Manual

Toolmonitor Visu









Softline

Modline —

Conline —

Boardline -----

Avidline -

Pixline —

Application -

MCD Elektronik GmbH

Table of Contents

1.	(GENER	RAL	4
2.	9	SOFTV	VARE AND DRIVER INSTALLATION	4
	2.1	L. Reg	QUIREMENTS	4
	2.2	2. Sta	ARTING THE INSTALLED SOFTWARE	4
3.	I	INTRO	DUCTION INTO OPERATION	5
	3.1	L. Bri	IEF INTRODUCTION	5
Λ		COETIA		7
4.	, / 1			,, 7
	4.1	4 1 1	Onenina Desianer	7
		412	Creating a New Form	7
		4.1.3.	Settinas and Parameters	8
		4.1.4.	Adjusting the Menu Structure	9
		4.1.5.	Properties and Events	10
		4.1.6.	Controls	11
	4.2	2. Scf	RIPT	13
		4.2.1.	Script Editor	13
		4.2.1.1	1. Base Script	14
		4.2.1.2	2. Explorer	15
		4.2.1.3	3. Context Menu	16
		4.2.1.4	4. Status Display	18
		4.2.1.5	5. Text Editor	19
		4.2.2.	Run Command	23
		4.2.3.	Stop Command	23
		4.2.4.	Integrating DLLs	24
	4	4.2.5.	Global Variables	25
		4.2.6.	Virtual Interface	25
	4.3	B. Ref	PORT	29
5.	I	EVENT	rs	34
	5.1	L. Loo	GGING	34
	5.2	2. TRA	ACE	34
6.	I	DEBUG	GGER	35

6.1.	DEBUGGER INSTALLATION	35
6.2.	STARTING THE DEBUGGER	36
6.3.	DEBUGGING THE SCRIPT	39
6.4.	Using Breakpoints	39
6.5.	Monitoring Variables	40

1. General

The "Visualization Toolmonitor", also referred to "Toolmonitor Visu", offers an easy and fast way to generate operating and user interfaces in connection with the MCD TestManager. No external software is required. The interfaces created using the *Designer* can be linked to own procedures and functions via the script editor.

Order number: # 122427

2. Software and Driver Installation

2.1. Requirements

- Windows (Windows XP[®] Windows 8.1[®], 32 or 64 bit)
- .Net Framework 3.0

Copy the VisuMonitor.exe into a user - defined directory on the target system to install the MCD Toolmonitor Visu.

2.2. Starting the Installed Software

The software can be started by executing *VisuMonitor.exe*.

3. Introduction into Operation

3.1. Brief Introduction

On first starting of the Toolmonitor Visu, the still empty interface of the monitor appears.

👳 Visu								1 23
Project	Setup	Events	Script	Visualization	Info	Help		

Figure 1: Default Interface

To obtain an overview of the functions and possibilities of Toolmonitor Visu, a sample preset can be loaded via **Project** \rightarrow **Presets** \rightarrow **Sample**, which offers an overview of all monitor functions.

Project	Setup Events	s Script	
Sa Sa Co Re P	ve S ve as bad istore inport window posit	itrg+S	
Pr	esets	•	Default
X CI	ose all		Sample
V Lo	Local (unlock)		Empty
Re	emote (lock)		Reset window positions
Hi	de		
-			

Figure 2: Loading a Sample Preset

Buttons	MCDCONTOR	Text input	Button1
Standard button	Color button	Single line	Button2
Rat button	Picture button	Mulit line text box	Button 3
Direct Check boxes	access button	Rich text box	
CheckBox 1 CheckBox 2 CheckBox 3	Line A Line B Line C Line D Line E		
Combo- & List boxes		Numeric input Timer	
Item A Item A Item B Item C	Item A Item B Item C Item D Item E	0.000 (\$) (\$)	
item A 👻	Item F	0	

Figure 3: Sample Preset Interface

The Script Editor for the generation of own functions and procedures can be reached via the Script \rightarrow Script Editor menu item.



Figure 4: Open Script Editor

4. Software Manual

4.1. Designer

Custom operating and user interfaces can be created in the **Designer** of Toolmonitor Visu. Standard components and additional MCD controls are available for this purpose.

4.1.1. Opening Designer

To open the *Designer* of the previously created project, the desired project (here "Sample") is selected under the **Visualization** \rightarrow **Designer** menu item.

Visualization	Report	Info	Help
Sample			
Designer	•	Sam	ple

Figure 5: Opening Designer

4.1.2. Creating a New Form

Operate the **New** button in the **Form** menu to create a new form in the *Designer*. It is also possible to import an already existing interface or to export the newly created interface using this menu. The button **Clear** deletes the current interface.



Figure 6: Creating a New Form

After a new form is created, a name can be assigned. This **Form Title** is inquired automatically and required for creating the new form.

ter form title	
Form title	
Ok	Cancel

Figure 7: Naming the form

4.1.3. Settings and Parameters

A list with the settings and parameters of the selected object can be found on the right side in the *Designer*. These configurable parameters are also referred to as **Properties** and are different for every form and control. The exact size and position can be defined for most objects. In the case of analog displays, the minimum and maximum value and the display scaling can be configured as well.

To change the Properties of a control / form, it must be marked in the design window (center window).

In the example below, Button1 (left red frame) was selected. Thus, the color (upper red frame) or the text of the button (lower red frame) can be changed in the right list.

Designer		×
Form Edit Format		
Toolbox		
MCD Controls		
Frames	BackColor Control	
Windows Forms	Backgroundimage (Neine)	
<pointer></pointer>	BackgroundImageLayout Tile	E
ab Button Textbox	Cursor Default	
CheckBox	HatAppearance	100
CheckedListBox	The Style Standard Microsoft Sans Serif: 8.25r	ot
ComboBox = Q Button1 Q	ForeColor ControlText	~
DataGridView	Image (Keine)	
DateTimePicker	ImageAlign MiddleCenter	
DomainUpDown	ImageIndex (Keine)	
GroupBox	ImageKey (Keine)	
 ▲ B HScrollBar 	ImageList (Keine)	
A Label	Text Button 1	
E ListBox	rexualion widolecenter	
#- MaskedTextBox	TextImageRelation Overlay	-
😹 MediaPlayer	Text	
MonthCalendar	Der dem Steuerelement zugeordnete Text.	
1 NumericUpDown -		
		_
GetValue("Form1.Button1.UseCompatibleTextRendering")		-
GetValue("Form1.Button1.UseMnemonic")		
GetValue("Form1.Button1.UseWaitCursor")		
GetValue("Form1.Button1.Visible")		
GetValue("Forml.Buttonl.Width")		
Remote events:		
public void Form1_Button1_Click()		
public void Form1_Button1_DoubleClick()		T

Figure 8: Properties of a Button

4.1.4. Adjusting the Menu Structure

The structure of the **Menu Tree** can also be adjusted about the properties of the form. After the desired form was selected in the *Designer*, the *Menu Tree* property is available on the right side under *Properties*. Here it can be defined, under which sub - group the interface should be displayed in the Toolmonitor.



Figure 9: Menu Tree

In this example, the form is called "Test". It contains an analog voltage display. The structure of sub menu, in which the "Test" interface is displayed, can be seen in Figure 10. "Visualization" is at first position, and thus the menu item to be displayed in the top bar of the Toolmonitor. "Voltage" is a sub - item in the "Visualization" menu. The next point would then be a sub - item of "Voltage".

ichenfolgen-Editor	8 ×
Zeichenfolgen in der Auflistung eingeber	n (eine pro Zeile):
Visualization Voltage	*
1	4
	OK Abbrechen

Figure 10: Structure of the Sub Menu

Any number of sub items can be created according to this scheme. However, this does not always ensure clarity. In our example, the "Test interface" can be seen under "Voltage", as can be seen in Figure 11.

roject Setup Events Script	Visualization	Info Hel	p
ftp test	Voltage	•	Test
	UserScreen	12	
27 36 45 54 63	Designer	•	
mV	·		

Figure 11: Sub Menus in the Toolmonitor

4.1.5. Properties and Events

A list of the changeable *Properties* and **Events** of a control can be found in the lower *Designer* window. For example, the **Remote Events** of a button can be found on the first figure. The *Events* and changeable *Properties* are different for every control. The respective *Remote Events* are listed under **Controls**. *Properties* of a control can be read out using the **GetValue** function in the **Script Editor**. They can be changed using the **SetValue** function.

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

Figure 12: GetValue and SetValue Functions

Reactions to changes or interactions with a control or form can be designed using *Events*.



Figure 13: Displaying Event

A double - click on the desired *Event* switches from *Designer* to **Script Editor**, where the reactions to the respective *Event* can be programmed.

```
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_SizeChanged()
public void example_Button1_TextChanged()
```

Figure 14: Properties and Events

4.1.6. Controls

The list of usable Controls can be found on the left side of Designer.

^{KD} Designer	
Form Edit	
Toolbox	
MCD Controls	
<pointer></pointer>	~
🔨 AnalogDisplay	1
42 AnalogInputControl	
42 AnalogOutputControl	
AnalogValueDisplay	
AutoSizeLed	
BitMaskControl	
CIE1931Display	
🎦 DataDisplay	
bigitalIoControl	
DutDisplay	
📕 DutDisplayPanel	
🔵 Led	
ab MCDButton	8
MCDCheckBox	
MCDCheckedListBox	
MCDComboBox	
MCDDataGrid	
MCDHScrollBar	
MCDListBox	
MCDNumericUpDown	
MCDPanel	
MCDTablePanel	
 MCDRadioButton 	
abl MCDTextBox	
MCDTimer	
MCDTrackBar	
MCDVScrollBar	
MCDDockPanel	T
Windows Forms	
ActiveX	
User Controls	

Figure 15: List of available Controls

In addition to the standard **Windows Controls**, additional MCD related controls are available unter the **MCD Controls** tab. These *MCD Controls* contain controls, such as the **CurveAnalysis Control**, which are not included in the Toolmonitor by default, but can be activated via license. The difference between a standard *Windows Control* and a *MCD Control* can be seen in the following example. For example, if a form with a MCD button is controlled via the TestManager, hence **Remote**, the button is automatically locked on the interface and cannot be manually operated. However, a Windows button cannot be locked and can be further manually operated (figure on the right). Thus, the desired type can be selected for every application case. To test this behavior in the Toolmonitor, the view can be switched between **Local and Remote** via the Project \rightarrow **Local / Remote** menu item (figure on the left).

💓 Vis	su		Visu 🗆 🛛 🕱
Proj	ject Setup Events Script	Visualization Info	Project Setup Events Script Visualization Info
	Save Strg+S Save as Load Restore Import window positions	• X	Button 1
1000	Presets +		MCDButton 1
×	Close all		
4	Local (unlock)		
	Remote (lock)		
	Hide		
\blacksquare	Exit		

Figure 16: Windows Controls and MCD Controls

4.2. Script

4.2.1. Script Editor

It is possible to create custom procedures and functions in the **Script Editor** to expand the Toolmonitor. Programming uses the C# programming language of the .Net framework. Available .Net assemblies can be used in the scripts.

The *Script Editor* mainly exists of three areas. The **Text Editor**, the **Explorer** and the **Status and Debug Display**. Most settings and additional functions can be reached via the context menu of the *Script Editor* and are described in the following.

🧏 Visu	
Project Setup Events Script Visualization Info Help	
Script editor Global variables	~ X
Base script ReflectionWindow	▼ X
<pre>// Toolmonitor base script // Toolmonitor base script // Toolmonitor base script is a global script for all // toolmonitors and instances of this toolmonitor // type at the same location (path) and // non-project related! // using System; lusing System.Threading; lusing System.Threading; lusing MCD.Forms; lusing MCD.Framework; lusing MCD.Framework.Potions; lusing MCD.Framework.Controls; lusing MCD.Framework.Controls; lusing MCD.VISU; lusin</pre>	Project explore Script explore III +
	← 📔 Run
Time Message	
12:28:32.0 Script successfully compiled! 12:28:32.0 Script successfully created!	

Figure 17: Script Editor Interface

4.2.1.1. Base Script

The so - called **Base Script** is a global script for all Toolmonitors and their instances. The *Base Script* is the standard script that opens on first opening of *Script Editor*. Any other created scripts and / or tabs can access the *Base Script*.

In the lower example, a "Testfunktion()" function was created in the *Base Script*, which is used to call a "Testfunktiontab()" function in the "tab1" script. In the "tab1" script, it is possible to access the variables (here "Testwert") of the "Base Script" as can be seen in the example below.



Figure 18: Function Call in the Base Script

The *Base Script* of a Toolmonitor is the only script that is saved as separate C# file (.cs). All other scripts, settings, and contents of the project are saved as .xml in the project file.

Thus, if a project with several scripts is moved to a different PC, the project file and the corresponding *Base Script* must be moved. The advantage of this structure is that several scripts and several different Toolmonitors can access the content / functions of the *Base Script* at the same time.

4.2.1.2. Explorer

The **Script Explorer and / or Project Explorer** can be found on the right side of the Toolmonitor interface. Here, the inserted **Usings**, as well as the existing variables and functions are displayed hierarchically (**Script Explorer**) and / or the different scripts are listed (**Project Explorer**). When double - clicking on an element, the cursor in the text editor jumps automatically to the respective position in the source text.



Figure 19: Script Explorer

The so - called **Script Explorer** displays all namespaces with their respective classes, all functions, and all created variables hierarchically. When double - clicking on one of the displayed elements, the cursor in the text editor jumps automatically to the respective position in the source code.



Figure 20: Displaying Script Explorer

The so - called **Project Explorer** contains an overview of all available scripts. Double - clicking on a script opens the respective script automatically.

Project explorer		Ψ×
🖃 🧓 Scripts		
Base script		
- Analyze		
UserScreen	Controls	
	C	1
Project explorer	Script explorer	

Figure 21: Displaying the Project Explorer

4.2.1.3. Context Menu

The different options and settings of the *Script Editor* can be reached via the **Context Menu**. For this purpose, the right mouse button is clicked within the editor. After clicking, the *Context Menu*, shown below, opens. An overview of the individual functions of the *Context Menu items* can be found in the table below. Some of these functions can also be reached via the icons in the top menu bar of the text editor.

Base script Reflect	ionWindow			
// Toolmonitor // // Please note	base script : This script is a	global	scr	ipt for all
11	Import			of this toolmonito
11	Export			partition and
//	Create		-	
0 1 Using System 2 Using System	Run Stop			
3 using System 4 using System 5 using MCD.Fo 6 using MCD.Op	Create with debug infor Launch debugger Open reflection window	mation		
7 using MCD.Fr 8 using MCD.Fr 9 using MCD.Fr 10 using MCD.Fr 11 using MCD.VI	Cut Copy Paste Delete	Strg+X Strg+C Strg+V Entf		
4 {	Select all	Strg+A		
25 🖻 public pa 26 (27	Undo Redo	Strg+Z Strg+Y	c	ript
8 -) 9 -)	Find Replace	Strg+F Strg+H		
	Goto line	Strg+G		
	Code snippets Format source code	Strg+W Strg+S		
	Toggle region outlines	Strg+O		
	Update user screen con	trols		
	Add project tabulator Delete project tabulator			
	Print			
	Options			
	Import all		-	

Figure 22: Context Menu of the Script Editor

The following table shows the Context Menu items, the corresponding functions, and, if available, the corresponding icons in the toolbar.

	Context menu	Description
	Import	Imports an existing script
	Export	The created script can be stored externally
401 01	Create	Creates the script
	Run	Runs the script
	Stop	Stops a running script
	Create with debug information	The script is created with debug information
	Launch Debugger	Launches the debugger
	Open reflection window	The overview window for the project opens. Right click for refresh
÷	Cut	The selected text/code is cut out
1	Сору	The selected text/code is copied to the clipboard
	Paste	Text/code is inserted from the clipboard
	Delete	The selected text/code is deleted
	Select all	The entire content of the script is selected
6	Undo	Undo the last step
(2)	Redo	Repeat the last undone action
C.	Find	Opens the search window
58	Replace	Open the search and replace window
	Goto line	Jumps to the entered line
	Code snippet	Here you will find often needed, ready-made pieces of code
	Format source code	The code is automatically formatted
	Toggle region outlines	#regions are collapsed / expanded
	Update user screen controls	Updates the created user interfaces
	Add project tabulator	Adds a new tab (tab) to the project
×	Delete project tabulator	The currently opened tab will be deleted
3	Print	Prints the script
:	Options	Opens the syntax setting of the Script Editor
	Import all	Imports all scripts from an MCD Toolmonitor XML file

Figure 23: Function Overview of the Context Menu

4.2.1.4. Status Display

The third area with the **Status and Debug Display** is located at the bottom in the *Script Editor* window.



Figure 24: Status Display

This area allows debugging individual functions and commands via the *Debug line* and button. Furthermore, the bottom window also displays status messages and *Debug outputs*.

To call a desired function, the name of the function is entered into the *Debug line* and debugging is started with *Debug button* on the right. A *Debug command* is available to display variable values for test and verification purposes. In the example below, the "**Main() method**" is called and the values of variables Testvalue 0 to Testvalue 5 are outputted in the *Debug window*.

The entered Debug commands are saved in the Registry and can be selected again for another call.

13	🕂 na	mes	space MCD		Pro
.4	{				l e
.5	白	pu	ublic partial class UserScript : VisuScript		l d
6		{			Шê
.7	白		<pre>public override void Main()</pre>		ll e
8			{		
9			for (int i = 0; i < 5; i++)) I MICE
20			{		¥
1			<pre>Debug("TestValue", i, "has the value:", i);</pre>		<u> </u>
22			}	E	
23					l∥ĕ
24			}		d
25		}			-
26	}				
27				-	
ne l	_				
٩				P	
ía:	in()		•	Debuc	
Tir	ne		Message		
09	:28:06.	2	TestValue 0 has the value: 0		
09	:28:06.	2	TestValue 1 has the value: 1		
09	:28:06.	2	TestValue 2 has the value: 2		
09	:28:06.	2	TestValue 3 has the value: 3		
09	:28:06.	2	TestValue 4 has the value: 4		
	L3 L4 L5 L6 L7 L8 20 21 22 23 24 25 26 27 26 27 26 27 26 27 26 27 26 27 09 09 09 09 09	13 P na 44 { 55 P 16 17 P 10 17 P 10 10 17 P 10 10 17 P 10 10 10	13 □ name: 13 □ name: 14 { 15 □ put 15 □ put 16 { 17 □ 18 19 20 21 22 23 24 27 24 25 } 27 24 25 } 27 27 27 27 27 27 27 27 27 27	<pre>Image Space MCD Image Spa</pre>	<pre>Image Space MCD Image Message I</pre>

Figure 25: Debug Output of the Main() Method

4.2.1.5. Text Editor

The actual **Text Editor** for source text creation can be found in the left window of the *Script Editor*.



Figure 26: Text Editor Interface

If you start with an empty script and do not want to enter the required U*sings* manually, any character can be written into the empty text field. After clicking on the **Create** command of the *Context Menu*, the required usings are inserted automatically.



Figure 27: Entering Usings into the Empty Script

The *Text Editor* offers different possibilities for automatic code completion (*Intelli - Sense*). This way, properties and functions can be displayed for existing variables. If you enter a period after a variable, the available functions and properties are displayed automatically. In the figure below, the properties of a *double variable* are called to convert it into a *string* (the ".ToString()" method is used for this purpose).



Figure 28: Code Completion (variable. / function.)

Furthermore, **operator this.** and a *Context Menu* can be used to call the available functions. The example shows the call of the *SetEvent()* function. For this purpose, simply enter text "this.". After the period is entered, a window opens automatically, displaying the functions and their parameters. Select the desired function via double - click or enter.



Figure 29: Code Completion (.this)

If you have entered the correct function name, the possible parameters are displayed when you enter the open parenthesis. If the used function is overloaded, you can select using the arrows to display the different parameters.



Figure 30: Displaying Parameters

Detailed help regarding programming - specific questions is available via the integrated Microsoft online offering (MSDN). Keyboard shortcut "Ctrl + F1" opens "MSDN" automatically in your internet browser. You can mark a keyword and / or place the cursor into the word to call up the "MSDN help" for this keyword via "Ctrl + F1". In the figure, the document for keyword "double" has been called up.



Figure 31: MSDN Online Help (Ctrl + F1)

The source text presentation can be adjusted via the item **Options** in the *Context Menu*. For example, the colors for certain keywords, strings or numbers can be defined here. Furthermore, the number of spaces for a tab or the division of the editor window can be adjusted as well. The figure shows the main page of the *Options*.

Syntax Settings			
 Options General Fonts and Colors Additional Keyboard 	Document Word Wrap Highlight Urls Vrag and drop text	✓ Vertical scroll bar ✓ Horizontal scroll bar ✓ Forced scroll bars	White space
	Gutter&Margin	Gutter width: Margin position:	16 70
	Line Numbers Une Numbers Image: Show Line Numbers Image: Display on Gutter		
		į	OK Cancel

Figure 32: Script Editor Options (Context Menu)

If you would like to display and edit different areas of your script in parallel, a second window can be opened. This window can be opened via the rectangular top right button (via the scrollbar). Move the mouse over the button and pull open another window from the top.



Figure 33: Second Window (Top Right Button)

4.2.2. Run Command

Using this command, the **Main() method** is exclusively called in the script asynchronously, assuming that it is available. There are two possibilities to execute the **Run command**:

Scri	pt	Visualization	Info
Z	Sc	ript editor	
~	Vi	rtual interface	
	GI	obal variables	
	Ru	in	
	St	op	

Figure 34: Possibility 1 (Script \rightarrow Run)

Import	
Export	
Create	
Run	
Stop	

Figure 35: Possibility 2 (Right click → Run)

4.2.3. Stop Command

A possibly running script will be stopped. There are two possibilities to stop the running script:

Scri	pt	Visualization	Infe
3	Sc	ript editor	
~	Vii	rtual interface	
	GI	obal variables	
	Ru	in	
	St	op	

Figure 36: Possibility 1 (Script → Stop)

Import	
Export	
Create	
Run	
Stop	

Figure 37: Possibility 2 (Right click \rightarrow Stop)

4.2.4. Integrating DLLs

External .Net assemblies can be integrated into the *script engine* of a Toolmonitor via the script settings. Their content will then be available in the Toolmonitor.

As can be shown in the figure below, three important .Net assemblies are already integrated by default for new projects. The *System.dll*, the *System.Drawing.dll* and the *System.Windows.Form.dll*.

Categories	Script	
General Logger Script Vsualization Status light Report Remoting User settings Tools	Net references System Drawing dli System Windows Forms.dt	
Corist antiana		- cod

Figure 38: .DLLs in a New Project

In order to integrate further .Net assemblies, the name of the **.DLL** is entered into the list and confirmed with "OK" as shown below. Here, the *System.Xml.dll* and a self - created *ClassLibraryBeispiel.dll* were integrated for demonstration purposes. It is important that the **.DLLs** are in the same directory as the Toolmonitor. To jump one line lower in the input window, "Ctrl + Enter" must be pressed.

Categories	Script	
General	.Net references	
- Logger - Script - Visualization - Status light - Report - Remoting - User settings - Tools	System.dll System.Drawing.dll System.Windows.Forms.dll System.Yani.dl ClassLibaryBeispiel.dll	
		Þ

Figure 39: Newly Added .DLLs

4.2.5. Global Variables

Using the **Global Variables** window, global variables can be created, deleted or changed.



Figure 40: Opening Global Variables

New variables can be added ("Add") or deleted ("Delete") via right click. Four data types (*Real, String, Vector, StringVector*) are available. The syntax for the respective data types can be seen in the example below.

Global varia	ables		
Тур	Name	Value	
Real	integer_test	20	
String	string_test	"hello world"	
StringVect	stringvector_test	["hello world"]	
Vector	vector_test	[20]	

Figure 41: Examples of Global Variables

4.2.6. Virtual Interface

Using the form for the **Virtual Interface**, the Toolmonitor functions which are available there can be executed and tested. It presents the interface for communication via the COM interface.



Figure 42: Opening the Virtual Interface

To control a Toolmmonitor via this interface, the Toolmonitor must be registered as COM server first.

Time	Command	Name	Value	Comment			
14:27:38.4	SetEvent	Adapter.GoHome		Goto home position			
14:27:44.3	GetValue	Adapter.IsMoving	0	Get is moving status			
14:27:51.5	SetValue	Adapter XPosition	100	Set X position			
14:28:02.6	GetValue	ControllerX.Veloci	100	Set target velocity			
⊟-Adapter	·	Name		Value	Con	nment	
Enable/	II : Enable controlle	Adapter Enable Al			Ena	ble controllers	
GoHom	e : Goto nome pose	Adapter GoHome			Got	o home position	1
Homing	Start homing	Adapter Halt			Qui	ck stop	
IsMovin	. Get is moving st	Adapter.Homing			Star	t homing	
IsBefer	nced : Get is refer	Adapter.IsMoving			Get	is moving statu	s
SetPosit	ion : Goto DUT 1	Adapter.IsReferenced	ł		Get	is referenced s	tatus
SetPosit	ion : Goto home vi	Adapter Set Position		DUT1	Got	o DUT 1	
- Shutdov	vnAll : Shutdown o	Adapter.SetPosition		X/Y Home	Got	o home via pos	tion inte
- XPositio	n : Set X position	Adapter.ShutdownAll			Shu	tdown controlle	irs.
-YPositio	n : Set Y position	Adapter XPosition		100	Set	X position	
- Z1Positi	on : Set Z1 position	Adapter YPosition		200	Set	Y position	
Z2Positi	on : Set Z2 positior 🖕	Adapter Z1Position		300	Set	Z1 position	
<	•	* [•
Name Cor	trollerX.Velocity	(Comment Se	t target velocity	Write	Update	Up
					12: 12	10-10	_

Figure 43: Virtual Interface

All possible commands for the *Virtual Interface* can be displayed by right - clicking "Add Defaults" in the virtual interface.

Project Setup	Events Script	Visualiza	al interface					
ime	Command	Name	Value	Comment				
		Nam	e Value Select Write Read	-	Connert			
			Move up Move down					
			Delete Delete all					
			Load Save					
			Show tree view Show comments Comm					
lame			Copy all	omment		Wite	Add	100
							1	

Figure 44: Displaying the Defaults

Before adding the commands a second time, they should be deleted by right - clicking "Delete All". Otherwise, every command will appear twice in the list.

ClearEvents Name Value Commert CurrentPreset EvertCourt SetEvert SetEvert GlobalVanables CommertMexet SetEvert SetEvert GlobalVanables CommertMexet SetEvert SetEvert GlobalVanables CommertMexet SetEvert SetEvert GlobalVanables CommertMexet SetEvert SetEvert GlobalVanables EvertCourt GlobalVanables SetEvert GlobalVanables EvertCourt GlobalVanables Move up GlobalVanables SetEvert Read Present Preset GlobalVanables GlobalVanables SetEvert SetEvert GlobalVanables SetEvert SetEvert SetEvert Bible SetEvert SetEvert SetEvert SetEvert HedmanTerru HedmanTe	Time	Command	Name	Value	Comment				
Current Preset Celevits SetEvent Global/Variables Current Preset SetEvent Global/Variables SetEvent Current Preset Global/Variables SetEvent Move up SetString Global/Variables Move down SetStringStream Global/Variables SetEvent Hide/ManiMeru Interface Type SetEvent Interface Type SetEvent Current Preset Interface Type SetEvent SetEvent Global/Variables SetEvent Current Preset Global/Variables SetEvent Current Preset Interface Type SetEvent Current Preset Global/Variables SetEvent Current Preset<	ClearEvents		Name			Value	Comment		
Cummt/Prest Select Select GlobalVariables EvertCourt Select GlobalVariables GlobalVariables Delte GlobalVariables GlobalVariables Delte GlobalVariables GlobalVariables Delte GlobalVariables GlobalVariables Get GlobalVariables GlobalVariables GlobalVariables GlobalVariables GlobalVariables GlobalVariables Read GlobalVariables Read GlobalVariables Read GlobalVariables Read GlobalVariables Selfsen	- Current Preset	ſ	ClearEvents				SetEvent		1
th: SetStmgStream BCDEF SetEvent HideMainMenu HideMainMenu Delete InterfaceNames InterfaceNames Delete all InterfaceNames InterfaceNames GetValue InterfaceNames InterfaceNames GetValue InterfaceNames IsVaidCardName.Globs Add defaults GetValue GetValue GetValue IsVaidCardName.SetSert LogLavel Save MainForm LogLevel Save Sotpt MainForm.Form.Accept Show tree view StringStrate Show comments GetValue Sotpt MainForm.Form.Accept Show comments Copy Write Add Use	GlobalVanables GlobalVanables Delete Exists Get Get Get Get SetReal SetStream		CurrentPresst EventCount GlobalVariables.Dek GlobalVariables.Get GlobalVariables.Get GlobalVariables.Set GlobalVariables.Set	ote. ts." Te Typ Res Stre Stre	Select Write Read Move up Move down Sort	23 3C	SetVatue, Ge GetValue Deletes the v Checks, if the Read the value Returns the to Sets the globu Sets the globu Sets the globu	String ariable "Test 1 is variable "Test ue of the globa ype of the varia al variable "Te al variable "Te al variable "Te	" (SetEx t1" exis al variab able "Te est1" to 1 est3" to 1 est2" to 1
Hidd MamMenu Interface Names Delete all Get String Interface Names Interface Type. Interface Get Value. Get String Interface Type IsValidCardName. Global Add defaults Get Value. IsValidCardName IsValidCardName. Script Load Get Value LogLevel IsValidCardName. Virtue Load Get Value NamForm LogLevel Save SetValue. NextSvert MainForm. Form. Accept Save SetValue. Script MainForm. Form. Accept Show tree view SetValue. GetValue. Script Show tree view SetValue. GetValue. Show tore view SetValue. GetValue. GetValue. Script MainForm. Form. Accept Show tore view SetValue. Show tore view SetValue. GetValue. GetValue. Show tore view Show tore view SetValue. GetValue. Show tore view Show tore view SetValue. GetValue. Show tore view SetValue. GetValue. Value. Status Copy Virte Add Ub	E Set String Stri	mea	GlobalVariables.Set: HideMainMenu	Star	Delete	BC DEF	Sets the globi SetEvent	al variable "Te	ist4" to 1
Interface Type Interface Type Interface Type Get Value Get String Interface Type Invelface Type Invelface Type Get Value Get Value Get Value Invelface Type IsValidCardName Get Value Get Value Get Value Get Value Invelface Type IsValidCardName Get Value Get Value Get Value Get Value Invelface Type IsValidCardName Sorpt Load Get Value Get Value Invelface Type Invelface Type IsvalidCardName Sorpt Save Set Value Get Value Invelface Type Invelface Type Invelface Type Sorpt Show tree view Set Value Get Value	HideMainMenu		InterfaceNames		Delete all		GetString		
Svard Softwine Soft Soft	Interface Type		InterfaceType.Interf IsValidCardName.Gi	oba	Add defaults		GetValue, Ge GetValue	tString	
ResetToolmonitorLookident MainForm.Form.Accest Show tree view SetValue, GetValue, GetString Brogit Show MainKenu MainForm.Form.Accest Show tree view SetValue, GetValue, GetString Show MainKenu MainForm.Form.Accest Show comments SetValue, GetValue, Get	 Is valocard valie LogLevel MainForm NextEvent 	•	IsValidCardName.Sc IsValidCardName.Vi LogLevel	crip(itua	Load Save		GetValue GetValue SetValue		
Copy Name Comment Comment Library III Write Add Library	Reset Toolmonito	orLockident	MainForm.Form.Acc MainForm.Form.Acc MainForm.Form.Acc	ess ~	Show tree view Show comments		SetValue, Ge SetValue, Ge SetValue, Ge	tValue, GetStri tValue, GetStri tValue, GetStri	ing ing
	Name		Commer	vt	Copy		Write	Add	Up

Figure 45: Deleting All Commands

A more detailed documentation of the available commands and parameters can be found in the class documentation of the **General Help** and in the specific help.

Analyze Script editor	· Virtual interf	lace				•
ime Command Na	me	Value	Comment			
Analyze - ClearEvents : SetEvent	Name	Value		Comment		
-CumentPreset : SetValue, GetShing ExerctCount : GetValue 9 Globa/Variabiles - HidsManMonu : SetEvent - httefaceNames : GetShing	Analyze Fom Analyze Fom Analyze Fom Analyze Fom Analyze Fom	laa . laa . laa . laa .		SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting		
- Interface Type - IsValidCardName - LopLevel : SetValue - MainForm	Analyze Form Analyze Form Analyze Form Analyze Form	lan har har		SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting		
- NextEvent : GetString - ResetToolmonitorLockident : SetEvent - Script - Script	Analyze Form Analyze Form Analyze Form	han han han		SetValue, GetValue, GetString SetValue, GetValue, GetString SetValue, GetValue, GetString		
ToolmonitorLookident : SetValue, GetStin ToolmonitorTestMode : SetValue, GetStin TraceLevel : SetValue	Analyze Form Analyze Form Analyze Form			Set Value, Get Value, Get String Set Value, Get Value, Get String Set Value, Get Value, Get String		
- UserScreen 1	Analyze Form Analyze Form Analyze Form	lar Lar Ion		SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting		
	Analyze Form Analyze Form Analyze Form	haa haa haa		SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting SetValue, GetValue, GetSting		
	reading round	98)		and room, our route, decising	 	

Figure 46: All available Commands for the Virtual Interface

Double - clicking on an element (here: *Global. Variables. SetReal. Test1*) applies its properties to the **Name, Value** and **Comment** fields (bottom). In the example here, the value of the "Test 1" variable is set to 10. The commands for reading and writing can be sent using the **Read and Write** buttons. Alternatively, the *Read* and *Write* commands can be reached via right mouse click. The return values and results of the instruction then appear in the window at the top of the form.

41828.6 InterfaceType GlobalVariables.S. SetValue Sets the global vanable 41828.6 SetValue GlobalVariables.S. SetValue Sets the global vanable 41828.6 SetValue GlobalVariables.S. SetValue Sets the global vanable ClearEventa: SetValue SetSite SetSite CurrentPreset: SetValue GlobalVariables GetType. Text 1 Read the value of the global vaniable Text 1: (GetValue) CobalVariables GlobalVariables GetType. Text 1 Read the value of the global vaniable Text 1: (GetValue) GlobalVariables GlobalVariables SetSites of text 1 Read the value of the global vaniable Text 1: (GetValue) GlobalVariables GlobalVariables SetSites global vaniable Text 1: Delete the value of the global vaniable Text 1: Delete the value of the global vaniable Text 1: SetSite global vaniable Text 1: SetSites global vaniable Text 1: SetSites global vaniable Text 2: to the string value: 7ABC': "DEF". "XYZ"] (SetSitingSites m) GlobalVariables GlobalVaniables SetSite global vaniable Text 1: SetSite global vaniable	Time	Command	Name	Value	Comment		
ClearEvents: SetEvent Name Value Comment CurrentPreset: SetValue, GetString GlobalVatables GetTrest 1 Read the value of the global variable "Test 1" (GetValue) GlobalVatables GlobalVatables GetTrest 1 Read the value of the global variable "Test 1" (GetValue) GlobalVatables GlobalVatables GetTrest 1 Resums the type of the vatable "Test 1" (GetValue) GlobalVatables GlobalVatables GetTrest 1 Sets the global variable "Test 1" (GetValue) GlobalVatables GlobalVatables SetString Test 2 ABC Get Test 1: Checks, if the variable "Test 1" exists GlobalVatables SetString Test 2 Get GlobalVatables SetString Test 2 ABC GetString Test 1: Read the value of the global variable GlobalVatables SetString Stream Test 4 GetString Test 1: Set the global variable Test 1" to the string vector: ["ABC", "DEF", "XYZ"] (SetStringStream) HideMainMenu SetEvent GetValue, GetString GetValue, GetString I = Test 1: Set the global variable "Test 1" to the ValueCarlName Strip GlobalVariables GetValue GiolaVariables SetString Test 3" to the ValueCarlName Strip GlobalVariables GetValue GetValue GetValue GetValue GetValue	4:18:28.6 4:18:28.6	InterfaceType SetValue	GlobalVariables.S GlobalVariables.S	SetValue 10	Sets the global va Sets the global va		
ClearEvents SetEvent CurrerPreset: SetEvant CurrerPreset: SetEvant CurrerPreset: SetEvant Global Variables: Get Global Variables: SetEvant Global Variables: Get Defet Image: SetEvant Global Variables: SetEvant Global Variable: SetEvant HiedMainMenu SetEvent HiedMainMenu SetEvent HiedMainMenu SetEvent InterfaceName GetValue SetEval Global Variable: ValuC-ardName: Global Variable: SetEval Global Variable: Global Variable:							
- EvertCourt : Get Value Global/Variables Get Type. Test 1 Returns the type of the variable "Test 1" [De-Unknown, 1=Real, 2=String, 3=Vector, 4=String/Vector] (Get Value) - Delete Global/Variables SetString Test 1 1 Sets the global variable "Test 1" (De-Unknown, 1=Real, 2=String, 3=Vector, 4=String/Vector] (Get Value) - Exist - Test 1: Deletes the variable "Test 1" (SetEver - Get 1 Sets the global variable "Test 1" to the real value: 1 [SetValue) - Get - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 1: Read the value of the global variable "Test 1" rest - Test 2: Not the rest - Test 1: Read the value of the global variable "Test 1" rest - Test 2: Value SetString SetString SetString SetString SetString SetString SetString - Test 2: Value SetString - Test 2: Value SetString - SetString <td>ClearEvents CurrentPrese</td> <td>SetEvent : SetValue, GetString</td> <td></td> <td>Name Global Variables (</td> <td>Get Test 1</td> <td>Value</td> <td>Comment Read the value of the clobal variable "Test1" (GetValue)</td>	ClearEvents CurrentPrese	SetEvent : SetValue, GetString		Name Global Variables (Get Test 1	Value	Comment Read the value of the clobal variable "Test1" (GetValue)
Detecte L Test 1: Detects the variable "Test 1" (SetEvers Exists Detects Detectts Detects Detects Detects Detect Detects	- EventCount GlobalVariab	GetValue es		Global Variables	Get Type.Test1 Set Type.Test1	4	Returns the type of the variable "Test 1" (De value) Returns the type of the variable "Test 1" [0=Unknown, 1=Real, 2=String, 3=Vector, 4=StringVector] (GetValue) Sets the off-out variable "Test 1" to the real values 1 (SetValue)
Get Type Get Type Get Type Get Type Get Type Get Stread Get Stread Get Stread Get Stread Get Stread Get Stread Get Stread Get Value Get Value	E- Exists	: Deletes the variable	"Test 1" (SetEve ≡	GlobalVariables. GlobalVariables. GlobalVariables.	Set Stream. Test 3 Set String. Test 2 Set String Stream. Test 4	123 ABC ABC DEF	Sets the global variable "Test 3" to the vector: [1, 2, 3] (SetStream) Sets the global variable "Test 2" to the string value: "ABC" (SetValue) Sets the global variable "Test 4" to the string value: "ABC" (SetValue)
GetType InterfaceType.InterfaceName GetValue, GetString InterfaceType.InterfaceName GetValue, GetString InterfaceType.InterfaceName GetValue InterfaceType.InterfaceName SetValue InterfaceType.InterfaceName SetValue InterfaceType.InterfaceName SetValue InterfaceType.InterfaceName	⊟-Get	: Read the value of the	e global variable	HideMainMenu InterfaceNames			SetEvent GetString
SetReal Is Test 1: Sets the global variable "Test 1" to th Is ValidCardName.Script GetValue SetStream Is ValidCardName.Vitual interface GetValue SetString SetString SetString Image: Test 2: Sets the global variable "Test2" to th MainForm.Form AccestButton SetValue. GetValue. Ge	GetType	: Returns the type of th	ne variable "Tes	InterfaceType.In IsValidCardName	terfaceName GlobalVariables		Get Value, Get String Get Value
SetStream LogLevel SetStream LogLevel SetStream LogLevel SetStream LogLevel SetStream LogLevel SetStream SetValue SetValu	SetReal	: Sets the global variat	ble "Test1" to th	IsValidCardName IsValidCardName	Script Virtual interface		GetValue GetValue
SetStmg Test2 : Sets the global variable "Test2" to th MainForm.Form AccessibilityObject SetValue, GetValue, GetV	E-SetStrea	n I : Sets the global variat	ole "Test3" to th	LogLevel MainForm.Form.J	AcceptButton		SetValue SetValue, GetValue, GetString
	E- SetString	: Sets the global variat	ole "Test2" to th ↔	MainForm.Form	AccessibilityObject		SetValue, GetValue, GetString CatValue, GetValue, GetMing

Figure 47: Read / Write Commands

4.3. Report

Using the menu **Report**, a user - defined report can be created depending on the Toolmonitor and the settings in the setup. To activate this menu item, the number of **Report Views** must be created in the options of the Toolmonitor. These are independent forms, in which different reports can be designed and created via **FastReport**.

Options	
Categories	Report
General Logger Script Status light Report Remoting User settings Tools Visualization	Report Number of report forms B
Report options	OK Cancel

Figure 48: Changing the Number of Reports

To create a **new Report View** select the *Report tab* under **Setup** \rightarrow **Options** in the menu bar as described here and adjust the desired number of forms in the selection box. In the figure below, three forms were created. After creation, the additional *Reports* option appears in the menu bar. Here, it can be selected between the created forms.



Figure 49: Opening the Report View

After the created **Report View** was opened as shown above, an empty interface appears with the menu bar shown below. Four buttons are available at the top of the menu: *Refresh, Load report, Designer* and *Export.*

The *Report preview* can be refreshed via the **Refresh** button. For example, if a time is displayed or a change was made in *Designer*, it can be refreshed here. For this purpose, the *Designer* must be exited. Otherwise, an error message appears at the bottom of the screen.

CED Por	ort view						1.07081		
Refresh		oad report	Desig	gner Exp	ort				
Drucken		Speicherr	• 🖂		7 11	4 4	1 von 1	Þ	M Schliesser

Figure 50: Refresh Button

An already available report can be loaded via the **Load report** button. During this process, the *Designer* must be exited. Otherwise, an error message appears at the bottom of the screen.

Report	view 1				
Refresh	Load report	Designer Ex	port		
Drucken 📔	Speichern	· 🖂 👬 🖂 🦉		1 von 1 🕨	Schliesser

Figure 51: Load Report Button

The **FastReport Designer** opens via the **Designer button**. Using the *Designer*, layouts for new reports can be created as well as existing reports changed and / or edited. A detailed description of *Designer* and all available components can be found in the *FastReport User's Manual* under

https://www.fast-report.com/public download/FRNetUserManual-en.pdf.

CO Report view	1	in Report into t	чыр	
Refresh Loa	ad report Designer	Export		
🚔 Drucken 📑 🔚	Speichern 🔹 🖂 🚮 🛅 🌽		1 von 1 🕨 🕅	Schliesser

Figure 52: Designer Button

Using the **Export** function, the report can be directly exported (see example "Beispielbericht.html" below). If no specific path is indicated, but only the file name, the export is stored in the directory, where the Toolmonitor can be found.

Project Setup	Events Se	cript Visualiz	stion Report	t Info H	elp	
Report vi	ew 1	Designer	Export	Beisnielbe	richt html	1
Douckers				beispielbe	1 von 1	M Schlerren

Figure 53: Export Button

In addition to the four described buttons, several icons are available in the toolbar. The report can be printed via the *Print* icon as in any Windows application.

Project	Setup	Events	Script	Visualization	Report	Info	Help	
CO Re	port viev	v 1						
Refresi	h	oad report	Desi	gner Exp	oort			
	100				- 1000 i i			

Figure 54: Printing the Report

Already existing reports can be opened via the Open icon.

Project	Setup	Events	Script	Visualization	Report	Info	Help		
CO Re	port viev	v 1							
Refres	h L	oad report	Desi	gner Exp	ort				
Doucker		Speicher	• •		2 1	4 4	1 von 1	P N	Schliessen

Figure 55: Opening the Report

The created report can be saved in different formats using the *Save* icon. The following selections are supported: PDF, ODS, ODT, Excel, XML, RTF, HTML, text, CSV, BMP, Jpeg, Tiff, and Gif.

Project	Setup	Events	Script	Visualization	Report	Info	Help		
CO Rep	ort view	1							
Refresh	Lo	ad report	Design	ner D	opr				
Drucken	E B	Speichern	- 100				1 von 1	D N	Schliessen

Figure 56: Saving the Report

Using the Search icon, the report can be searched for keywords.

Project	Setup	Events	Script	Visualization	Report	Info	Help	
ato Rep	port viev	v 1						
Refrest		oad report	Desig	gner Ex	port			
Drucken		Speicher		23 T 21	3 1	4 4 1	1 von 1	Schliessen

Figure 57: Searching for Keywords

The report can be displayed in a frame using the Frame icon.

Project	Setup	Events	Script	Visualization	Report	Info	Help		
CO Re	port viev	v 1							
Refres	h L	oad report	Desi	gner Exp	ort				
Drucker		Speicher	• •		2 MI	4 4 1	1 yon 1	N NI	Schliessen

Figure 58: Showing / Hiding the Frame

The paper format or the report dimensions can be adjusted using the Setup Page icon.

Project	Setup	Events	Script	Visualization	Report	Info	Help	
ato Re	port viev	v 1						
Refres	h L	oad report	Desig	gner Exp	ort			
Drucker		Speicherr	• 🖂		a 💷 🖡	4.4.	1 von 1	Schliessen

Figure 59: Editing Page Layout

The Designer can be directly accessed and the report edited via the Edit Report icon.



Figure 60: Editing Report in the Designer

Using the Watermark icon, an image, or text can be added to the report as watermark.

Project	Setup	Events	Script	Visualization	Report	Info	Help		
Re Re	port viev	v1							
Refree	h []	nad report	Desi	oner Evr	vort				
Tierres		odu ropon		Augi Cret		111 1/2 ×			
Drucker	n 🔚 🗎	Speichen	n • 🖂	A 1 2	M 🔳 🛔		1 von 1	N N	Schliesser

Figure 61: Inserting a Watermark

This toolbar can be used to switch between all available pages of the report.

Project	Setup	Events	Script	Visualization	n Report	Info	Help		
CO Re	port viev	w 1							
Refres	h L	oad report	Desig	aner E	xport				
Drucker		Speicherr	• 🖂		17 m	4 4	1 von 1	Þ M	Schliessen

Figure 62: Displaying a different Page

The Report view form can be exited using the Close button.

Project	Setup	Events	Script	Visualization	кероп	Info	Help		
CO Rep	port viev	v1							
Refresh	n L	oad report	Desig	ner E	xport				
Drucken	1	Speicherr	• 🖂	A	301	N 4 1	1 von 1	P N	Schliessen

Figure 63: Closing the Report

5. Events

Here, the views for the log and trace messages can be accessed.

ſ	Events	Serial line
1	Log	gging
	Tra	ce

Figure 64: Events Menu

5.1. Logging

Using this menu, the log messages view for general events, warnings, errors, etc. is called up.

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
•					•

Figure 65: Log Monitor

5.2. Trace

Using this menu, the trace messages view (sent and received messages) is called up.



Figure 66: Trace Monitor

6. Debugger

Using the **Debugger**, you can debug scripts in the MCD Toolmonitor. The *Debugger* can be called directly from the script. This simplifies the creation of your C# functions. The user interface of the *Debugger* is shown in Figure 67.

VisuMonitor File Edit V	(Debugging) iew Debua	- Microso Tools	ft C W	LR Debugger				
			10000	- 5 5 -				
	a 🌳 🏂 (,≣ ~ <u>⊒</u> I	Hex					
Visu.ProjectS	cript.724.cs	1		•	×	Solution Exp	orer - Solution	* ‡ X
	Debug()	;						
	922 222					Solution	n 'VisuMonitor'	(0 projects)
	int sta	ervel :	JU; = 1	0.		🛓 🦾 🦢 Mis	cellaneous Files	
	int len	oth =	20.	0,			Visu.ProjectScrip	pt.724.cs
	100 100	igon i	20,					
0	int cou	inter =	st	art;				
-								
	for (in	it i = (D;	i < length; i++)	III			
	{							
	coun	ter +=	i	<pre>* interval;</pre>	_			
	}							
, , , , , , , , , , , , , , , , , , , ,								
1								
1					-			
•		11	Í	Þ				
Locals		– 4	×	Breakpoints				• 4 ×
Name	Value	Туре	*	New - 🗙 🧶 🌠 🐖 Column	s •			
🗄 🥥 this	{MCD.UserSo	MCD.User						
🥥 start	100	int		Name	C	ondition	Hit Count	
🥥 interval	10	int		H Visu.ProjectScript.724.cs, line 2	5 (no condition)	break always	
🧳 length	20	int		🗄 🔽 🥥 Visu.ProjectScript.724.cs, line 29	(no condition)	break always	
🥥 counter	0	int						
			Ŧ					
🐻 Locals 🗾 W	at 🚺 Im	- Thr	2	🖓 Call Stack 🔜 Modules 🗐 Output 📑	Brea	kpoints		
Ready						Ch1		INS
mouty			_	CH125 COT1		CHI		11.10 J

Figure 67: Debugger User Interface

6.1. Debugger Installation

The *Debugger* must be installed on your system. For this purpose, execute the provided installer *Microsoft.Net SDK.exe*.

You can also download the required installer from the Internet: <u>Download Microsoft.Net SDK 2.0</u>

Search for "Microsoft.Net SDK 2.0". You can use the Debugger after the installation.

6.2. Starting the Debugger

Please open the *context menu* in the *Script Editor* via right click to start the *Debugger*. You can then open the *Debugger* via the **Launch debugger** entry.

Load	
Save	
Create	
Run	
Stop	
Launch debugg	ler
Cut	Strg+X
Сору	Strg+C
Paste	Strg+V
Delete	Entf
Select all	Strg+A
Undo	Strg+Z
Redo	Strg+Y
Find	Strg+F
Replace	Strg+H
Goto line	Strg+G
Code snippets	Strg+W
Options	

Figure 68: Starting the Debugger

There are two possibilities to stop running your script in the *Debugger*. You can set a **Breakpoint** in the *Debugger*. The use of *Breakpoints* is described in *chapter Using Breakpoints*.

It is also possible to define the start point for the *Debugger* in the script using the **Debug() command** (see figure 69, line 19). The *Debugger* starts at this position. This way, you can start the *Debugger* in certain situations, e.g., after inquiring a condition in an "if instruction".



Figure 69: Debug Command in the Script Editor

You can now start actually debugging your source code. For this purpose, switch into the *Script Editor* and / or your *visualization interface* and start the desired code section buy operating a control or starting the script via "run". Now, switch into the *Debugger*. If you inserted a *Breakpoint*, you can now see a yellow arrow at this position. It signals that program processing stopped here.



Figure 70: Stopping the Debugger with a Breakpoint

If you entered a Debug() command, the debugging is started at this position. This is marked by a green arrow.

```
namespace MCD
{
   public partial class UserScript : VisuScript
      public void Sample Button1_Click()
      ş
          // Call debugger
         Debug();
         int start = 100;
         int interval = 10;
         int length = 20;
         int counter = start;
         for (int i = 0; i < length; i++)</pre>
         ł
             counter += i * interval;
         }
      }
   }
```

Figure 71: Starting Debugging at the Start Position

6.3. Debugging the Script



Figure 72: Debugger Toolbar

The easiest way of navigating the *Debugger* is via the **Debug toolbar** (see figure). The following table shows an overview of the most important functions:

	Run - Runs the program until the end or the next breakpoint
	Stop - Stops the debugger
SI	Single step - The debugger's instruction pointer is set to the next instruction
Ç≡	Process step - The debugger's instruction pointer is set to the next instruction (Functions are processed in one step)
Č_	Return - The instruction pointer returns to the calling function
Hex	Hexadecimal display - The representation of variables can be switched between decimal and hexadecimal

Figure 73: Function Overview of the Toolbar

6.4. Using Breakpoints

Breakpoints can be used for stopping script execution at a certain point. A *Breakpoint* can be activated via mouse click on the gray column on the left of the desired command. To deactivate the *Breakpoint* simply click it again.

Once you started the *Debugger* and set a *Breakpoint*, you can activate the execution of all commands up to the *Breakpoint* using the button *Next*. As can be seen in the figure, the *Debugger* stops with a yellow arrow at the desired command. You can now analyze the variables and, e.g. debug further commands in *step mode*.



Figure 74: Stopping the Debugger with a Breakpoint

6.5. Monitoring Variables

Using the *Debugger* variable values can be displayed and changed. The easiest way of displaying a variable value is moving the mouse pointer over the desired variable name. The current value is then displayed automatically.

```
int interval = 10;
int length = 20;
int counter = start;
for (int i = 0; i < length; i++)
{
    counter += i * interval;
}
</pre>
```

Figure 75: Displaying variable Values using the Mouse Pointer

Furthermore, you can display the currently applicable variables in the **window Local**. You can also add custom variables for monitoring in the **window Watch**. For this purpose, simply add the variable name to the **Name column**.

To change the value of a variable, you can adjust the entry in the **Value column** to the desired value. Please be aware that this is a manual intervention into the software sequence, which will not be available later on in the source code. Thus, this intervention should only be used for debugging purposes.



Figure 76: Monitoring Variables