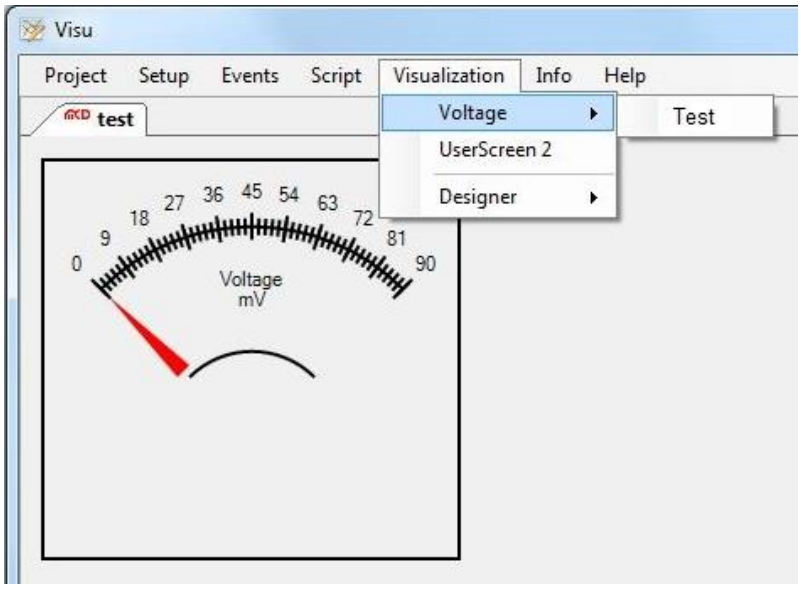


Abbildung 11: Untermenüs im Toolmonitor

4.1.5. Properties und Events

Im unteren Fenster des *Designers* findet man eine Liste der veränderbaren *Properties* und **Events** eines Controls. In der ersten Abbildung findet man beispielsweise die **Remote Events** eines Buttons. Die *Events* und die veränderbaren *Properties* sind bei jedem Control unterschiedlich. Die jeweiligen *Remote Events* sind unter **Controls** gelistet. *Properties* eines Controls können mit der Funktion **GetValue** im **Script Editor** ausgelesen und mit **SetValue** verändert werden.

```
GetValue("Test.Form.Visible");
SetValue("Test.Form.Size", "295; 305");
```

Abbildung 12: Funktionen *GetValue* und *SetValue*

Mit den *Events* ist es möglich, Reaktionen auf Veränderungen oder Interaktionen mit einem Control oder einem Formular zu gestalten.

```
public void Test_Button1_Click(object sender, EventArgs e)
{
    ...
}
```

Abbildung 13: Darstellung eines Events

Mit einem Doppelklick auf das gewünschte *Event* springt der *Designer* sofort in den **Script Editor**, in dem man die Reaktionen auf das entsprechende *Event* programmieren kann.

```
GetValue("example.Button1.UseCompatibleTextRendering")
GetValue("example.Button1.UseMnemonic")
GetValue("example.Button1.UseVisualStyleBackColor")
GetValue("example.Button1.UseWaitCursor")
GetValue("example.Button1.Visible")
GetValue("example.Button1.Width")
Remote events:
public void example_Button1_Click()
public void example_Button1_DoubleClick()
public void example_Button1_KeyDown()
public void example_Button1_SizeChanged()
public void example_Button1_TextChanged()
```

Abbildung 14: Properties und Events

4.1.6. Controls

Auf der linken Seite des *Designers* befindet sich eine Liste von verwendbaren **Controls**.

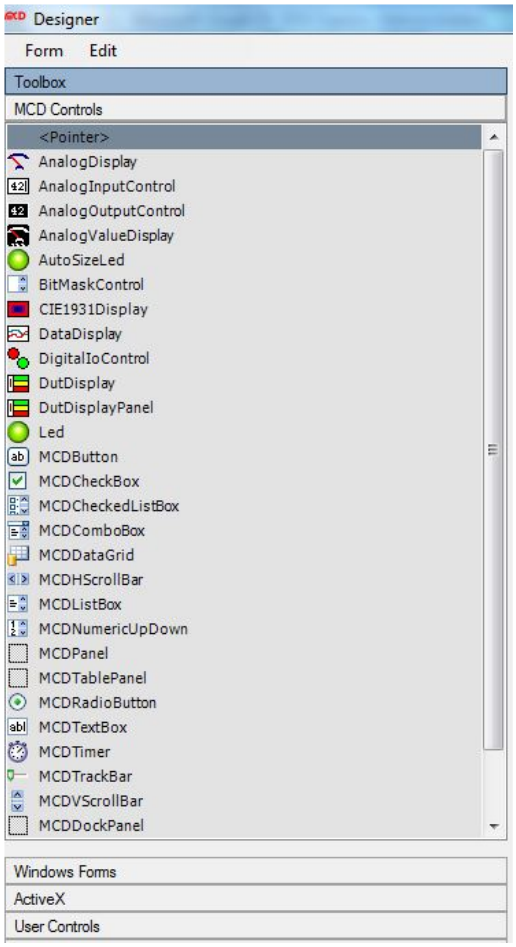


Abbildung 15: Liste der verfügbaren Controls

Neben den standardmäßigen **Windows Controls** stehen unter dem Reiter **MCD Controls** zusätzliche, MCD eigene Controls zur Verfügung. Unter diesen *MCD Controls* befinden sich Controls, wie zum Beispiel das **CurveAnalysis Control**, die nicht standardmäßig im Toolmonitor enthalten sind, sondern erst über eine Lizenz freigeschalten werden müssen. Der Unterschied zwischen einem *Standard Windows Control* und einem *MCD Control* wird in folgendem Beispiel deutlich.

Wird beispielsweise ein Formular mit einem MCD Button über den TestManager gesteuert, also **Remote**, so wird der Button in der Oberfläche automatisch gesperrt und kann nicht manuell bedient werden. Ein Windows Button wird hingegen nicht gesperrt und kann weiterhin auch manuell bedient werden (rechte Abbildung). So kann je nach Anwendungsfall der gewünschte Typ ausgewählt werden. Um dieses Verhalten im Toolmonitor zu testen, kann über den Menüpunkt **Project → Local / Remote** die Ansicht zwischen **Local und Remote** gewechselt werden (linke Abbildung).

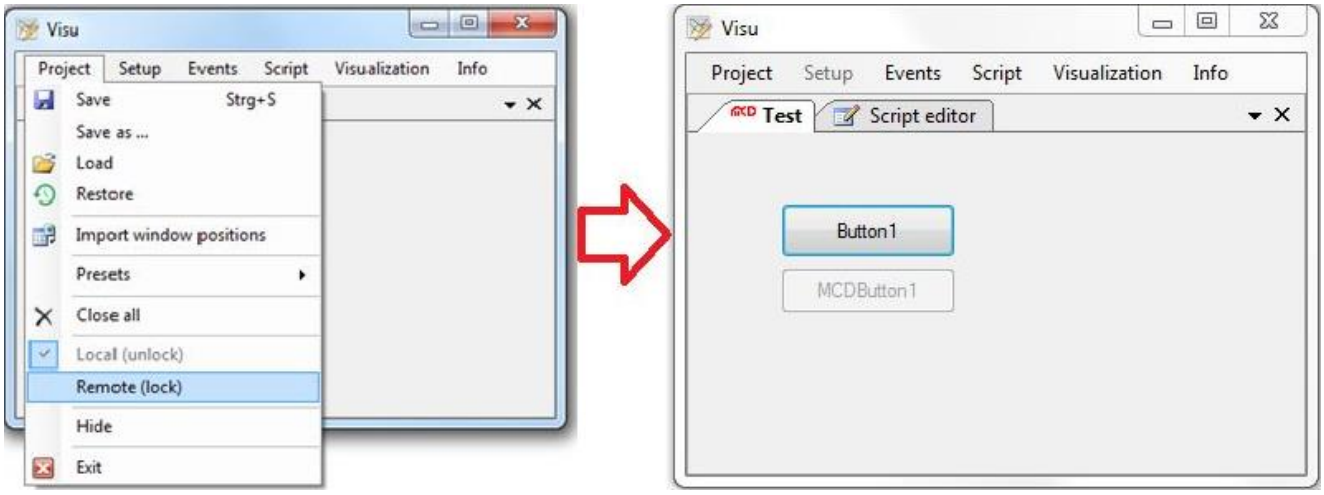


Abbildung 16: Windows Controls und MCD Controls

4.2. Script

4.2.1. Script Editor

Im **Script Editor** können eigene Prozeduren und Funktionen zur Erweiterung des Toolmonitors erstellt werden. Die Programmierung erfolgt in der Programmiersprache C# des .Net - Frameworks. Vorhandene .Net - Assemblies können in den Scripten genutzt werden.

Der **Script Editor** besteht hauptsächlich aus drei Bereichen. Dem **Texteditor**, dem **Explorer** und der **Status - / Debuganzeige**. Die meisten Einstellungen und Zusatzfunktionen sind über das Kontextmenü des *Script Editors* erreichbar und werden im Folgenden beschrieben.

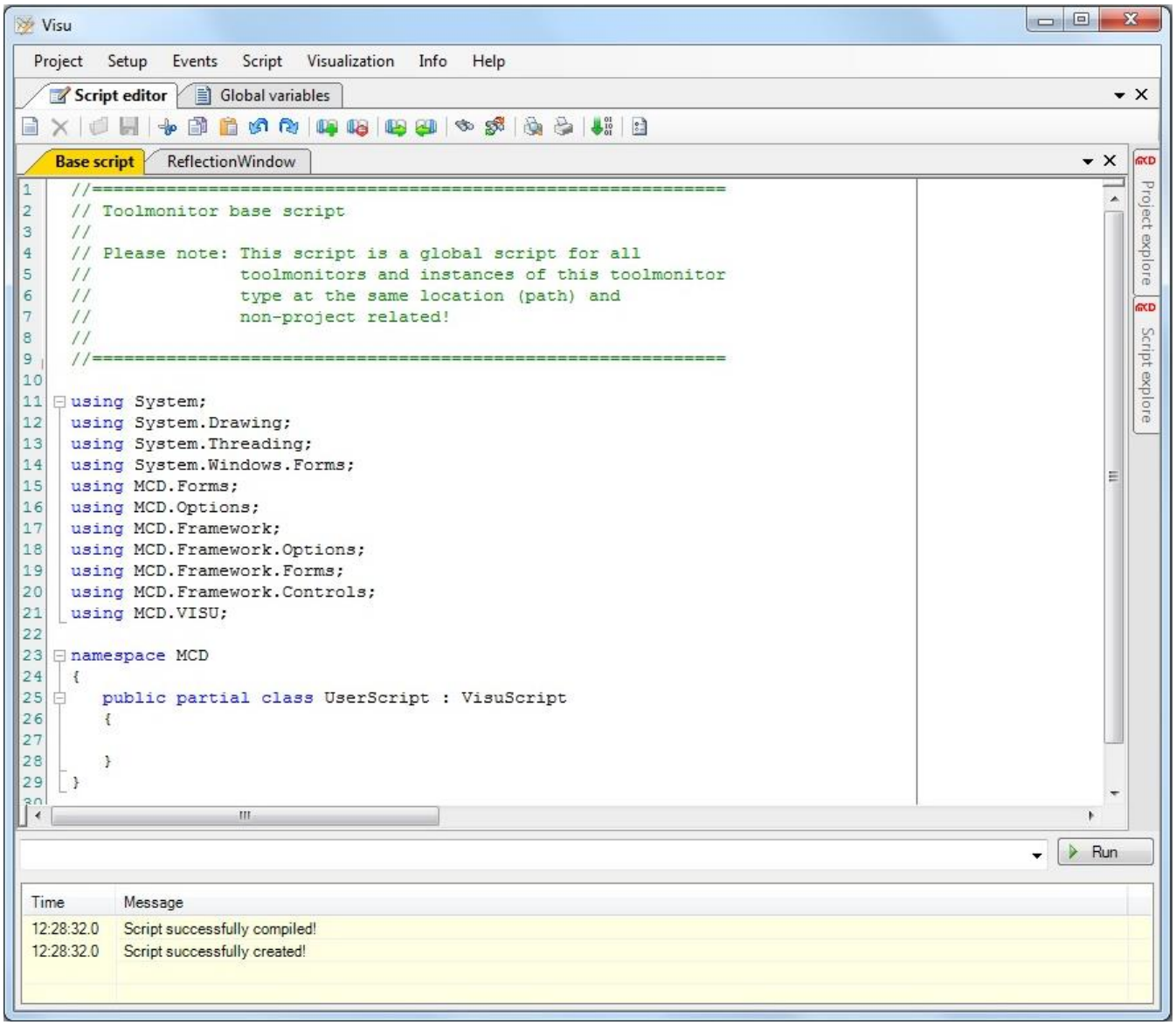


Abbildung 17: Oberfläche Script Editor

4.2.1.1. Base Script

Das sogenannte **Base Script** ist ein globales Script für alle Toolmonitore und deren Instanzen. Das *Base Script* ist das Standardscript, das sich beim ersten Öffnen des *Script Editors* öffnet. Alle weiteren Scripte bzw. Tabs, die erstellt werden, können auf das *Base Script* zugreifen.

Im unteren Beispiel wurde im *Base Script* eine Funktion „Testfunktion()“ erstellt, die über eine Funktion „Testfunktiontab()“ im Script „tab1“ aufgerufen wird. Ebenso besteht im Script „tab1“ Zugriff auf die Variablen (hier „Testwert“) des *Base Scripts*, wie ebenfalls im unteren Beispiel zu sehen ist.

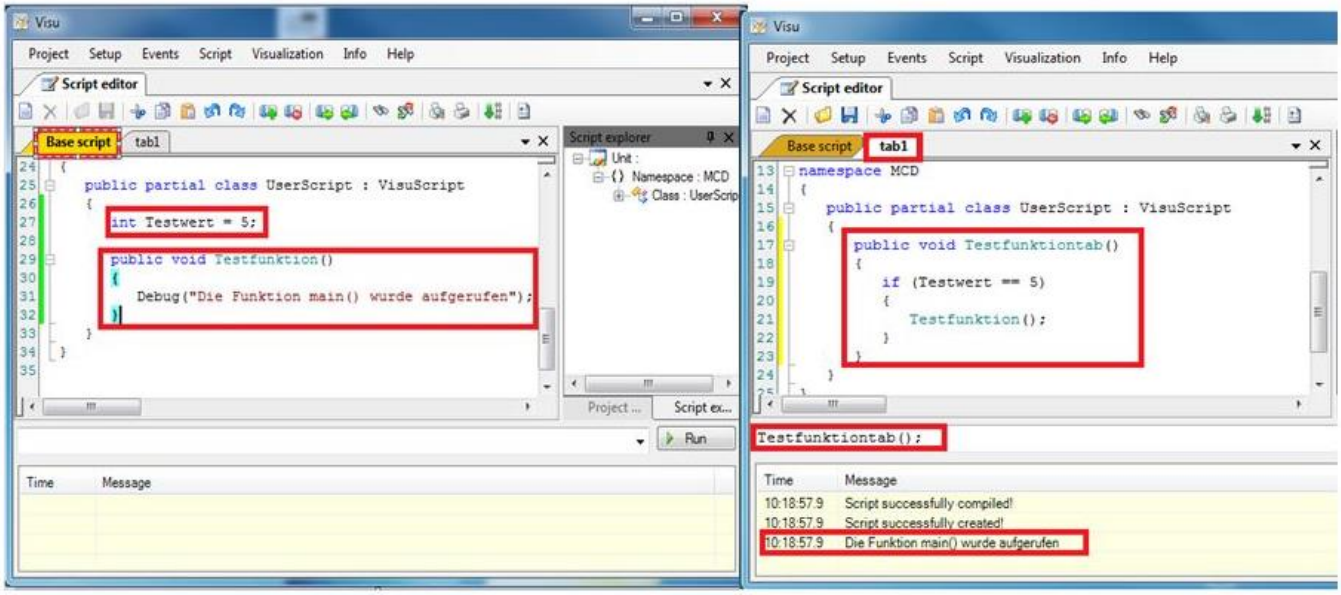


Abbildung 18: Funktionsaufruf im Base Script

Das *Base Script* eines Toolmonitors ist das einzige Script, das als eigenständige C# Datei (.cs) gespeichert wird. Die übrigen Scripte, Einstellungen und Inhalte des Projekts werden in der Projektdatei als .xml gespeichert.

Wenn also ein Projekt mit mehreren Scripten auf einen anderen PC umgezogen werden soll, muss die Projektdatei und das dazugehörige *Base Script* mitgenommen werden. Der Vorteil dieser Struktur besteht darin, dass mehrere Scripte und auch mehrere verschiedene Toolmonitore gleichzeitig auf den Inhalt / die Funktionen des *Base Scripts* zugreifen können.

4.2.1.2. Explorer

Auf der rechten Seite der Toolmonitor - Oberfläche befindet sich der **Script - bzw. Project Explorer**. Hier werden die eingefügten **Usings**, sowie die bestehenden Variablen und Funktionen hierarchisch angezeigt (**Script Explorer**) bzw. die verschiedenen Scripte aufgelistet (**Project Explorer**). Über einen Doppelklick auf ein Element springt der Cursor im Texteditor automatisch zu dieser Stelle im Quelltext.

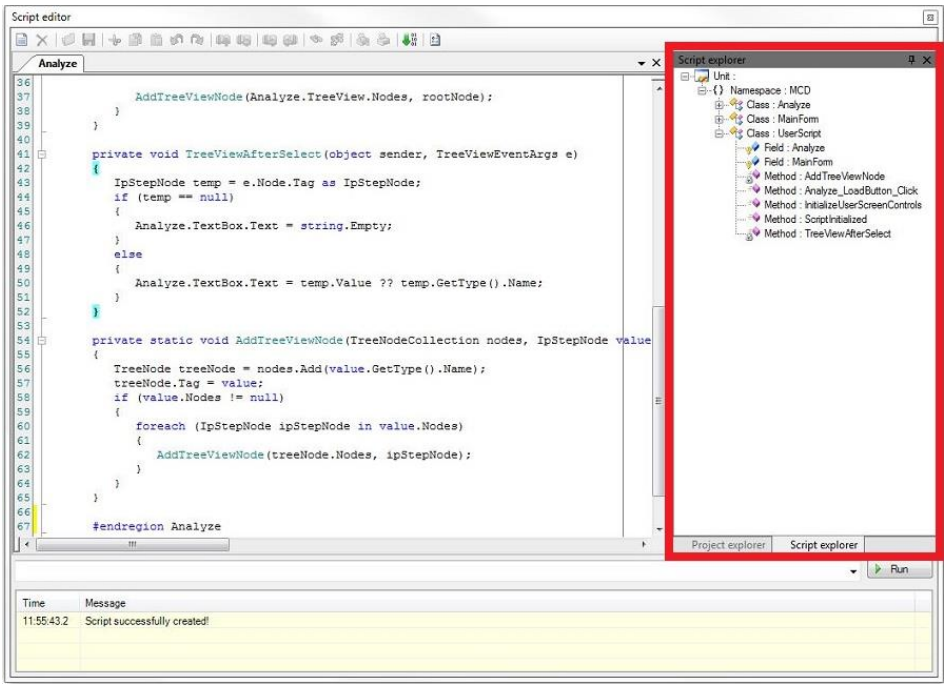


Abbildung 19: Script Explorer

Im sogenannten **Script Explorer** werden alle **Namespaces** mit ihren entsprechenden Klassen, alle Funktionen und alle angelegten Variablen hierarchisch angezeigt. Bei einem Doppelklick auf eines der angezeigten Elemente springt der Cursor im Texteditor automatisch zu der entsprechenden Stelle im Quellcode.

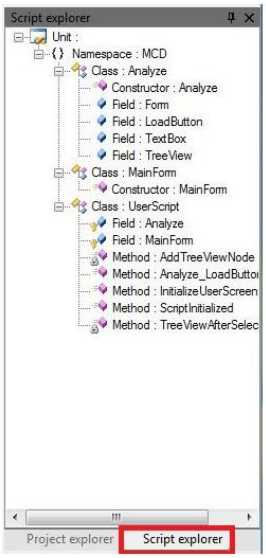


Abbildung 20: Script Explorer anzeigen

Im sogenannten **Project Explorer** erhält man eine Übersicht über alle verfügbaren Scripte. Bei einem Doppelklick auf ein Script, wird das entsprechende Script automatisch geöffnet.

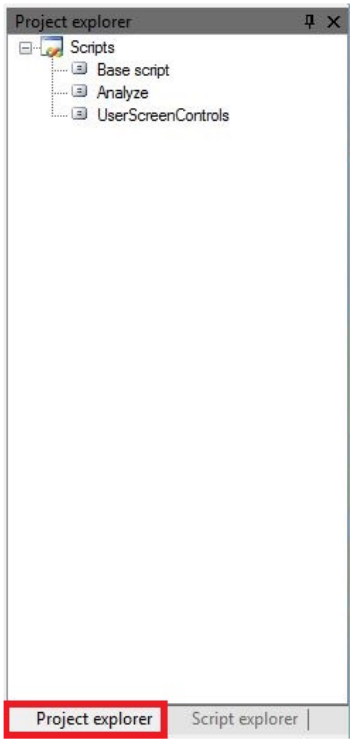


Abbildung 21: Project Explorer anzeigen

4.2.1.3. Kontextmenü

Die verschiedenen Optionen und Einstellungen des *Script Editors* können über das **Kontextmenü** erreicht werden. Hierzu betätigt man die rechte Maustaste innerhalb des Editors. Nach dem Klick öffnet sich das unten zu sehende *Kontextmenü*. Eine Übersicht über die einzelnen Funktionen der *Kontextmenü - Einträge* findet man in der untenstehenden Tabelle. Einige dieser Funktionen sind auch über die Icons an der oberen Menüleiste des Texteditors erreichbar.

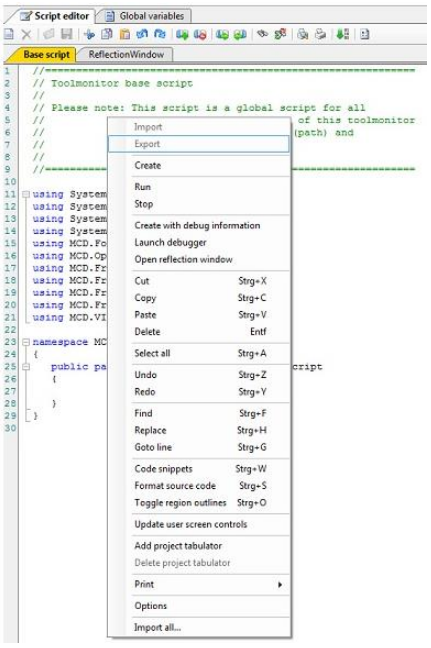


Abbildung 22: Kontextmenü des Script Editors

In der nachfolgenden Tabelle werden die Einträge des Kontextmenüs, die zugehörigen Funktionen und, falls vorhanden, die dazugehörigen Symbole aus der Symbolleiste aufgezeigt.

	Kontextmenü Eintrag	Funktion
	Import	Ein bereits bestehendes Script kann importiert werden
	Export	Das erstellte Script kann extern gespeichert werden
	Create	Das Script wird erstellt
	Run	Das Script wird gestartet
	Stop	Das bereits laufende Script wird angehalten
	Create with debug information	Das Script wird erstellt und die Informationen vom Debugger angezeigt
	Launch Debugger	Der Debugger wird gestartet
	Open reflection window	Das Überichtsfenster für das Projekt wird geöffnet. Rechtsklick für refresh
	Cut	Ein markiertes Element wird ausgeschnitten
	Copy	Das markierte Element wird in den Zwischenspeicher kopiert
	Paste	Das Element aus dem Zwischenspeicher wird eingefügt
	Delete	Das markierte Element wird gelöscht
	Select all	Der gesammte Inhalt des Scripts wird markiert
	Undo	Macht den letzten Arbeitsschritt rückgängig
	Redo	Wiederholen der zuletzt rückgängig gemachten Aktion
	Find	Öffnet die Suchfunktion
	Replace	Öffnet das suchen und ersetzen Fenster
	Goto line	Springt zu der eingegebenen Zeile
	Code snippet	Hier finden sich häufig benötigte, vorgefertigte Codestücke
	Format source code	Der Code wird automatisch programmiertypisch formatiert
	Toggle region outlines	#regions werden ein-/ausgeklappt
	Update user screen controls	Update der erstellten Benutzeroberfläche
	Add project tabulator	Fügt dem Projekt einen neuen Tabulator (Tab) hinzu
	Delete project tabulator	Der aktuell geöffnete Tabulator (Tab) wir gelöscht
	Print	Menü um das Script zu drucken
	Options	Hier öffnen sich die Syntax Einstellungen des Script editors
	Import all...	Importiert alle Skripte von einer MCD Toolmonitor xml-Datei

Abbildung 23: Funktionsübersicht des Kontextmenüs

4.2.1.4. Statusanzeige

Der dritte Bereich mit der **Status - / und Debuganzeige** befindet sich unten im Fenster des *Script Editors*.

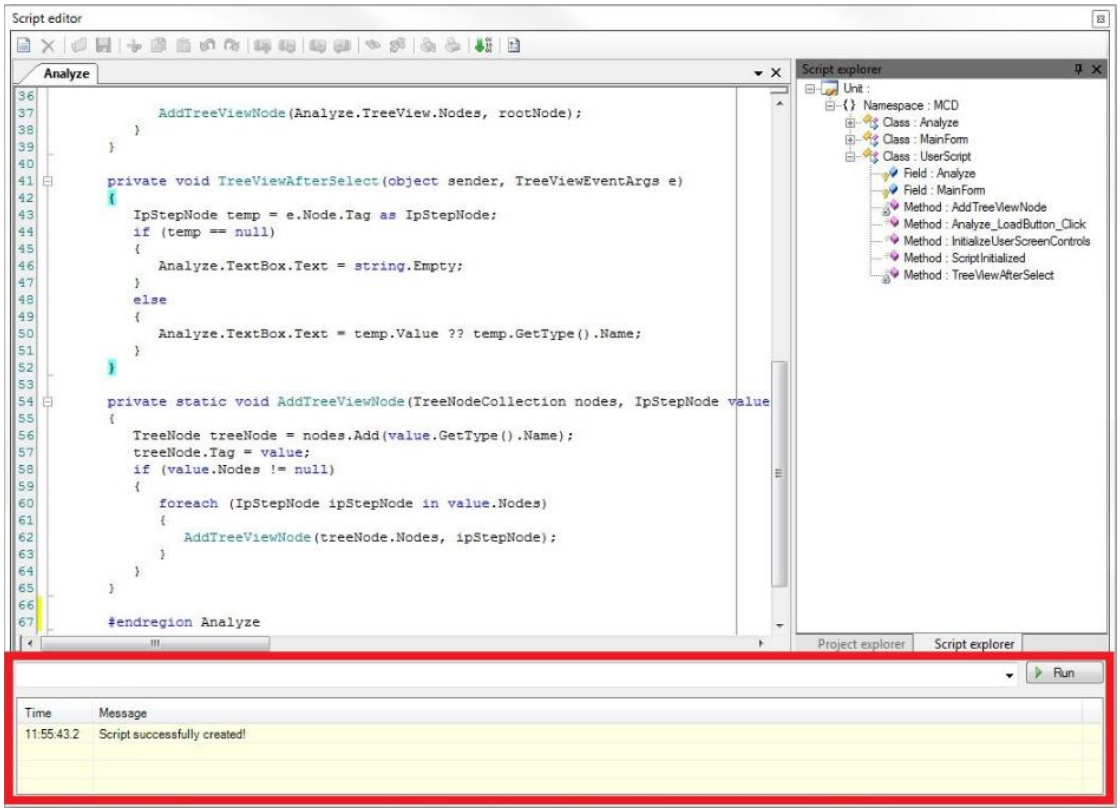


Abbildung 24: Statusanzeige

Dieser Bereich gibt die Möglichkeit, einzelne Funktionen und Befehle über die *Debug - Zeile* und Schaltfläche zu debuggen. Des Weiteren werden im unteren Fenster Statusmeldungen und *Debug - Ausgaben* angezeigt.

Um eine gewünschte Funktion aufzurufen wird in die *Debug - Zeile* der Name der Funktion eingegeben und das Debugging mit dem *Debug - Button* rechts gestartet. Um sich zu Test - oder Kontrollzwecken Werte von Variablen anzeigen zu lassen, steht eine *Debug - Anweisung* zur Verfügung. Im unteren Beispiel wird die **Methode Main()** aufgerufen und die Werte der Variablen Testwert 0 - Testwert 5 im *Debug - Fenster* ausgegeben.

Die eingegebenen *Debug - Befehle* werden in der **Registry** gespeichert und können bei Bedarf für einen erneuten Aufruf ausgewählt werden.

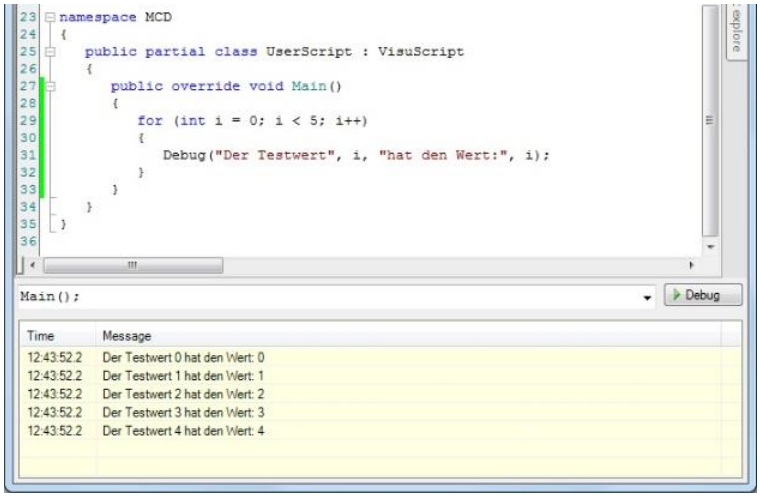


Abbildung 25: Debug - Ausgabe der Methode Main()

4.2.1.5. Texteditor

Im linken Fenster der *Script Editor* Oberfläche befindet sich der eigentliche **Texteditor** zur Erstellung des Quelltextes.

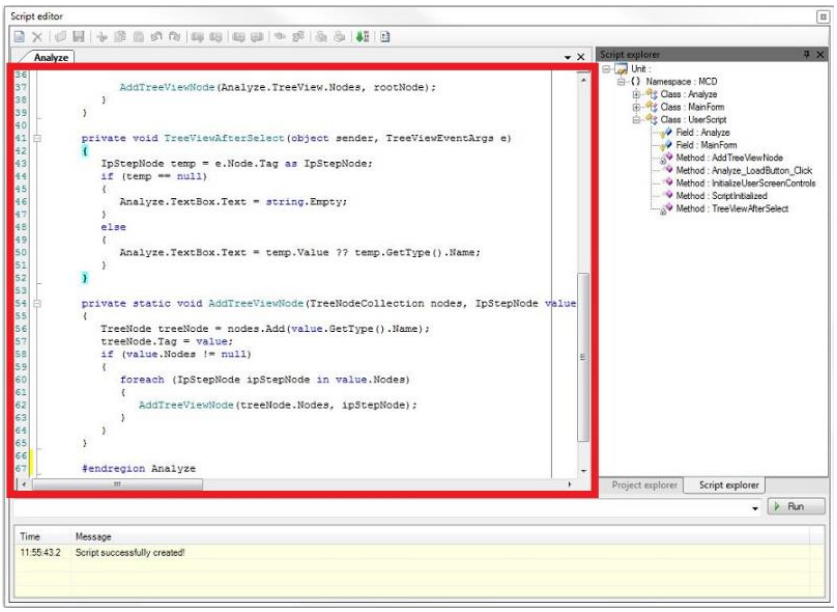


Abbildung 26: Texteditor Oberfläche

Wenn Sie mit einem leeren Script beginnen und die notwendigen *Usings* nicht manuell eingeben wollen, kann ein beliebiges Zeichen in das leere Textfeld geschrieben werden. Nach einem Klick auf den Befehl **Create** des *Kontextmenüs*, werden automatisch die benötigten *Usings* eingefügt.

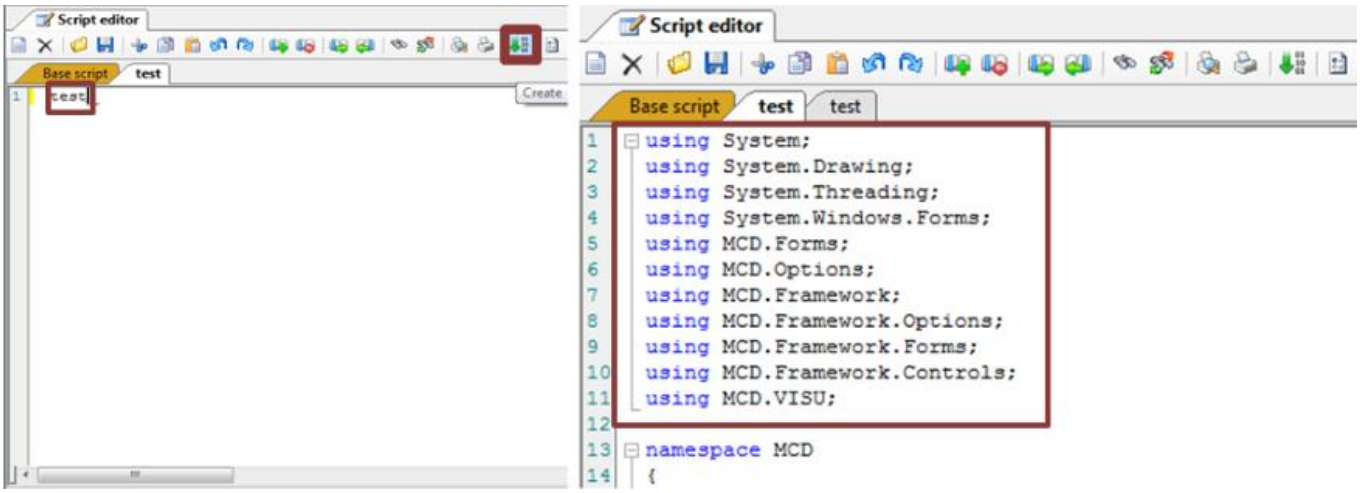


Abbildung 27: Using in leeres Script einfügen

Der *Texteditor* bietet verschiedene Möglichkeiten zur automatischen Codevervollständigung (*Intelli - Sense*). So können Eigenschaften und Funktionen für bestehende Variablen angezeigt werden. Bei der Eingabe eines Punktes nach einer Variablen werden automatisch die verfügbaren Funktionen und Eigenschaften angezeigt. Im unteren Bild werden z.B. die Eigenschaften einer *double - Variablen* aufgerufen, um diese in einen *String* zu wandeln (dies erfolgt über die Methode „.ToString()“).

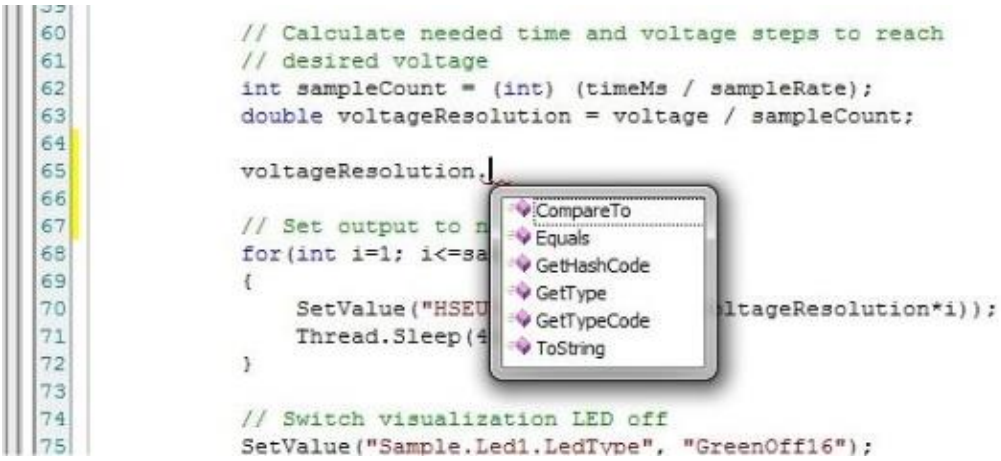


Abbildung 28: Codevervollständigung (variable. / function.)

Des Weiteren gibt es die Möglichkeit, über den **Operator this.** und ein *Kontextmenü* die zur Verfügung stehenden Funktionen abzurufen. Im Beispiel ist der Aufruf der Funktion *SetEvent()* gezeigt. Geben Sie hierzu einfach den Text „this.“ ein. Nach der Eingabe des Punkts wird automatisch ein Fenster geöffnet, in welchem die Funktionen und deren Parameter angezeigt werden. Wählen Sie die gewünschte Funktion über Doppelklick oder Enter aus.

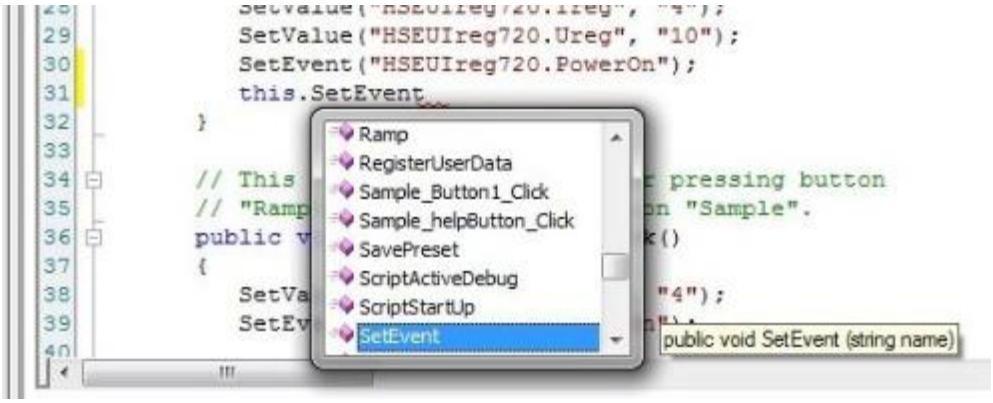


Abbildung 29: Codevervollständigung (this.)

Haben Sie den richtigen Funktionsnamen eingegeben, werden Ihnen bei der Eingabe der geöffneten Klammer die möglichen Parameter angezeigt. Ist die genutzte Funktion überladen, können Sie eine Auswahl über die Pfeilsymbole treffen, um die verschiedenen Parameter anzuzeigen.

```
26      public override void Main()
27      {
28          SetValue("HSEUIreg720.Ireg", "4");
29          SetValue("HSEUIreg720.Ureg", "10");
30          SetEvent("HSEUIreg720.PowerOn");
31          this.SetValue(
32      }   ▲1 of 3▼ void CamtecScript.SetValue (string name, double value)
33
```

Abbildung 30: Anzeigen von Parametern

Ausführliche Hilfe zu programmierspezifischen Fragen erhalten Sie über das integrierte Onlineangebot von Microsoft (MSDN). Über das Tastenkürzel „Strg + F1“ wird „MSDN“ automatisch in Ihrem Internet - Browser geöffnet. Sie können ein Schlüsselwort markieren bzw. den Cursor in das Wort setzen, um dann über „Strg + F1“ die „MSDN - Hilfe“ für exakt dieses Schlüsselwort zu erhalten. Im Bild ist z.B. die Dokumentation für das Schlüsselwort „double“ aufgerufen worden.

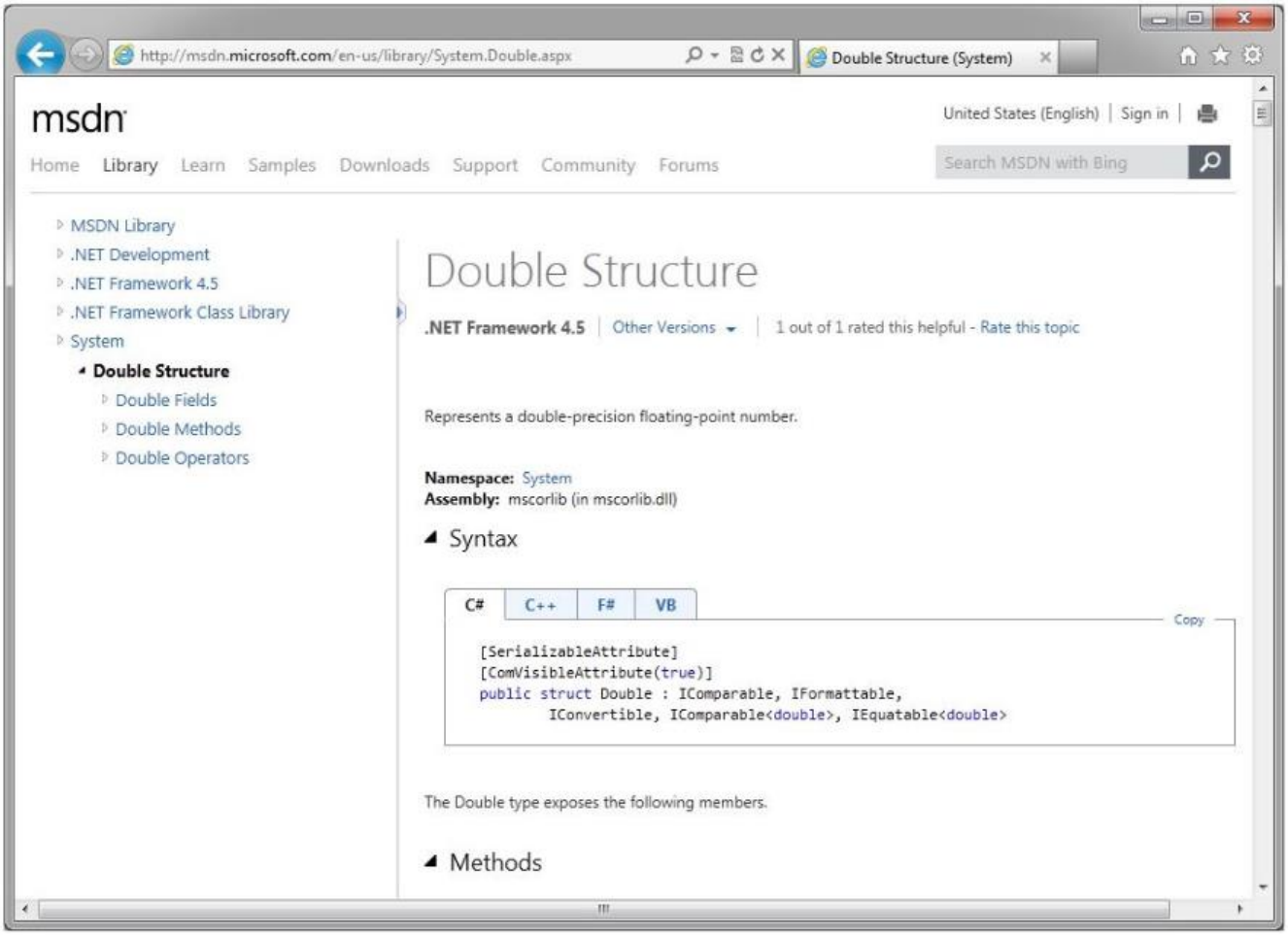


Abbildung 31: MSDN Onlinehilfe (Strg + F1)

Die Darstellung des Quelltextes können Sie über den Punkt **Optionen** im *Kontextmenü* einstellen. Hier ist es z.B. möglich die Farben für bestimmt Schlüsselwörter, Strings oder Zahlen festzulegen. Des Weiteren kann z.B. auch die Anzahl der Leerzeichen für ein Tab oder die Aufteilung des Editorfensters angepasst werden. Im Bild sieht man die Hauptseite der *Optionen*.

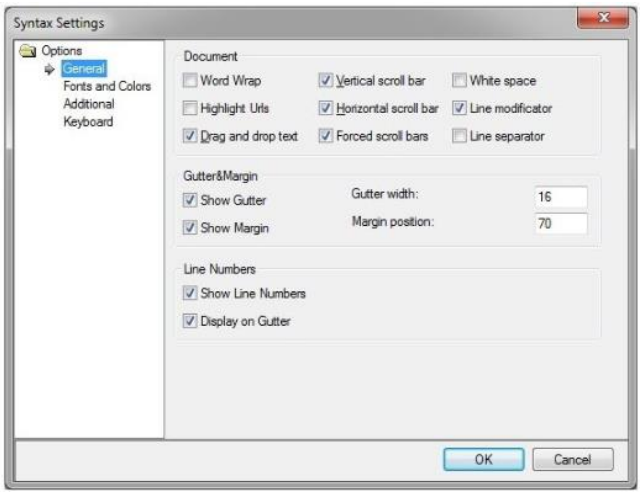


Abbildung 32: Optionen des Script Editors (Kontextmenü)

Wollen Sie verschiedene Bereiche Ihres Scripts parallel ansehen und bearbeiten, so gibt es die Möglichkeit ein zweites Fenster zu öffnen. Dieses erreichen Sie über die rechteckige Schaltfläche rechts oben (über der Scroll-Leiste). Führen sie die Maus auf die Schaltfläche und ziehen Sie dann ein weiteres Fenster von oben auf.

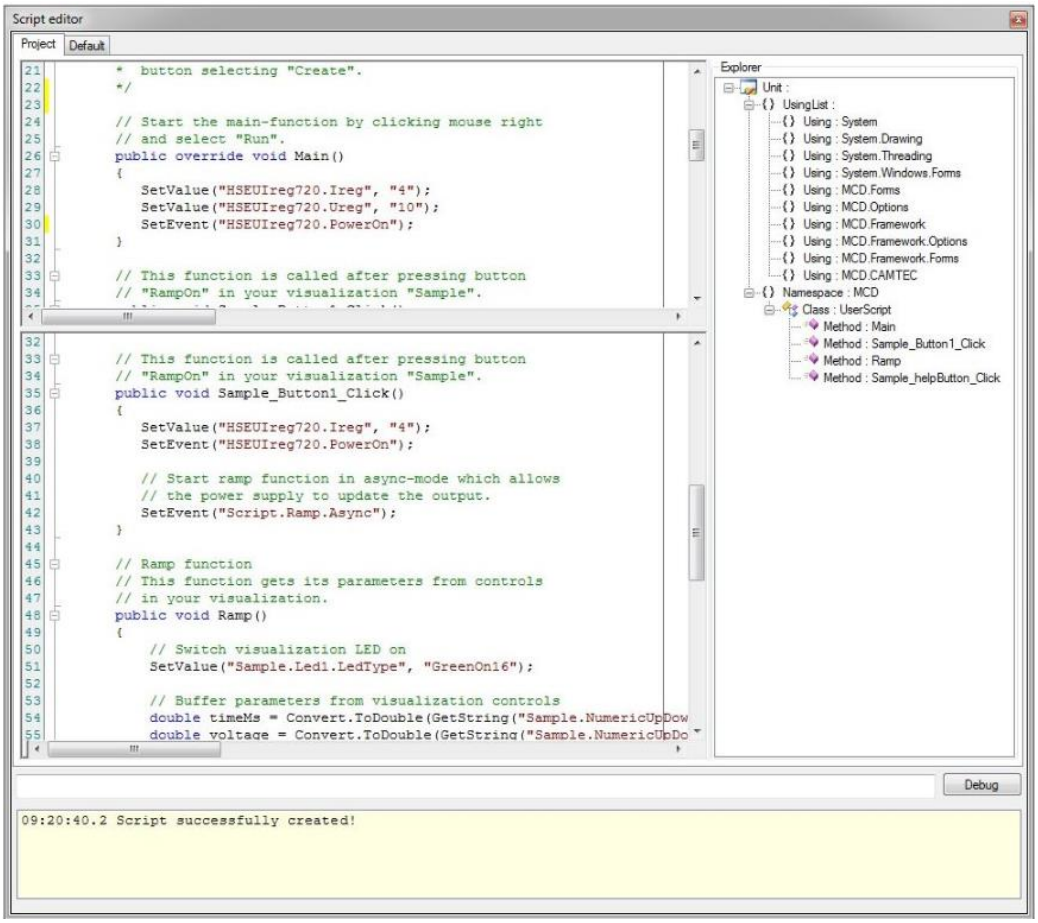


Abbildung 33: Zweites Fenster (Schaltfläche oben rechts)

4.2.2. Run - Befehl

Über diesen Befehl wird ausschließlich die **Methode Main()** im Script asynchron aufgerufen, sofern diese vorhanden ist. Es gibt zwei Möglichkeiten den **Run - Befehl** auszuführen:

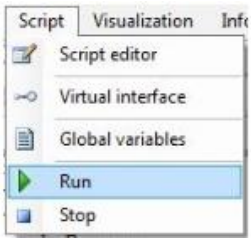


Abbildung 34: Möglichkeit 1 (Script → Run)

Abbildung 35: Möglichkeit 2 (Rechtsklick → Run)

4.2.3. Stop - Befehl

Ein eventuell laufendes Script wird abgebrochen. Es gibt zwei Möglichkeiten das laufende Script zu stoppen:

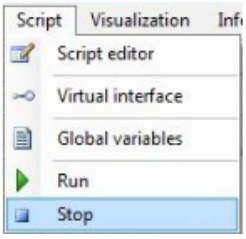


Abbildung 36: Möglichkeit 1 (Script → Stop)

Abbildung 37: Möglichkeit 2 (Rechtsklick → Stop)

4.2.4. DLLs einbinden

In die *Script Engine* eines Toolmonitors können über die Script Einstellungen externe .Net - Assemblies eingebunden werden, deren Inhalt danach im Toolmonitor zur Verfügung steht.

Wie im unteren Bild zu sehen, sind standardmäßig bei einem neuen Projekt bereits drei wichtige .Net - Assemblies eingebunden: die *System.dll*, die *System.Drawing.dll* und die *System.Windows.Form.dll*.

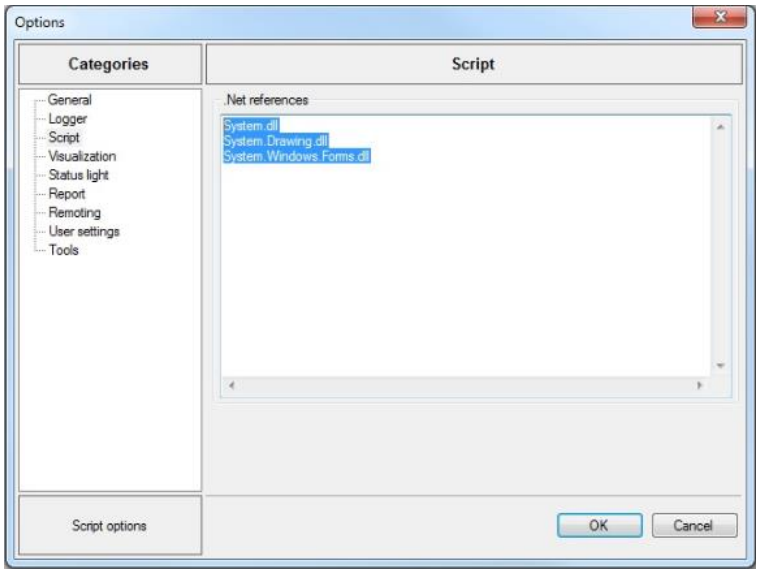


Abbildung 38: DLLs in einem neuen Projekt

Um weitere .Net - Assemblies einzubinden, wird der Name der .**DLL**, wie unten zu sehen ist, in die Liste eingetragen und danach mit „OK“ bestätigt. Hier wurden zur Veranschaulichung die *System.Xml.dll* und eine selbst erstellte *ClassLibraryBeispiel.dll* eingebunden. Wichtig ist, dass sich die *DLLs* im selben Verzeichnis wie der Toolmonitor befinden. Um im Eingabefenster eine Zeile tiefer zu springen, muss „Strg + Enter“ gedrückt werden.

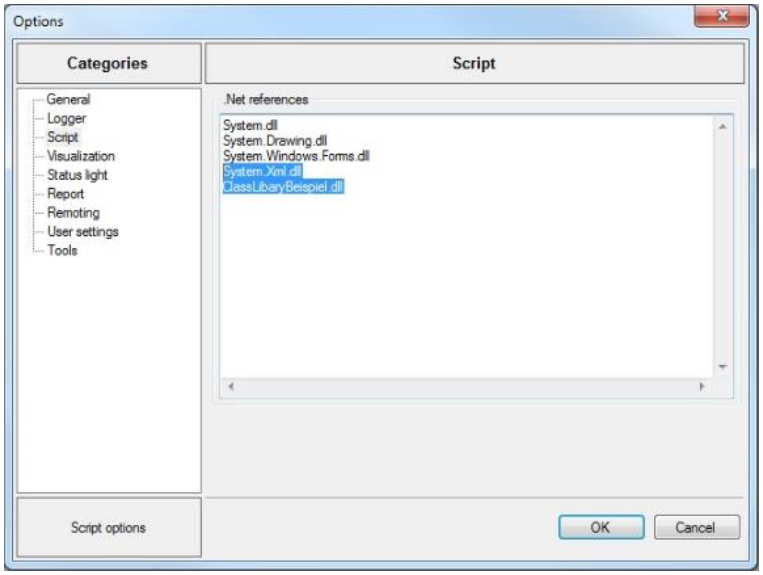


Abbildung 39: Neu hinzugefügte DLLs

4.2.5. Global Variables

Über das Fenster **Global Variables** können globale Variablen angelegt, gelöscht oder verändert werden.

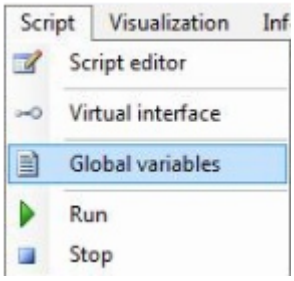


Abbildung 40: Global Variables öffnen

Neue Variablen können über einen Rechtsklick hinzugefügt („Add“) bzw. gelöscht („Delete“) werden. Hierbei stehen vier Datentypen zur Auswahl (*Real, String, Vector, StringVector*). Die Syntax zu den jeweiligen Datentypen ist dem unteren Beispiel zu entnehmen.

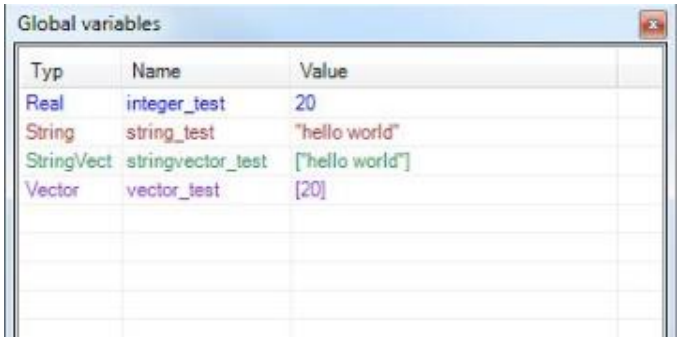


Abbildung 41: Beispiel Global Variables

4.2.6. Virtual Interface

Mit Hilfe des Formulars für das **virtuelle Interface** können die dort verfügbaren Funktionen des Toolmonitors ausgeführt und getestet werden. Es stellt die Schnittstelle zur Kommunikation über die COM Schnittstelle dar.

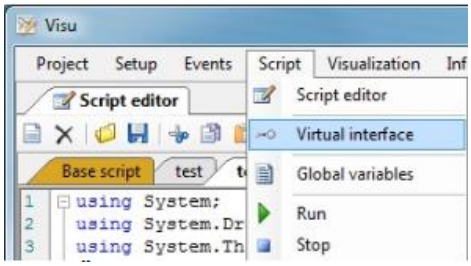


Abbildung 42: Virtual Interface öffnen

Um einen Toolmonitor über diese Schnittstelle steuern zu können, muss dieser zuvor als COM Server registriert werden.

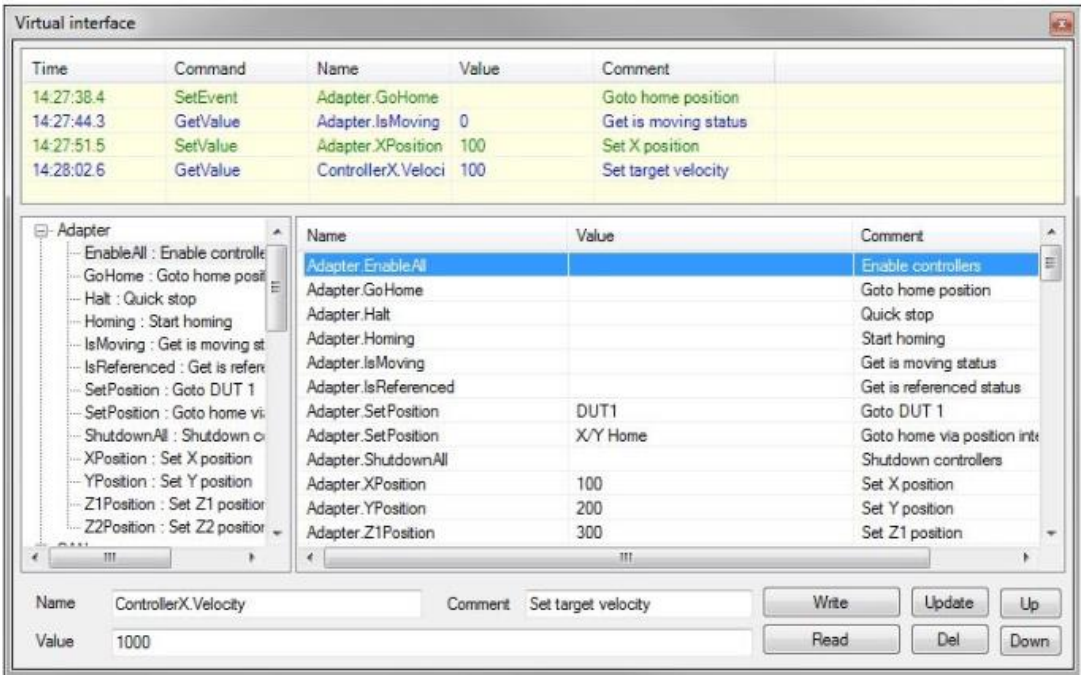


Abbildung 43: Oberfläche Virtual Interface

Alle möglichen Befehle für das *Virtual Interface* können durch einen Rechtsklick und die Auswahl „Add Defaults“ im virtuellen Interface angezeigt werden.

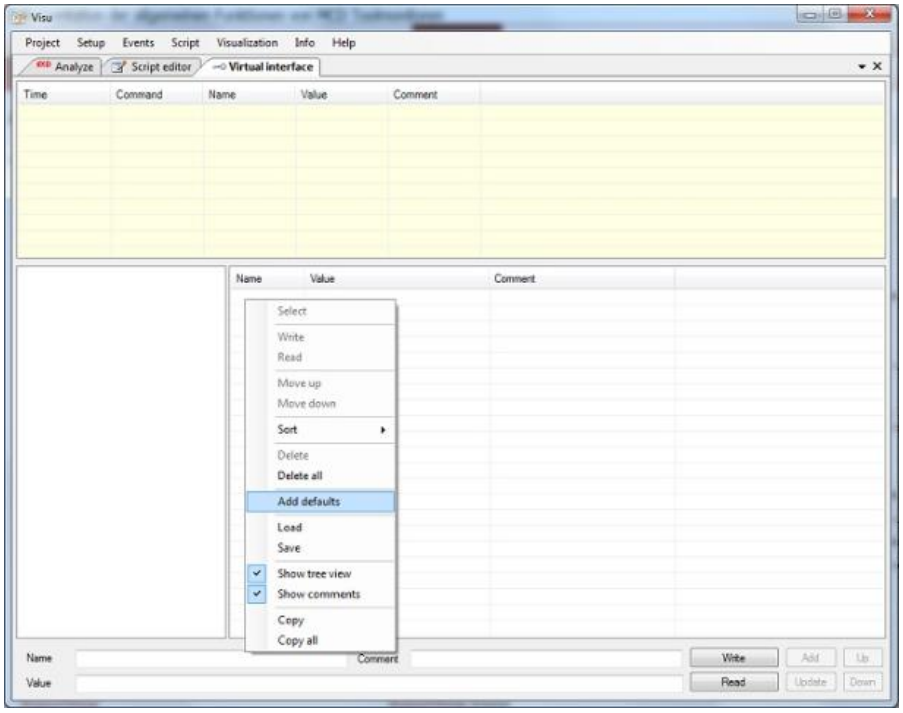


Abbildung 44: Befehle anzeigen

Bevor die Befehle ein weiteres Mal hinzugefügt werden, sollten sie über einen Rechtsklick auf „Delete All“ gelöscht werden, da sonst jeder Befehl in der Liste doppelt erscheint.

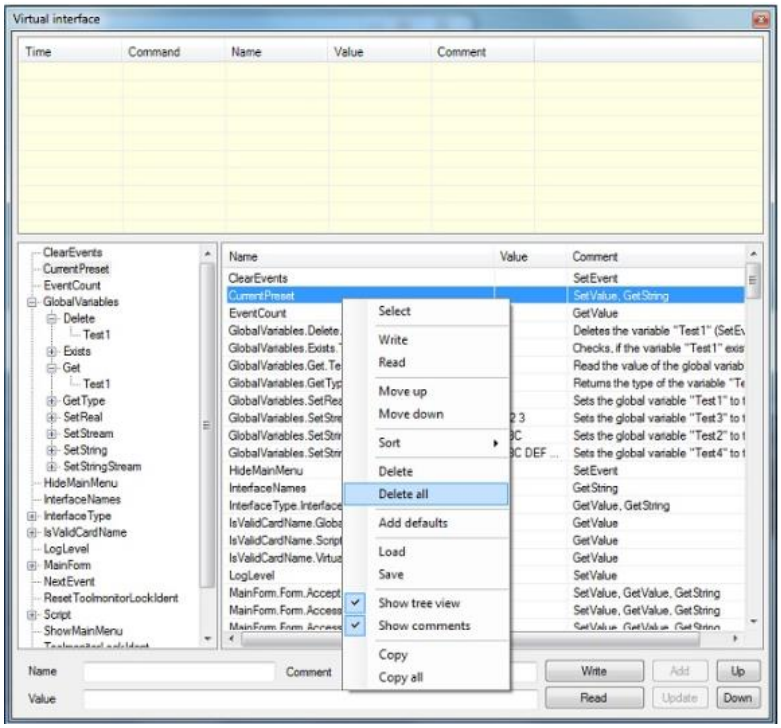


Abbildung 45: Alle Befehle löschen

Genauere Dokumentationen der vorhanden Befehle und Parameter sind in der Klassendokumentation der **General Help** und in der speziellen Hilfe zu finden.

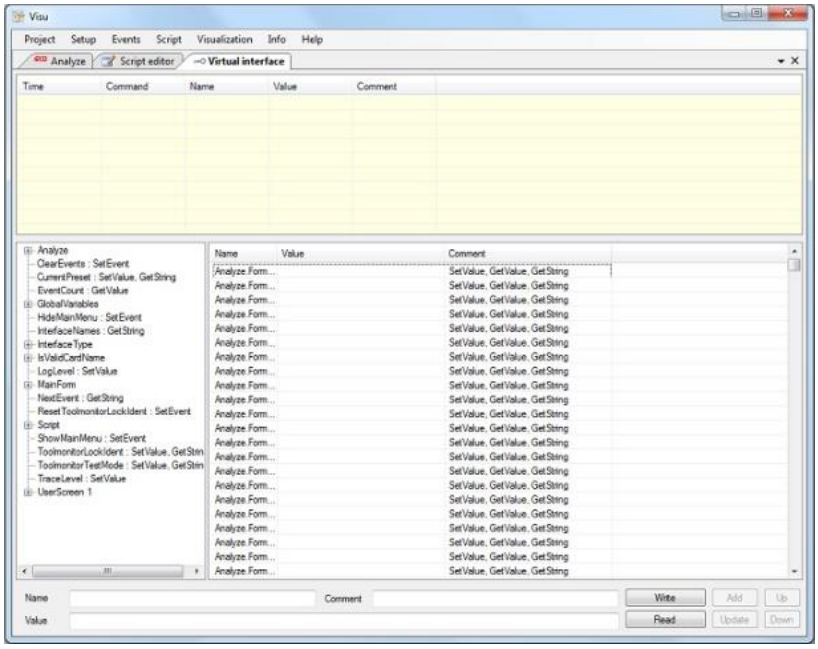


Abbildung 46: Alle verfügbaren Befehle für das virtuelle Interface

Bei einem Doppelklick auf ein Element (hier: *Global.Variables.SetReal.Test1*) werden dessen Eigenschaften in die Felder **Name, Value** und **Comment** (unten) übernommen. Im Beispiel hier wird der Wert der Variablen „Test 1“ auf 10 gesetzt. Über die Buttons **Read und Write** können die Befehle zum Lesen bzw. Schreiben abgeschickt werden. Alternativ dazu können die Befehle *Read* und *Write* auch über einen Rechtsklick erreicht werden. Die Rückgabewerte und Ergebnisse der Anweisungen erscheinen dann im Fenster am oberen Rand des Formulars.

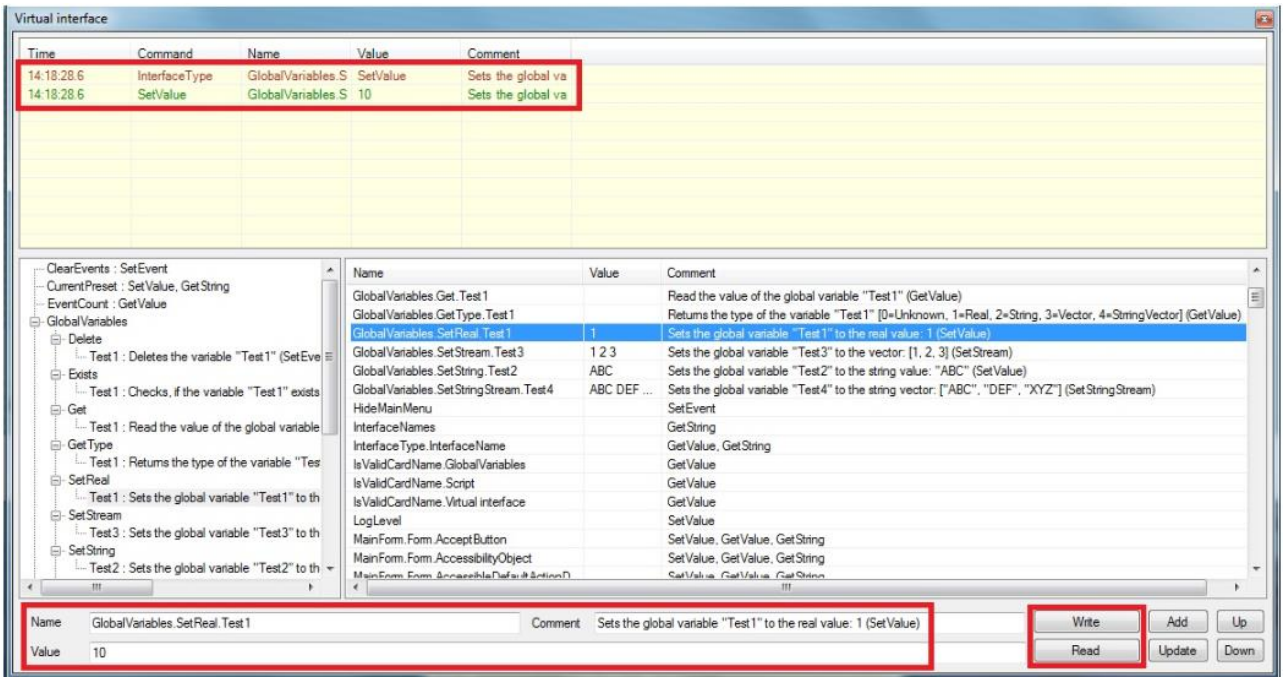


Abbildung 47: Befehle Read / Write

4.3. Report

Über den Menüpunkt **Report** kann je nach Toolmonitor und Einstellung im Setup ein anwenderspezifischer Report erstellt werden. Um diesen Menüpunkt zu aktivieren, müssen in den Optionen des Toolmonitors die gewünschte **Report View** - Anzahl erstellt werden. Das sind eigenständige Formulare, in denen über **FastReport** verschiedenste Reports gestaltet und erstellt werden können.

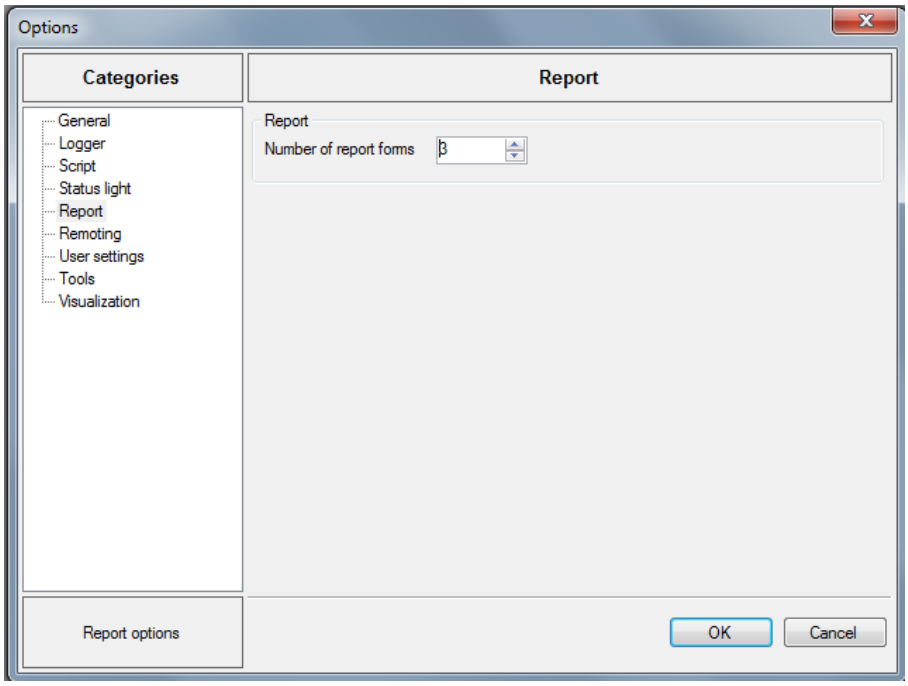


Abbildung 48: Anzahl der Reports verändern

Um ein **neues Report View** anzulegen, wählt man, wie hier beschrieben, in der Menüleiste unter **Setup → Options** den Reiter *Report* aus und stellt in der Auswahlbox die gewünschte Anzahl an Formulare ein. Für das untere Bild wurden drei Formulare angelegt. Nach dem Erstellen erscheint in der Menüleiste die zusätzliche Option *Reports*. Hier kann zwischen den erstellten Formularen ausgewählt werden.

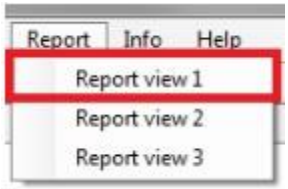


Abbildung 49: Report View öffnen

Nachdem der angelegte **Report View** geöffnet wurde, erscheint die leere Oberfläche mit der unten zu sehenden Menüleiste. In der oberen Reihe des Menüs stehen vier Buttons zur Verfügung: *Refresh, Load Report, Designer* und *Export.*

Über den **Refresh** Button lässt sich die *Vorschau des Reports* aktualisieren. Wenn beispielsweise eine Uhrzeit angezeigt wird oder eine Änderung im *Designer* vorgenommen wurde, kann diese so aktualisiert werden. Hierzu muss der *Designer* geschlossen sein, sonst erscheint eine Fehlermeldung am unteren Bildrand.

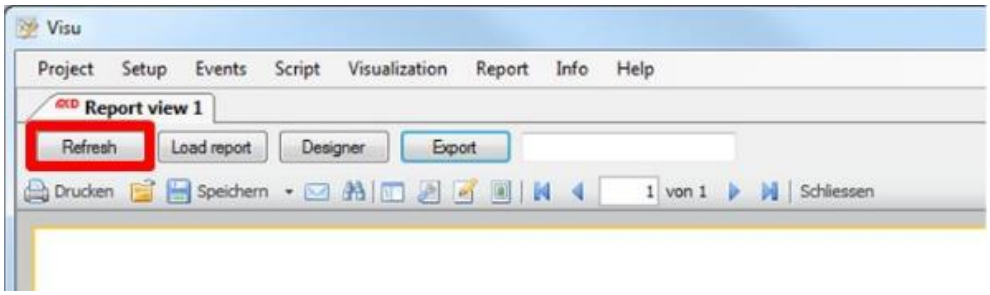


Abbildung 50: Refresh Button

Über den Button **Load Report** kann ein bereits vorhandener Bericht in die Oberfläche geladen werden. Während dieses Vorgangs muss der *Designer* geschlossen sein, da sonst eine Fehlermeldung am unteren Bildschirmrand erscheint.

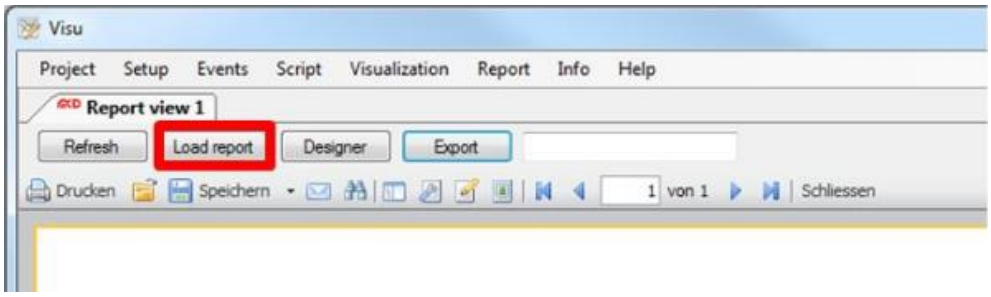


Abbildung 51: Load Report Button

Über den **Button Designer** öffnet sich der **FastReport Designer**. Über ihn können Layouts für neue Berichte erstellt werden oder vorhandene Berichte verändert bzw. bearbeitet werden. Eine detaillierte Beschreibung zum *Designer* und zu allen verfügbaren Komponenten finden man in dem *User's Manual von FastReport* unter https://www.fast-report.com/public_download/FRNetUserManual-de.pdf.

Abbildung 52: Designer Button

Über die Funktion **Export** kann der Bericht, wie im unteren Bild der „Beispielbericht.html“, direkt exportiert werden. Wird kein konkreter Pfad, sondern nur der Dateiname angegeben, wird der Export in dem Verzeichnis abgelegt, in dem sich der Toolmonitor befindet.

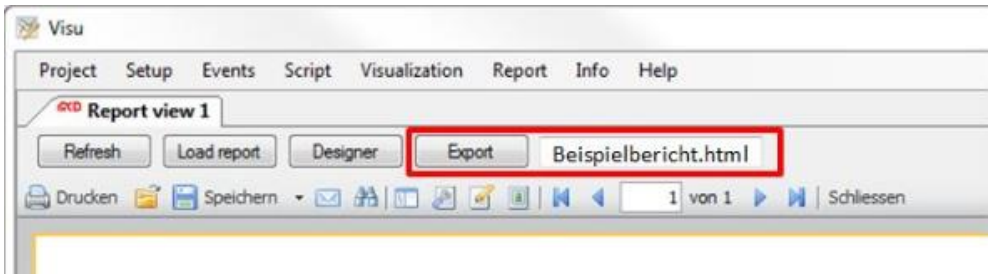


Abbildung 53: Export Button

Neben den vier beschriebenen Buttons stehen noch einige Symbole in der Symbolleiste zur Verfügung. Über das Symbol *Drucken* kann, wie unter Windows Anwendungen gewohnt, der Report ausgedruckt werden.

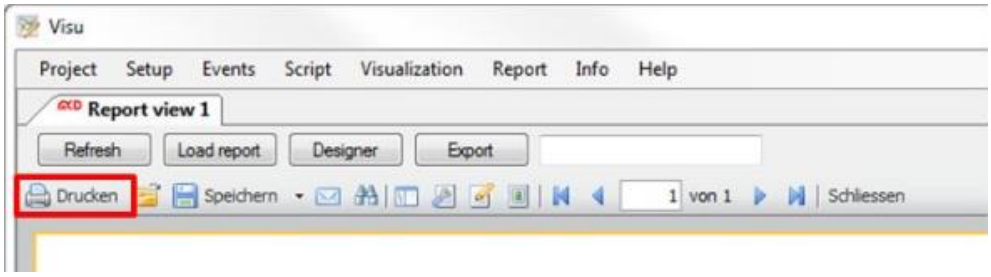


Abbildung 54: Report drucken

Über das Symbol *Öffnen* lassen sich bereits vorhandene Reports öffnen.

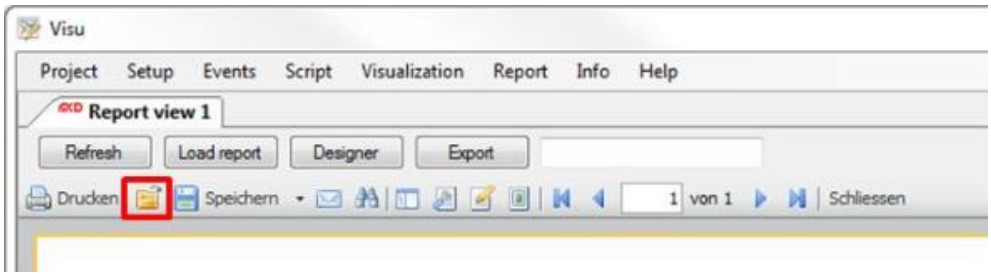


Abbildung 55: Report öffnen

Über das Symbol *Speichern* kann der erstellte Report in verschiedene Formate gespeichert werden. Zur Auswahl stehen: PDF, ODS, ODT, Excel, XML, RTF, HTML, text, CSV, BMP, Jpeg, Tiff und Gif.

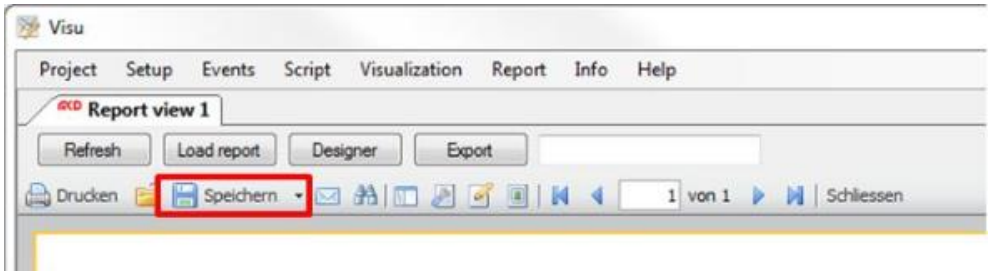


Abbildung 56: Reports speichern

Über das Symbol *Suche* kann der Report nach Schlagworten durchsucht werden.

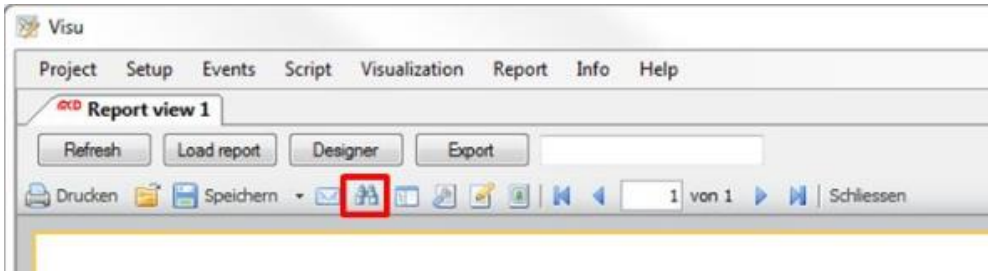


Abbildung 57: Report durchsuchen

Über das Symbol *Rahmen* kann für den Report ein Rahmen angezeigt werden.

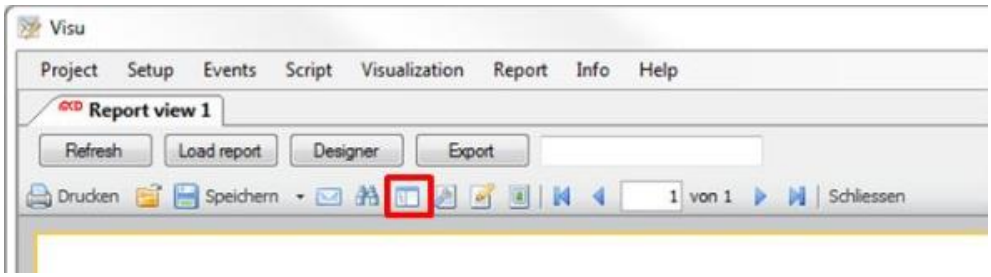


Abbildung 58: Rahmen ein - / ausblenden

Über das Symbol *Seite einrichten* kann z.B. das Papierformat oder die Maße für den Report angepasst werden.

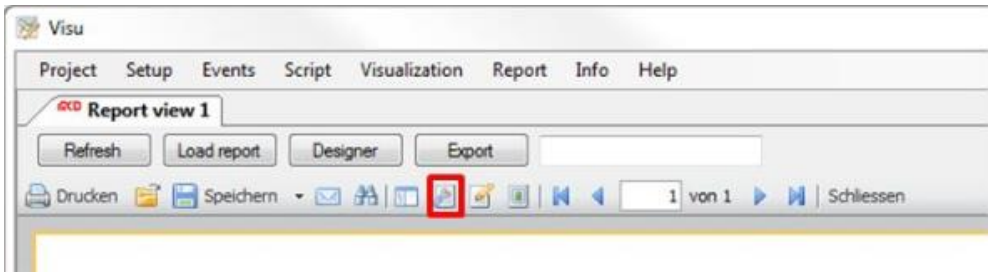


Abbildung 59: Seitenlayout bearbeiten

Über das Symbol *Bericht bearbeiten* gelangt man direkt in den *Designer* und kann dort den Report bearbeiten.

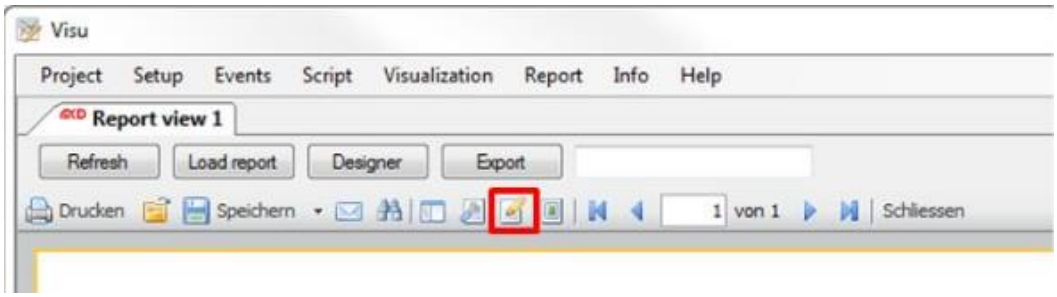


Abbildung 60: Report im Designer bearbeiten

Über das Symbol *Wasserzeichen* kann dem Report ein Bild oder Text als Wasserzeichen hinzugefügt werden.

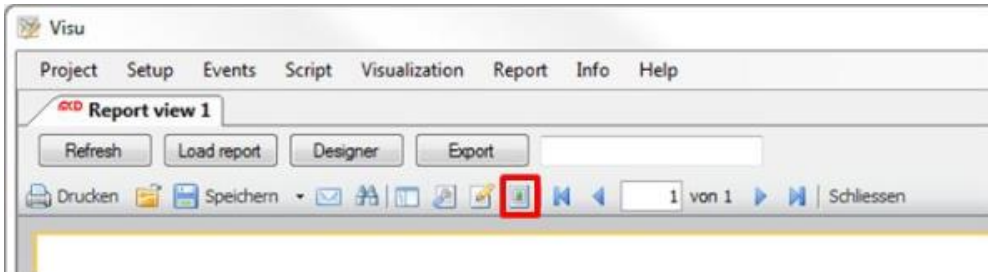


Abbildung 61: Wasserzeichen einfügen

In dieser Bedienleiste können die vorhandenen Seiten des Reports durchgeblättert werden.

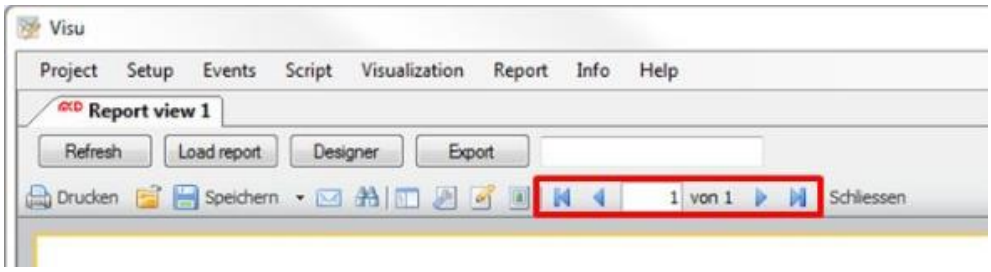


Abbildung 62: Andere Seite anzeigen

Über den Button *Schließen* lässt sich die Anzeige des *Report View - Formulars* schließen.

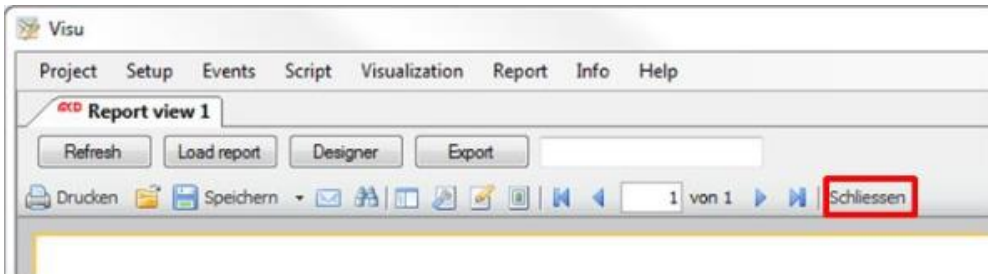


Abbildung 63: Report schließen

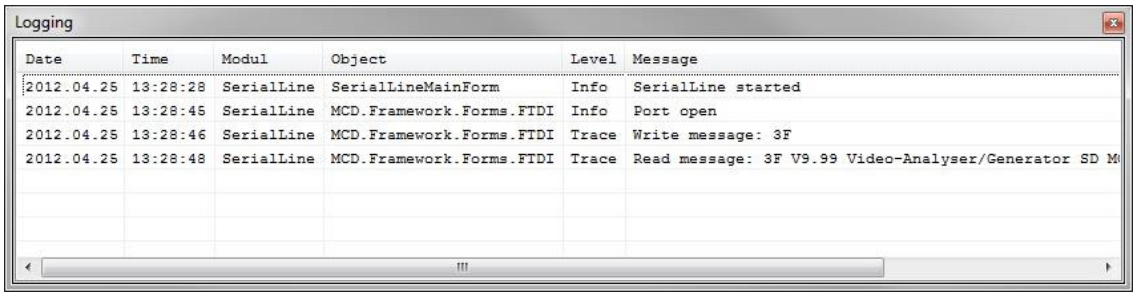
5. Events

Hierüber können die Anzeigen für die Log - und die Trace - Meldungen aufgerufen werden.

Abbildung 64: Events Menü

5.1. Logging

Über dieses Menü wird die Anzeige der Log - Meldungen für allgemeine Ereignisse, Warnungen, Fehler usw. aufgerufen.



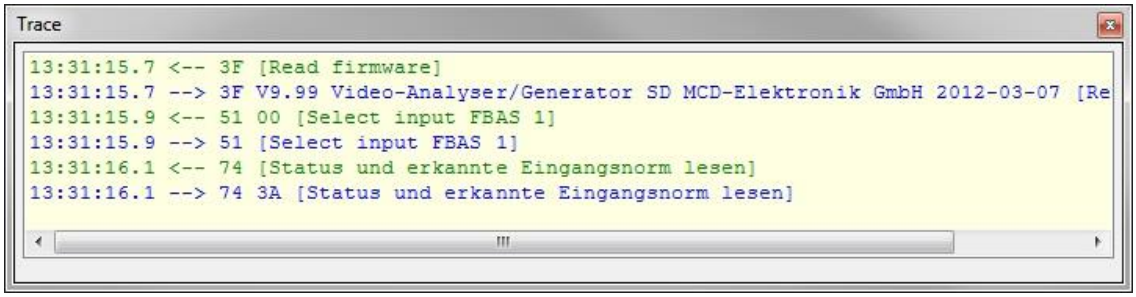
Logging

Date	Time	Modul	Object	Level	Message
2012.04.25	13:28:28	SerialLine	SerialLineMainForm	Info	SerialLine started
2012.04.25	13:28:45	SerialLine	MCD.Framework.Forms.FTDI	Info	Port open
2012.04.25	13:28:46	SerialLine	MCD.Framework.Forms.FTDI	Trace	Write message: 3F
2012.04.25	13:28:48	SerialLine	MCD.Framework.Forms.FTDI	Trace	Read message: 3F V9.99 Video-Analyser/Generator SD M

Abbildung 65: Log Monitor

5.2. Trace

Über dieses Menü wird die Anzeige der Trace - Meldungen (gesendete oder empfangene Nachrichten) aufgerufen.



Trace

```
13:31:15.7 <-- 3F [Read firmware]
13:31:15.7 --> 3F V9.99 Video-Analyser/Generator SD MCD-Elektronik GmbH 2012-03-07 [Re
13:31:15.9 <-- 51 00 [Select input FBAS 1]
13:31:15.9 --> 51 [Select input FBAS 1]
13:31:16.1 <-- 74 [Status und erkannte Eingangsnorm lesen]
13:31:16.1 --> 74 3A [Status und erkannte Eingangsnorm lesen]
```

Abbildung 66: Trace Monitor

6. Debugger

Mit dem **Debugger** kann man Scripte im MCD Toolmonitor debuggen. Der *Debugger* kann direkt aus dem Script heraus aufgerufen werden und vereinfacht die Erstellung Ihrer C# - Funktionen. In der Abbildung 67 ist die Benutzeroberfläche des *Debuggers* gezeigt.

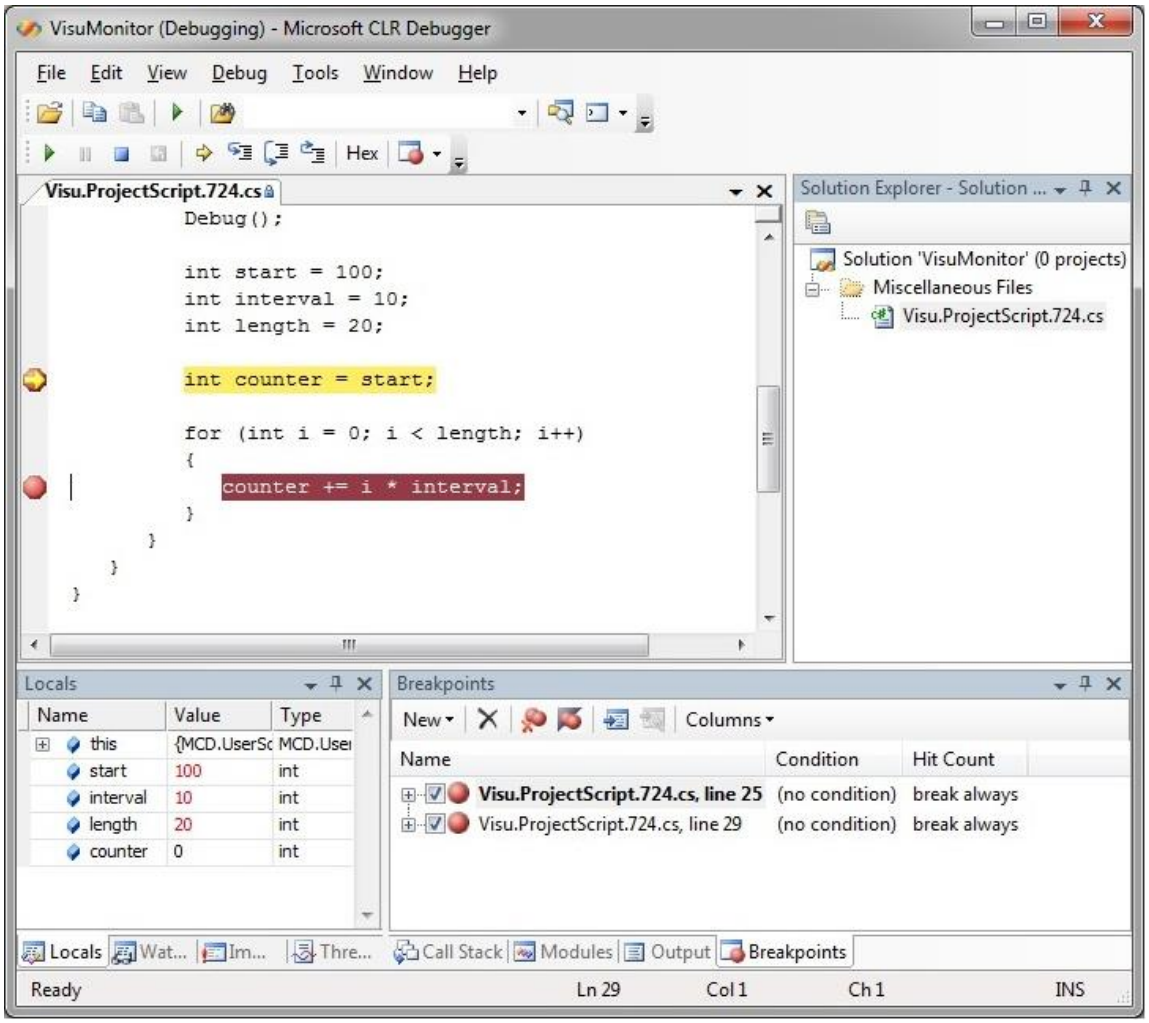


Abbildung 67: Benutzeroberfläche des Debuggers

6.1. Installation des Debuggers

Der *Debugger* muss auf Ihrem System installiert werden. Dazu führen Sie bitte den bereitgestellten Installer *Microsoft.Net SDK.exe* aus.

Sie können den benötigten Installer auch aus dem Internet herunterladen:
Download Microsoft.Net SDK 2.0

Suchen Sie bitte nach „*Microsoft.Net SDK 2.0*“. Nach der Installation können Sie den *Debugger* nutzen.

6.2. Starten des Debuggers

Zum Starten des *Debuggers* öffnen Sie bitte das *Kontextmenü* im *Script Editor* durch einen Rechtsklick. Im *Kontextmenü* können Sie dann über den Eintrag **Launch debugger** den *Debugge* öffnen.

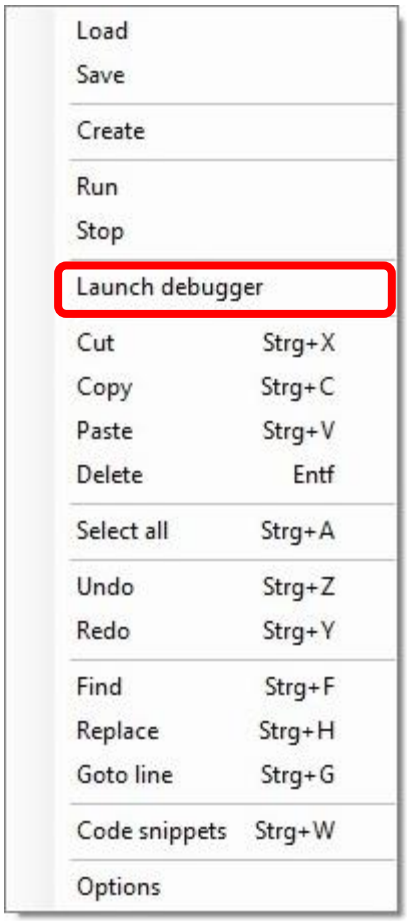


Abbildung 68: Debugger starten

Um den Ablauf ihres Scripts im *Debugger* anzuhalten gibt es zwei Möglichkeiten. Zum einen können Sie im *Debugger* einen **Breakpoint** setzen. Die Benutzung von *Breakpoints* wird im *Kapitel Breakpoints nutzen* beschrieben.

Eine weitere Möglichkeit ist beispielsweise, den Startpunkt für den *Debugger* im Script über den **Befehl Debug()** festzulegen (siehe Abbildung 69, Zeile 19). An dieser Stelle startet dann der *Debugger*. Über diesen Weg können Sie den *Debugger* in bestimmten Situationen starten, z.B. nach der Abfrage einer Bedingung in einer „if – Anweisung“.

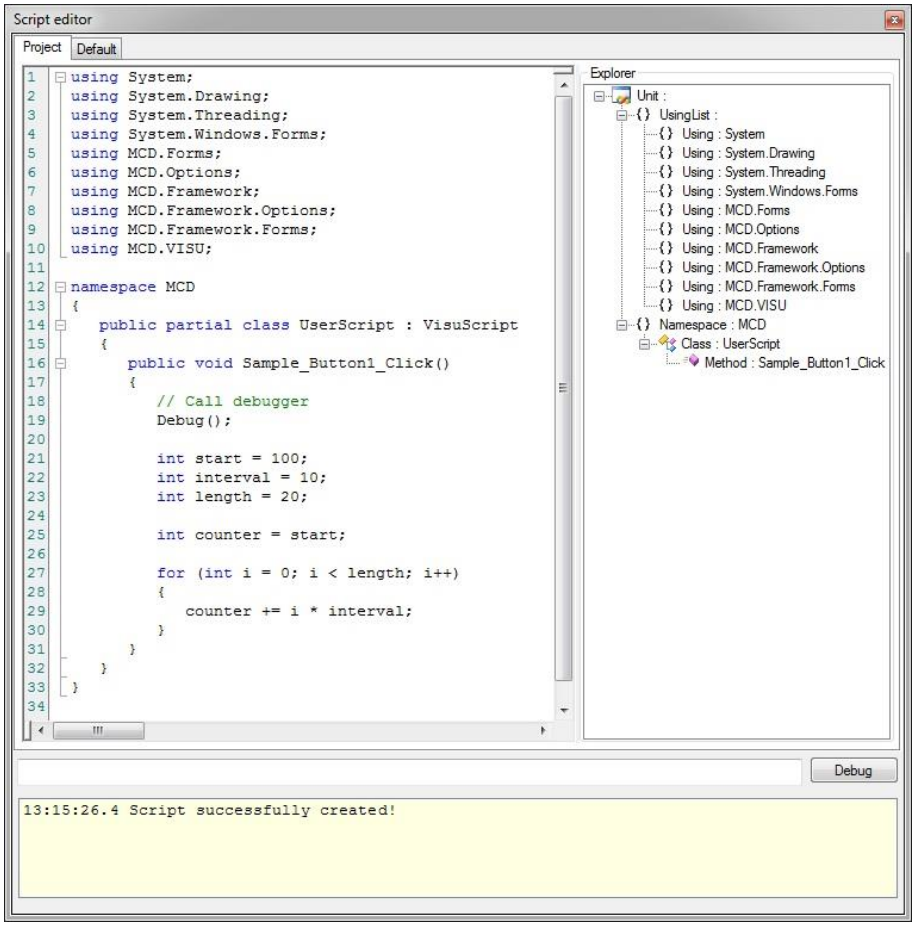


Abbildung 69: Debug - Befehl im Script Editor

Nun können Sie das eigentliche Debuggen Ihres Quellcodes starten. Dazu wechseln Sie bitte in den *Script Editor* bzw. Ihre *Visualization - Oberfläche* und starten den gewünschten Codeabschnitt durch Betätigen eines Controls oder durch Starten des Scripts über „Run“. Wechseln Sie nun wieder in den *Debugger*. Haben Sie zuvor einen *Breakpunkt* eingefügt, sehen Sie nun einen gelben Pfeil an dieser Stelle. Dieser signalisiert, dass die Programmabarbeitung hier gestoppt wurde.

```
            int start = 100;
            int interval = 10;
            int length = 20;

            int counter = start;

            for (int i = 0; i < length; i++)
            {
                counter += i * interval;
            }
        }
    }
}
```

Abbildung 70: Debugger mit Breakpoint stoppen

Haben Sie einen *Debug() - Befehl* eingefügt, wird das Debuggen an dieser Stelle gestartet, dieses wird durch einen grünen Pfeil markiert.

```
namespace MCD
{
    public partial class UserScript : VisuScript
    {
        public void Sample_Button1_Click()
        {
            // Call debugger
            Debug();

            int start = 100;
            int interval = 10;
            int length = 20;

            int counter = start;

            for (int i = 0; i < length; i++)
            {
                counter += i * interval;
            }
        }
    }
}
```

Abbildung 71: Debugging am Startpunkt beginnen

6.3. Script debuggen

Abbildung 72: Toolbar des Debuggers

Die Navigation des *Debuggers* können Sie am einfachsten über die **Toolbar Debug** (siehe Abbildung) durchführen. In folgender Tabelle finden Sie die wichtigsten Funktionen im Überblick:

	Weiter - Starten des Programmablaufs bis zum Ende oder zum nächsten Breakpoint
	Anhalten - Stoppen des Debuggers
	Einzelschritt - Der Befehlszeiger des Debuggers wird auf jeden nächsten Befehl gesetzt (auch in Unterfunktionen)
	Prozedurschritt - Der Befehlszeiger des Debuggers wird auf den nächsten Befehl gesetzt (Funktionen werden in einem Schritt abgearbeitet)
	Rücksprung - Der Befehlszeiger springt zurück zum Funktionsaufruf
Hex	Hexadezimale Anzeige - Die Darstellung für die Variablenwerte kann zwischen dezimal und hexadezimal umgeschaltet werden

Abbildung 73: Funktionsüberblick der Toolbar

6.4. Breakpoints nutzen

Um die Ausführung des Scripts an einer bestimmten Stelle zu stoppen können **Breakpoints** eingesetzt werden. Ein *Breakpoint* kann durch einen Mausklick auf die graue Spalte vor dem gewünschten Befehl aktiviert werden. Zum Deaktivieren des *Breakpoints* klicken Sie diesen einfach noch einmal an.

Haben Sie den *Debugger* gestartet und einen *Breakpoint* gesetzt, können Sie über die Schaltfläche *Weiter* die Ausführung aller Befehle bis zu dem *Breakpoint* aktivieren. Wie in der Abbildung zu sehen, bleibt der *Debugger* mit einem gelben Pfeil an dem gewünschten Befehl stehen. An dieser Stelle können Sie dann die Werte der Variablen analysieren und z.B. im *Einzelschritt - Modus* weitere Befehle debuggen.

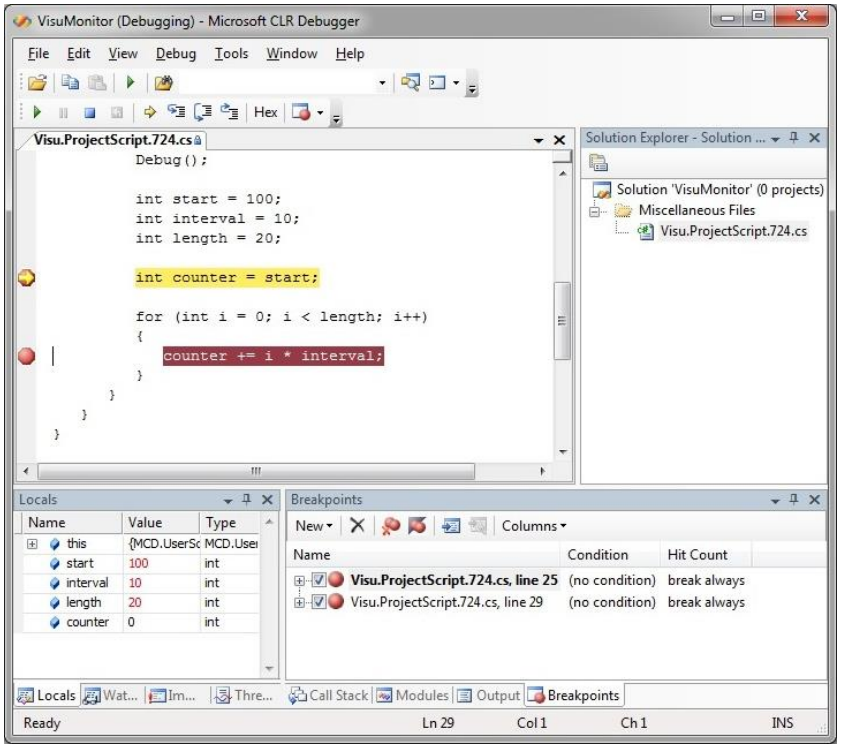


Abbildung 74: Debugger mit Breakpoint stoppen

6.5. Variablen überwachen

Der *Debugger* bietet die Möglichkeit, Variablenwerte anzuzeigen und zu verändern. Der einfachste Weg einen Variablenwert anzuzeigen ist es, den Mauszeiger über den gewünschten Variablennamen zu bewegen. Es wird dann automatisch der aktuelle Wert angezeigt.

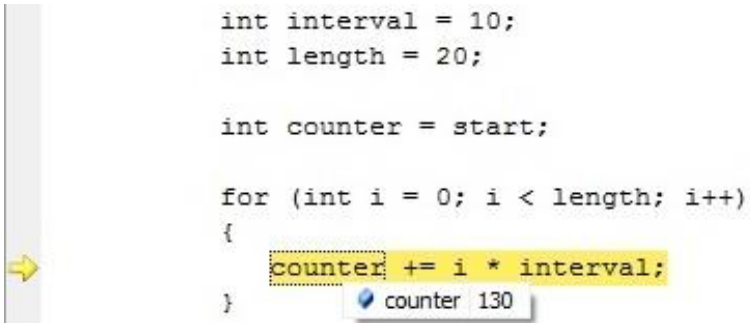


Abbildung 75: Variablenwert mittels Mauszeiger anzeigen

Des Weiteren können Sie im **Fenster Local** die aktuell gültigen Variablen ansehen. Im **Fenster Watch** können Sie außerdem eigene Variablen zur Überwachung hinzufügen. Dazu geben Sie einfach den Variablennamen in der **Spalte Name** ein.

Um den Wert einer Variablen zu verändern, können Sie den Eintrag in der **Spalte Value** auf den gewünschten Wert anpassen. Hierbei ist darauf zu achten, dass dieses ein manueller Eingriff in den Softwareablauf ist, der später nicht mehr im Quellcode zur Verfügung steht. Dieser Eingriff sollte daher nur zu Debug - Zwecken genutzt werden.

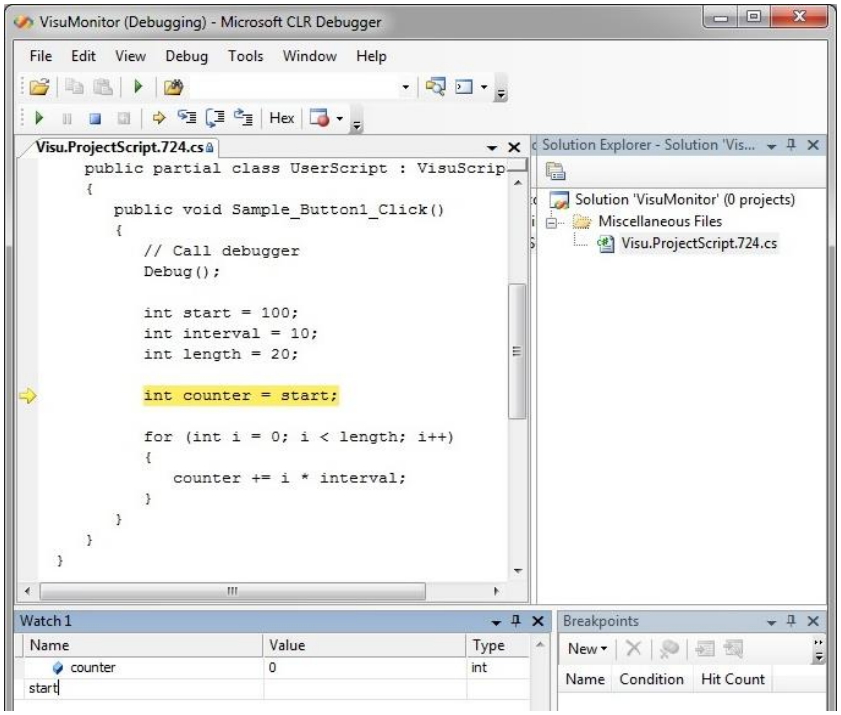


Abbildung 76: Überwachung von Variablen